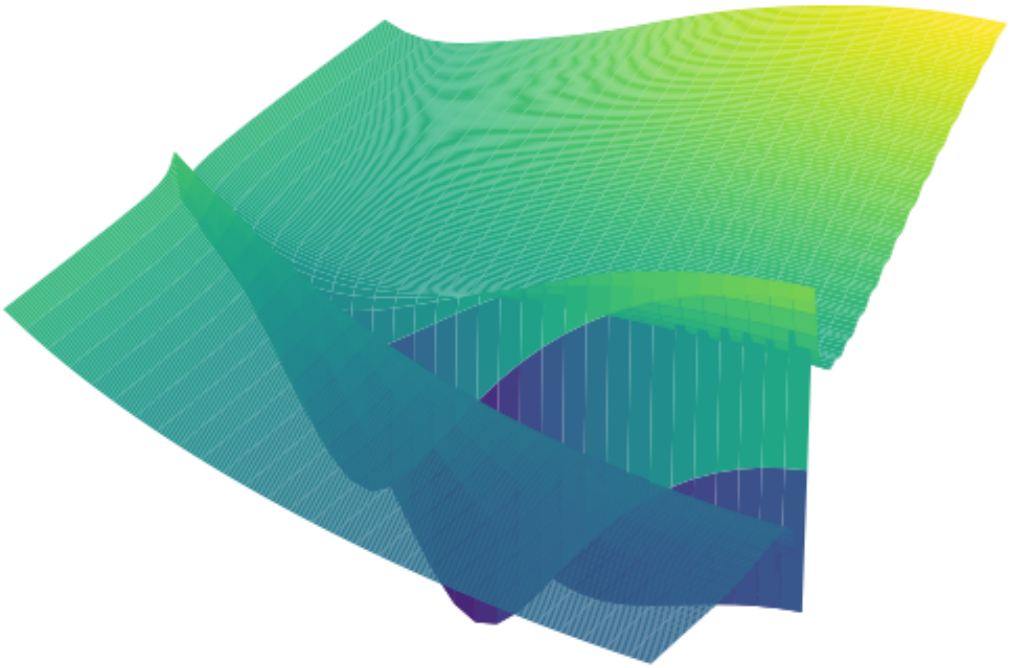
Identification of Nuclear Resonances through Machine Learning Models

PHYS8207 Research Project

Matthew Foster
Australian National University

October 2024



1 Abstract

This research explores resonance identification in low-energy nucleon and light-ion scattering, applying convolutional neural networks (CNNs) to examine resonance properties with machine learning. The study presents an approach where nuclear resonances—characteristic, quasi-bound states within scattering processes—are modelled using a multi-stage CNN inferencing architecture to predict resonance features such as energy, width, and nuclei angular momentum. Through synthetic datasets designed for model training, an array of pre-processing methods, and a multi-stage model pipeline, the CNN's ability to discern relevant features within scattering images shows excellent predictive capability. The findings suggest a capacity for CNNs to identify resonance traits with a level of precision that aids in interpreting nuclear data and addresses complexities in traditional scattering analysis. This research contributes to emerging methods in nuclear physics, augmenting interpretative capabilities where machine learning intersects with classical theoretical frameworks.

2 Introduction

In nuclear physics, resonance interactions in scattering provide insight into the intrinsic structure of atomic nuclei. Resonances are most commonly observed when the energy of an incident nucleus synchronises with discrete excitation levels of the nucleus, forming quasi-stable states. These states exhibit distinct scattering cross-sections that elude to behaviour of the nuclei. Standard identification methods, however, often rely on parameter-heavy calculations, which are computationally intensive and limited by underlying assumptions and the breadth of available data. Machine learning, with its flexibility and analytical power, brings a new approach to these issues, finding patterns and extracting features in data that may otherwise require exhaustive theoretical modelling. This study applies convolutional neural networks to recognise resonance features within generated cross-sectional data, training the model to learn complex associations between image-based scattering data and resonance parameters. This exploratory approach provides an alternate foundation for nuclear data analysis by leveraging machine learning's adaptability to supplement traditional computational techniques. Through these models the resonance properties such as energy, width, and angular momentum are mapped and allow the model to handle complex variations in scattering interactions. The results illustrate the potential for machine learning in nuclear resonance analysis, as the model extracts nuanced resonance information in an otherwise challenging analytical space.

This report is structured such that the reader would be familiar with the nuclear science concepts such as scattering, cross-sections, resonances, and the quantum mechanics principles that dictates the behaviour of these interactions. This will be covered in Chapter 3 and lead to the premise of the project. However, it is assumed that the reader will have little to no previous familiarity with machine learning models. As such, we give a detailed description of the methods used in the machine learning models that is developed. Chapter 4 will introduce the building blocks of a neural network and the many performance metrics, both of which are an expansive topic in itself. These will culminate in a prototyping model that demonstrates the proof-of-concept in Section 4.4 before detailing the project development of the final multistage inferencing model in Chapter 5.

3 Understanding the Characteristics of Resonances in Low Energy Nucleon and Light-Ion Scattering Interactions

Resonances are fundamental phenomena in nuclear physics that provide insights into the interactions and structure of atomic nuclei. Understanding the behaviour of resonances in scattering interactions is key for interpreting experimental data and advancing theoretical models of nuclear forces. The concepts within this chapter are intended to provide a brief synthesis of the characteristics of scattering interactions and their deviation from classical mechanics. This summary has been drawn from the following references, where far more comprehensive information on the matter can be found [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12].

3.1 Low Energy Scattering and Cross Sections

This section aims to provide an introduction into fundamental concepts in nuclear collisions. It should address:

- Differentiating nuclear scattering events from solid body collisions
- The various cross sections in scattering events.
- The scattering angle and the cause of its non-uniform distribution.
- The linkage between the angular distribution and the angular momentum of the scattered nucleus.
- An introduction into phase shifts.

3.1.1 Defining Scattering Events

In nuclear physics, a scattering event refers to the interaction of two nuclei, causing a change in their momenta, excitation or identities. Unlike solid body collisions, where macroscopic objects physically collide and exchange energy through purely the Coulomb forces, scattering at the subatomic level involves interactions mediated by additional fundamental forces. These interactions may include electromagnetic, strong nuclear, or weak nuclear forces, affecting the momentum and energy states of the involved nuclei. The quantum mechanical nature of these interactions necessitates a probabilistic description, distinguishing them from the deterministic outcomes observed in classical solid body collisions.

3.1.2 Scattering Cross-Sections

The cross section quantifies the likelihood of a scattering event between nuclei. It represents an effective area that quantifies the probability of interaction as a function of parameters like energy and scattering angle. Various cross sections are defined to characterise different aspects of scattering processes. The *total cross section* encompasses all scattering events, regardless of the final state. The *differential cross section* provides a more detailed description, expressing the probability of scattering into a specific solid angle, and the angular dependence offers insights into the nature of the interaction. Other important cross sections include the *elastic cross section*, describing events where kinetic energy is conserved and nuclei remain in their initial states, and the *inelastic cross section*, which accounts for processes where nuclei are excited to higher energy states or new nuclei are produced.

3.1.3 Influence of Angular Momentum

The angular momentum of an incident nucleus significantly influences the scattering cross section in nuclear physics. When nuclei interact at quantum scales, their behaviour is governed by the conservation of quantised angular momentum. The angular momentum determines the contribution of different partial waves to the scattering process, each characterised by an angular momentum quantum number l.

At lower energies, scattering is predominantly governed by the lowest angular momentum states, particularly the s-wave ($l = 0$). Higher angular momentum states are hindered by the centrifugal barrier, represented by the centrifugal potential

$$V_{\text{centrifugal}}(r) = \frac{\hbar^2 l(l+1)}{2\mu r^2},$$

where $\hbar$ is the reduced Planck constant, μ is the reduced mass, and r is the separation between the nucleus. This barrier reduces the likelihood of nucleus with higher angular momentum penetrating the region where they can interact via nuclear forces, diminishing their effect on the overall cross section.

3.1.4 Angular Distribution and Phase Shift

The angular distribution of scattered nucleus is also directly affected by the incident nucleus's angular momentum. The differential cross section $\frac{d\sigma}{d\Omega}$ depends on scattering amplitudes that include phase shifts δ_l associated with each l. In this framework, the scattering amplitude is expressed as a sum over partial waves, each associated with an angular momentum quantum number l. The phase shift δ_l for each partial wave encapsulates the effect of the interaction potential on the scattering process, modifying the phase of the outgoing wave relative to the incoming wave.

Mathematically, the scattering amplitude $f(\theta)$ for uncharged particles can be written in terms of phase shifts as

$$f(\theta) = \frac{1}{k}\sum_{l=0}^{\infty}(2l+1)e^{i\delta_l}\sin\delta_l P_l(\cos\theta) \tag{1}$$

where k is the wave number of the incident nucleus, P_l are the Legendre polynomials, and θ is the scattering angle. For charged particle scattering, this expression needs to be generalised to include the phase shift from the Coulomb interaction. However, the resultant expression is similar in its dependence on the phase shift. The differential cross section is then given by

$$\frac{d\sigma}{d\Omega} = |f(\theta)|^2.$$

where the total cross section can then be attributed to

$$\sigma_{total} = \int \frac{d\sigma}{d\Omega} d\Omega$$

The phase shifts δ_l encodes information about how each partial wave is affected by the scattering potential, including any resonant behaviour. Note that the total angular integrated cross-section is not defined in the charge particle case due to the long-range of the Coulomb interaction. For charged particle elastic scattering, the total cross section as defined here would be infinite, but is defined for non-elastic reactions.

Resonances cause the phase shift to rapidly change by $\pi/2$ radians. Given the sin term in δ_l this results in a rapid change in the cross section indicating the temporary formation of a quasi-bound state within the potential.

3.2 Resonances

This section is aimed at detailing the resonance phenomena observed in low energy particle scattering events. It should address:

- An understanding of how resonances form and the quasi-bound state.
- Resonance characterisation by phase shift.
- The effect resonances have on the cross section of scattering nuclei.
- The linkage between angular momentum and the angular distribution of scattering.

3.2.1 Understanding the cause and behaviour of resonances

Resonances in nuclear scattering occur when the energy of the incident nucleus coincides with the compound nucleus's energy levels during the interaction. At these specific energies, the incident nucleus and the target nucleus form a quasi-bound state, leading to a significant enhancement in the scattering cross section. This phenomenon occurs because the system's total energy nears one of the discrete excited energy states of the compound nucleus, allowing for a temporary trapping of the incident nucleus within the nuclear potential well. This can be referred to as a quasi-bound state.

Resonances are characterised by rapid changes in the phase shift δ_l of a particular partial wave l as a function of energy. Near a resonance, the phase shift varies sharply and can be described by the *Breit-Wigner formula*,

$$\delta_l(E) = \arctan\left(\frac{\Gamma/2}{E_R - E}\right) \tag{2}$$

where E_R is the resonance energy, Γ is the resonance width, and E is the energy of the incident nucleus. The Breit-Wigner formula models the resonant behaviour of the phase shift and, consequently, the scattering amplitude.

The resonance width Γ represents the energy range over which the resonance occurs and is inversely related to the lifetime τ of the quasi-bound state through the uncertainty principle,

$$\Gamma = \frac{\hbar}{\tau}.$$

A narrower resonance width indicates a longer-lived state, while a broader width corresponds to a shorter-lived state. The width is influenced by factors such as the coupling strength between the incident nucleus and the target nucleus, and the available decay channels. Stronger coupling and more decay pathways typically lead to a larger Γ, reflecting a higher probability of the quasi-bound state decaying back into scattering states.

3.2.2 Resonance Effects on Scattering Cross-sections

The presence of a resonance significantly affects the scattering cross-section. Near the resonance energy E_R, the differential cross section $\frac{d\sigma}{d\Omega}$ experiences a pronounced increase due to the enhancement of the scattering amplitude $f(\theta)$ due to the $\sin\theta$ term. Incorporating the Breit-Wigner formula into the expression for the scattering amplitude and assuming only a single partial wave contributes, the cross section near a resonance can be approximated as

$$\sigma(E) \approx \frac{\pi}{k^2}(2l+1)\frac{\Gamma^2/4}{(E-E_R)^2 + \Gamma^2/4},$$

where k is the wave number of the incident nucleus. This assumption assumes a single angular momentum dominates. The Lorentzian shape of the cross section as a function of energy reflects the resonant enhancement and is characteristic of the resonance phenomena in scattering processes.

As observed in the scattering amplitude formula, the scattering angle is dependent on the angular momentum. Since the effect of resonances is closely tied to the angular momentum l of the incident nucleus, it implies that the resonance features across the angular distribution is tied to the scattering angle. Higher l values correspond to more complex angular distributions with increased numbers of nodes (zero crossings) in $P_l(\cos\theta)$, leading to more pronounced angular variations in the cross section. As a result, resonances with higher angular momentum can enhance scattering at larger angles, although the centrifugal barrier generally suppresses their contribution at low energies.

3.3 Scattering Amplitudes

This section details the scattering amplitude imaging as a function of nucleus energy and scattering angle. It should address:

- The standard methods for evaluating the scattering process.
- Experimental and evaluation cross-sectional databases.
- Issues arising in experimental databases arising from sparsity, error and bias.
- Spectral representation of cross-sectional data and visual intuition of the feature space
- The premise for machine learning application of the spectral representation.

3.3.1 The R-Matrix

Resonances have a interference on the scattering cross-section that extend across the scattering angle. Methods such as the R-Matrix [13] are considered the gold standard for understanding the scattering process over the resonance region and have been traditionally used for this purpose. However, they have many challenges associated with their use, from computational complexity, reliance on parameter fitting, boundary approximation, and limitations arising from overlapping resonances and background contributions. The time cost associated and expertise required to perform the task is also considerable.

3.3.2 Experimental and Evaluated databases

The Ion Beam Analysis Nuclear Data Library (IBANDL) is a comprehensive repository of evaluated and experimental cross-section data [14]. The library facilitates accurate quantitative analysis for the interaction probabilities between incident ions and target nuclei over a range of energies, as measured by the angle of the detection device. For example, consider the $^{12}C(\alpha,\alpha_0)^{12}C$ interaction, where 154 datasets are publicly available. Isolating datasets at 165° reveals measured scattering cross-sectional data for given incident α particle in Figure 1. Included is the evaluated data from SigmaCalc which provides cross-sectional predictions by combining theoretical calculations with an analysis of discrepancies from benchmarked and measurement experiments [15]. Intuitively, the observed peaks can be attributed to the resonances. However, this plot represents a slice of angular distribution and imaging the energy-theta cross-sectional data illustrates the tension with this intuition when neighbouring resonances interfere.

3.3.3 Spectral Representation of the cross-Section

The images in Figure 2, 3, 4, 5 are simulations of the energy-theta cross-sectional values from the Python twobody module [16]. The data has been divided by the Rutherford cross-section to smooth the manifold and reduce the angular dependence, and a number of resonances have been implanted into model. See Section 4.4.2 for more details. The single resonance image (Fig 2) has a discernible effect on the cross section manifold. The narrow resonance with a 72 keV width and 0 angular momentum creates a single thin fissure with little variation in the fissure energy centroid across the angular distribution. In contrast, the 5 resonance image shows the complex interference from various overlapping resonances, with each resonance creating $l+1$ peaks across the angular distribution. Notable, there is also larger shifts in the fissure's energy centroid across the angular distribution.

E_{res} (MeV)	Γ_{res}	l_{res}
2.6736	0.0725	0

Table 1: Resonance parameters of the single resonance images.

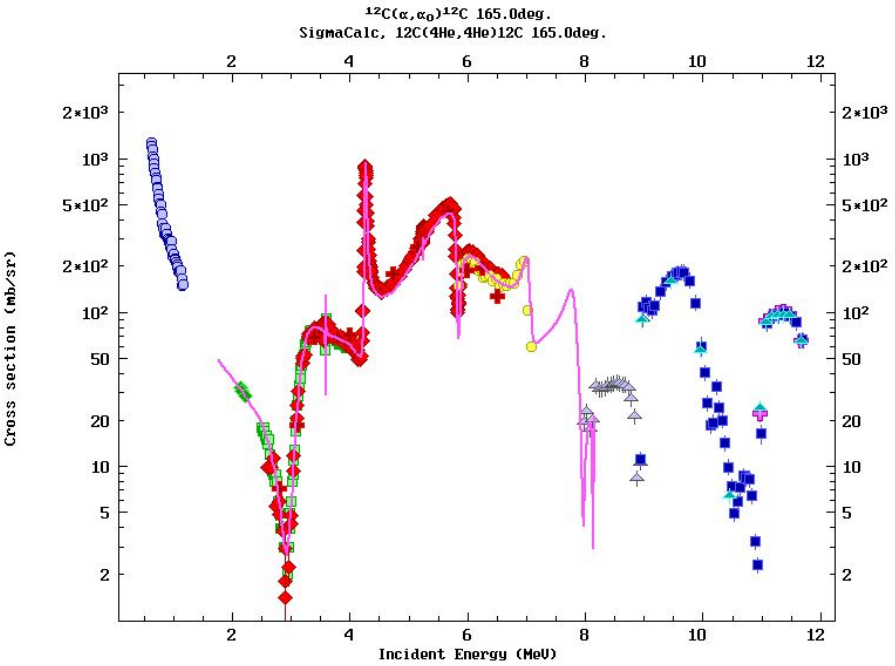


Figure 1: lin-log plot of $^{12}C(\alpha, \alpha_0)^{12}C$ at 165°. Theoretical calculations identify candidate areas for resonances with experimental data points provide confidences in the evaluation.

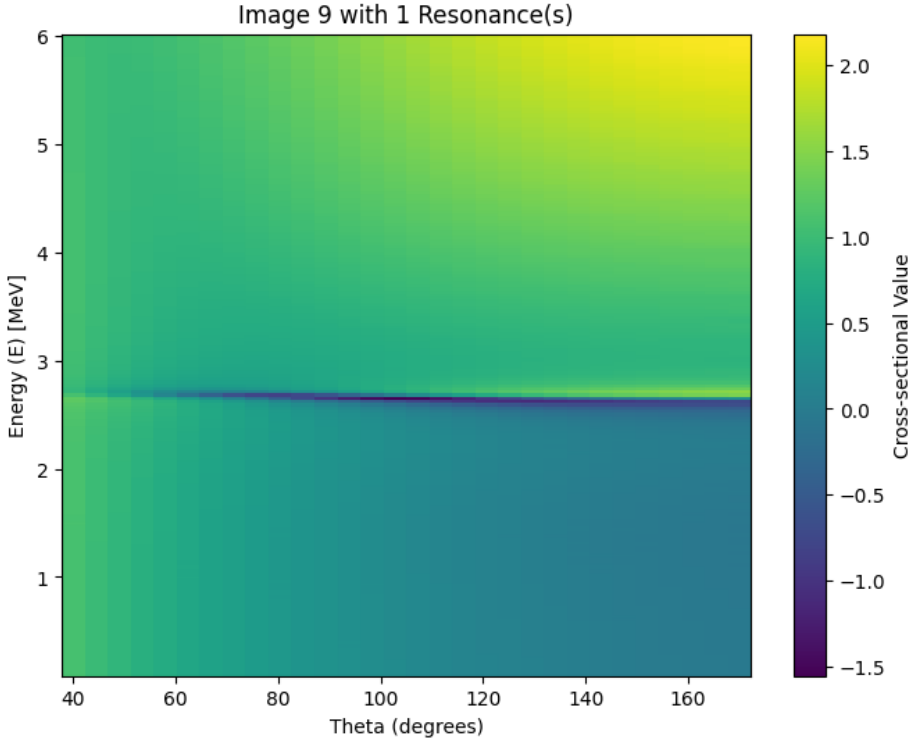


Figure 2: Energy-theta cross-section with 1 resonances at 2.67 MeV that runs the length of the angular distribution. Note the slight shift in the energy level of the resonance across theta.

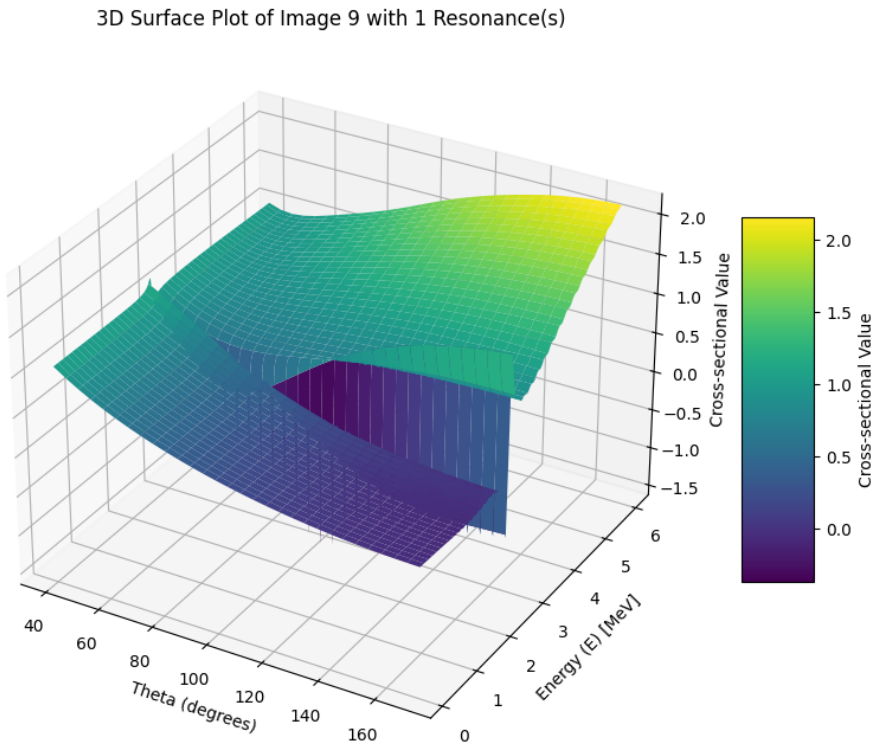


Figure 3: Energy-theta cross-section with 1 resonances. The amplitude across the manifold changes drastically even without the presence of resonances.

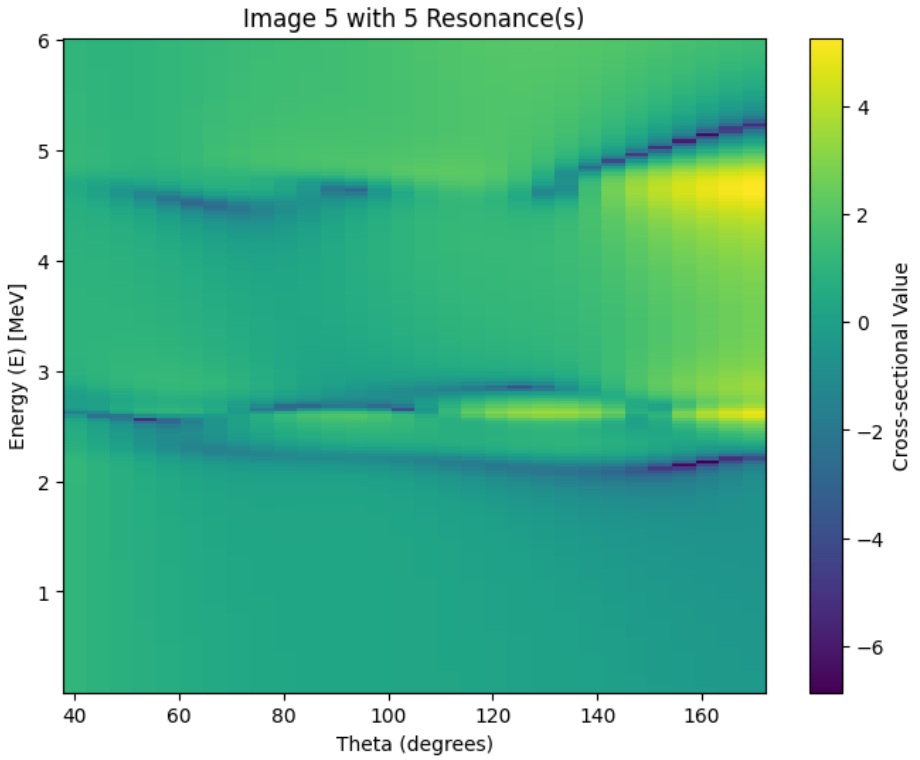


Figure 4: Energy-theta cross-section with 5 resonances. Neighbouring resonances interfere with each other creating constructive and deconstructive amplitudes.

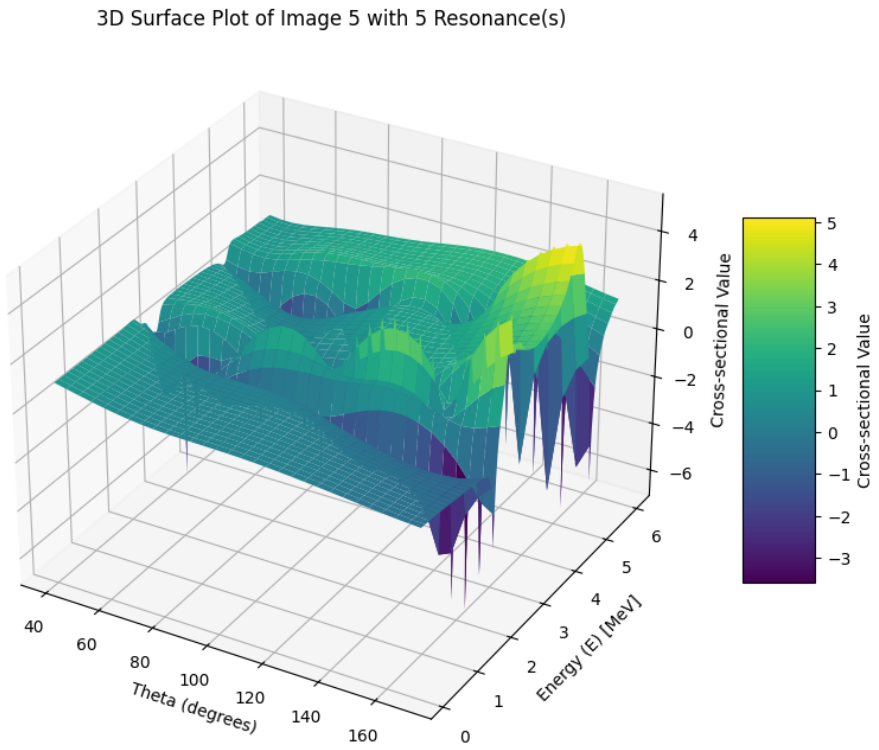


Figure 5: Energy-theta cross-section with 5 resonances. A morphed topology of convoluted amplitudes.

E_{res} (MeV)	Γ_{res}	l_{res}
2.8045	0.2688	1
2.2535	0.2621	0
2.6134	0.0784	4
4.7708	0.2574	2
4.6362	0.3175	3

Table 2: Resonance parameters of the image with 5 resonances.

This is the premise of the exploratory approach into machine learning techniques to support the evaluation of nuclear data. There is an extant rules-based system for the identification and characterisation of the resonance phenomena and its cross-sectional yet its formulation is hindered by interference. Yet clearly there is observable trends and patterns emerging from them. As such, this research project attempts the novel application of a machine learning model to predict resonance properties on a given energy-theta image.

4 Machine Learning Overview and Prototyping Model Design and Performance

Machine learning has emerged as a transformative tool in various fields of science and engineering, offering powerful methods for analysing complex datasets and uncovering underlying patterns. In the realm of nuclear physics, machine learning techniques provide innovative and novel approaches to modelling intricate nuclear phenomena from a variety of applications and offer researchers alternative approaches to experimental data interpretation. This chapter presents an overview of the fundamental principles and concepts of machine learning, laying the groundwork for understanding how these methods can be applied to aforementioned problem with resonance identification. The discussions draw upon established methodologies and are supported by foundational texts in the field [17, 18, 19, 20, 21, 22]. This chapter will include with the incorporation of these fundamentals into a simplified resonance feature extraction model that has been used as a proof-of-concept and test ground for prototyping pre-processing techniques and evaluating model sensitivity.

4.1 Machine Learning Concepts

This section covers the introductory concepts and principles of Machine Learning and what its applications seek to achieve. It should address:

- The difference between Artificial Intelligence, Machine Learning, and Neural Networks.
- Examples of Machine Learning being applied within the bounds of nuclear science research.
- An overview of the diversity of Machine Learning models employed to handle various tasks.
- Introduction to convolution neural network and their main application purpose
- Convolutional neural networks candidacy as an effective solution for this nuclear science problem.

4.1.1 Artificial Intelligence, Machine Learning, and Neural Networks

Artificial Intelligence (AI) is a broad discipline that involves the development of computer systems capable of performing tasks that typically require human intelligence, such as reasoning, learning, perception, and language understanding [23]. Machine Learning (ML) is a subset of AI focused on the development of algorithms that enable computers to learn from and make predictions or decisions based on data, improving their performance over time without being explicitly programmed for each task [24]. Within ML, Neural Networks (NNs) are computational models inspired by the structure and function of biological neural networks in the human brain. They consist of interconnected layers of nodes (neurons) that process input data by adjusting the weights of connections through a learning process [18]. Notably, the mechanisms that enable these networks are largely non-interpretable by human perception and are often considered a black-box in this regard.

4.1.2 Applications of Machine Learning in Nuclear Science

Machine learning techniques have been increasingly adopted in nuclear science to tackle complex problems involving large datasets, nonlinear relationships, and high-dimensional parameter spaces that are challenging for traditional analytical methods. The versatility of machine learning algorithms enables their application across various domains within nuclear physics, from fundamental research to practical engineering solutions.

One significant application is in the prediction of nuclear properties. Machine learning models, such as neural networks and Gaussian mixture models, have been employed to predict nuclear masses, binding energies, and decay half-lives with high accuracy [25, 26]. These models learn patterns from existing experimental data and theoretical calculations for the interpolation and extrapolation of nuclear properties in isotopes that are difficult to study experimentally.

In nuclear reactor physics, machine learning aids in monitoring and control systems. Algorithms have been developed to classify reactor transients, detect anomalies, and predict the remaining useful life of reactor components for the benefit of operational safety and efficiency [27, 28]. Reinforcement learning has also been explored for optimising reactor control strategies under varying operational conditions.

Artificial Neural Networks are being used in the neutron γ discrimination space. Traditional discrimination methods, such as pulse shape discrimination (PSD), rely on analysing the shape of the detector signals to differentiate between neutrons and gamma rays. However, these methods can face limitations in complex experimental environments or when dealing with overlapping signals. ANN methods have outperformed PSD classification of scintillation pulses and offer particle distinction in high flux radiation environments [29, 30].

Another example is the novel framework for the prediction of nuclear reaction cross sections using machine learning models [31]. Similar to this project, the experimental databases for nuclear reactions is sparse and subject to discrepancy and error. This model offers integration with both simulated data from ENDF/B-VIII.0 dataset and experimental data from EXFOR library.

There are many more examples and with the rapid uptick in ML partially attributed to the recent advances in popularity and accessibility. Various colloquiums have also been established to provide a broad meta analysis of the many major applications, such as [32, 33].

4.1.3 Neural Network Model Variations

Section 4.1.2 explores a select few applications in nuclear science where machine learning models have been employed to capture, detect, or detect underlying patterns and relationships within dataset. They pertain to a suite of research fields, from astrophysics to reaction cross-sections. Different applications require different model frameworks and

there is an ever growing diverse array of architectures designed to handle varying tasks. This section discusses some of the common model types with distinguishing features that make them suitable in novel frameworks.

Bayesian neural networks (BNNs) build on traditional feedforward neural networks by incorporating Bayesian inference, treating network weights as random variables with specified prior probability distributions instead of fixed parameters. This probabilistic framework allows BNNs to model prediction uncertainty, yielding full predictive distributions rather than simple point estimates. Key features include their ability to quantify epistemic uncertainty (uncertainty in model parameters) and aleatoric uncertainty (data noise), making them suitable for applications where prediction confidence is essential. Utama et al. [25] developed a BNN to model complex relationships between nucleon numbers and nuclear masses, capturing nonlinear patterns and providing credible intervals without the need for physical insights from the liquid drop model.

Convolutional neural networks (CNNs) process data with grid-like topology, such as images or sequential data. They include convolutional layers applying filters to local regions of the input, pooling layers reducing dimensionality, and fully connected layers for classification or regression. CNNs exploit spatial hierarchies by detecting local patterns through connectivity and weight sharing, and exhibit translation invariance. Their efficient parameter usage mitigates overfitting. In nuclear science, CNNs have been applied to gamma-ray spectroscopy for isotope identification [34] capturing features like peaks and edges in spectral data to distinguish between different radioactive isotopes even amid noise.

Recurrent neural networks (RNNs) and long short-term memory (LSTM) networks are designed for sequential data by incorporating loops that allow information persistence across time steps. LSTMs, a type of RNN, use gating mechanisms to control information flow, addressing the vanishing gradient problem. Their key features include the ability to model temporal dynamics, capture long-term dependencies through memory cells, and process ordered sequences. In nuclear reactor monitoring, LSTMs have been used to predict reactor core behaviour and detect anomalies over time [35] making them ideal for analysing sensor data streams and forecasting future system states.

Support vector machines (SVMs) are supervised learning models that classify data by identifying the optimal hyperplane separating classes in high-dimensional space. They handle nonlinearly separable data through kernel functions, aiming to maximise class margins, which enhances generalisation and reduces overfitting—particularly useful in high-dimensional settings. In nuclear science, SVMs detect anomalies in reactor operations [36], classifying operational data to identify deviations from normal behaviour, a critical factor for safety and maintenance.

Generative adversarial networks (GANs) include a generator that creates synthetic data and a discriminator that evaluates its authenticity. These networks train simultaneously within a minimax framework, where adversarial training enables high-quality data generation. GANs' versatility supports realistic data generation across domains. In nuclear science, GANs augment particle physics datasets, reducing the need for costly simulations [37]. This synthetic data aids in training other machine learning models by introducing more diverse examples.

4.1.4 Decision to use Convolutional Neural Networks

The selection of CNNs for this project is justified by their inherent architectural advantages in processing and analysing image-like data structures, which are directly applicable to scattering amplitude images in nuclear physics as discussed in Section 3.3. The scattering amplitude data, representing the variation of scattering intensity with respect to energy levels and scattering angles, can be effectively transformed into two-dimensional images making the input data suitable, while retaining ground truth targets developed from the artificially placed resonance features.

CNNs offer several key advantages for analysing scattering amplitude images in this project. They excel at local feature extraction through convolutional layers with learnable filters that scan over localised regions of the input image to detect features such as peaks, edges, and other distinctive patterns indicative of resonance phenomena. The hierarchical feature learning inherent in CNNs allows the network to combine these simple features into complex representations relevant to nuclear interactions while fully connected layers can identify and characterise resonances.

CNNs possess translation invariance due to convolution and pooling operations which protects feature extraction from shifts or translations in the input data. Additionally, CNNs eliminate the need for extensive manual feature engineering by automatically learning hierarchical features directly from raw data and can detect subtle and potentially overlooked patterns. Their ability to handle high-dimensional data effectively—reducing dimensionality through convolution and pooling while preserving information offers computational efficiency and allows for processing large datasets without compromising performance. These attributes make CNNs particularly well-suited for this approach and posit the external validation of this experiment.

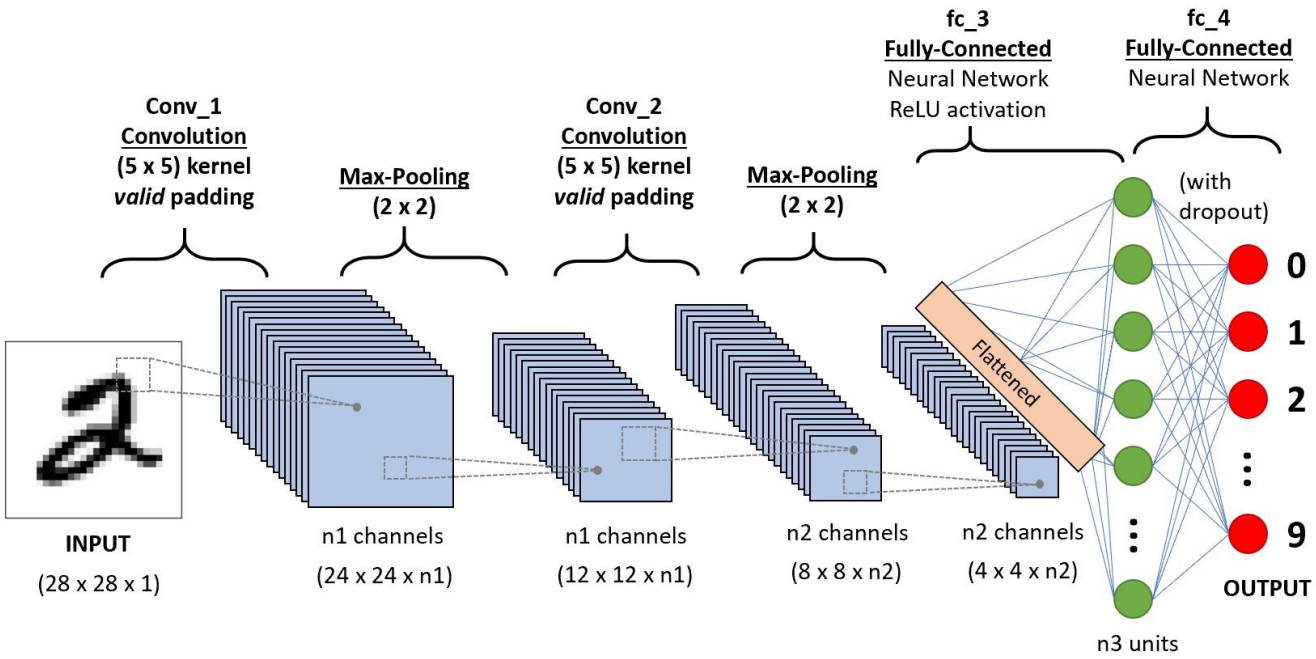


Figure 6: LeNet-5 CNN model is one of the earliest convolutional network structures developed by LeCun et al. in 1998 [38]. It's design has widespread adoption to modern CNN applications.

4.2 Convolutional Neural Networks

This section aims to describe how a CNN works and the details of a typical network architecture. It will do this by introducing the very well documented LeNet-5 CNN and break down the main mechanisms that make the model function. All of these mechanisms are relevant the resonance identification models proposed in this report, but there are cases where additional examples have been provided that are commonly used for the benefit of the reader.

- Standard network architecture construction and the exemplar LeNet-5 case.
- Introduction to convolutional and pooling layers.
- Visualising convolutional filters and model filter initialisation.
- Application of non-linearity through activation functions.
- Fully connected layers and output predictions.
- Loss functions, their variations, and multi-task weightings.
- Model training through back-propagation and gradient descent.

4.2.1 Network Architecture

Introducing CNN's often begins with the network architecture of LeNet-5, a historic model used to successfully depict images of hand written numbers as seen in Figure 6. The input passes through a sequence of convolutional layers that apply filters to extract local features, followed by pooling layers that downsample the feature maps, reducing dimensionality and computational complexity. After the feature extraction stages, the output is flattened and passed through fully connected layers, leading to a final softmax layer for classification.

Applying novel applications of CNN will begin with template architecture derived from this model. This report will break down each other steps in further detail in the following sections before exploring the specific parameters and additional steps in the development of the single resonance model in Section 4.4 and multiple resonance model in Section 5.2.

4.2.2 Convolutional Layers and Pooling Layers

Convolutional layers are the core building blocks of a CNN. They apply a set of learnable filters (interchangeably referred to as kernels) to the input data to extract high-level features. Each filter slides over the input feature maps, computing the dot product between the entries of the filter and the input at any position. An output feature map is then created per filter from the operation which can be fed forward into the model.

Mathematically, the convolution operation for a single output feature map can be expressed as:

$$y_{i,j}^{(k)} = \sum_{c=1}^{C} \sum_{m=1}^{M} \sum_{n=1}^{N} x_{i+m-1,j+n-1}^{(c)} \cdot w_{m,n}^{(c,k)}$$

where:

- $y_{i,j}^{(k)}$ is the output at position (i, j) in the k-th feature map.
- $x_{i,j}^{(c)}$ is the input at position (i, j) in the c-th channel (A channel is created for every kernel).
- $w_{m,n}^{(c,k)}$ is the weight at position (m, n) connecting the c-th input channel to the k-th output feature map.
- C is the number of input channels. In our case our input channel is one, but an photographic image may have multiple channels to represent the RGB values of a pixel. Each convolutional filter will create a channel downstream of the layer.
- $M \times N$ is the kernel size.

Notably, the kernel acts with the Hadamard product (or element-wise product) [39] on the window of the input with the same kernel size. The mapped value is then the summation of the resultant product.

The importance of the convolution operation is to capture local spatial patterns by considering small regions of the input and enabling the network to learn features such as edges, textures, and shapes.

The output dimensions of a convolutional layer are calculated using:

$$\text{Output Height} = \left\lfloor \frac{\text{Input Height} + 2P_{\text{height}} - M}{S_{\text{height}}} \right\rfloor + 1$$

$$\text{Output Width} = \left\lfloor \frac{\text{Input Width} + 2P_{\text{width}} - N}{S_{\text{width}}} \right\rfloor + 1$$

where:

- P_{height} and P_{width} are the padding sizes.
- S_{height} and S_{width} are the stride lengths.

The convolution operation is far easier to intuit with a visualisation and this can be demonstrated through image processing filter. Consider the example from [40] where we begin with an original image in Figure 7 which can attribute each of the 100×100 pixels to colour that can be represented as a tensor (assume a single channel for simplicity's sake). A Gaussian blur is an image processing filter that will blur an image to reduce the noise and sharpness of edges and can be represented by the matrix in Equation 3. For each pixel, the Hadamard product of the Gaussian blur filter and the same 3×3 convolutional window that is centred on the target pixel is computed, with the resultant being the normalised sum. This value is then stored as a pixel value in the new feature space which retains the same dimensionality, assuming the inclusion of a 1 pixel padding in this scenario. The resulting feature space can be observed in Figure 8.

$$\text{Gaussian blur } 3 \times 3 = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{3}$$

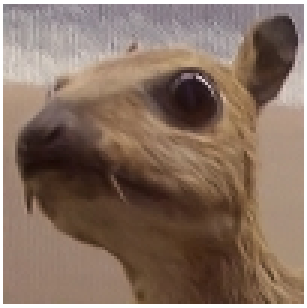


Figure 7: Original image [40]

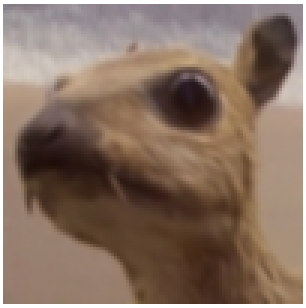


Figure 8: Image after Gaussian blur [40]

However in neural networks, the convolutional filters are far less interpretable that image filters, leading to question how they are initialised for neural networks and how they can be visualised.

4.2.3 Filter Initialisation

Any convolutional or fully connected layer in a CNN will require the model to initialise a filter for every desired output channel. The choice of how many filters to use in each convolutional layer lies in the expertise of model developers, but fortunately that ML modules such as PyTorch and Tensor Flow have in-built methods initialisation methods that aim to minimise the training process and reduce the vanishing or exploding gradient problem discussed in Section 4.2.10.

The default filter initialisation is done through He Uniform (also referred to as Kaiming Uniform) [41] where for a layer of n_l filter features, each kernel weight W is drawn from the uniform distribution U as shown in Equation 4.

$$W \sim \mathcal{U}\left(-\sqrt{\frac{1}{n_l}}, \sqrt{\frac{1}{n_l}}\right) \tag{4}$$

So for a 3×3 kernel giving 9 features, the bounds of the filter weights would then be $(-\frac{1}{3}, \frac{1}{3})$. As the filter weights are learnable in the neural network training process, these weights will adjust over time, but retain the uniform distribution. The convolution layer weights can be visualised by colour-mapping the weights within their distribution bounds. Figure 9 is an example of the initialised weights in the first convolution layer for the single resonance identification model. Notably, they may seem far less interpretable than the illustrations provided by the Gaussian blur example in Figure 7 and 8, but these weights and the feature maps they create are interpretable by machine learning.

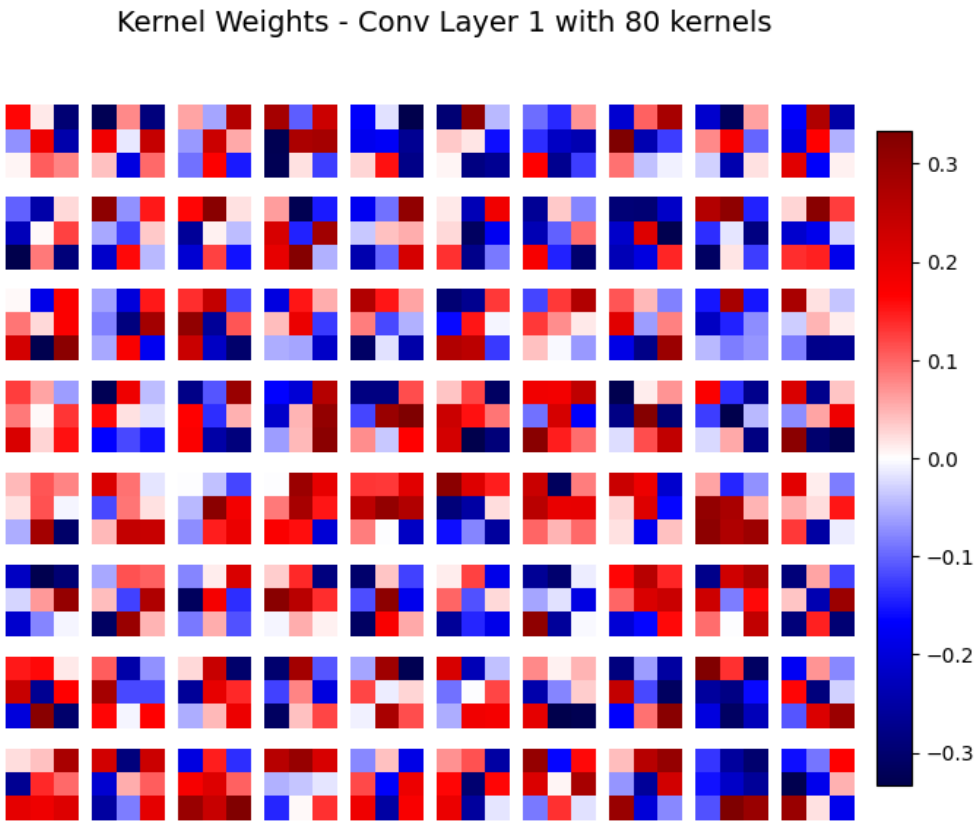


Figure 9: Convolution Filter Weights

4.2.4 Activation Functions

Activation functions introduce non-linearity into the network, enabling it to learn complex patterns. This is key to the ML operation, as without activation function the entire model can be condensed into a simple transformation matrix that is simply a linear map. Activation provides a mechanism for discrimination in the model. They are applied element-wise after convolutional and pooling operations. Some common activation for a given input x functions are:

- Rectified Linear Unit (ReLU):

$$f(x) = \max(0, x)$$

  ReLU accelerates convergence by mitigating the vanishing gradient problem (Section 4.2.10) seen by tanh and sigmoid functions that squash their outputs to a narrow range.

- Sigmoid Function:

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

  Maps input values to the range (0, 1), suitable for binary classification outputs or confidence measures.

- Softplus:

$$f(x) = ln(1 + e^{x})$$

  A smooth approximation to the ReLU function and ensures the output is always positive. Useful for predicting small values close to 0.

- Softmax:

$$\sigma(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$$

  Applied to layers to provide confidence values of the normalised exponential function.

- Hyperbolic Tangent (tanh):

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

  Maps input values to the range (-1, 1), centreing the data.

- Leaky ReLU:

$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha x, & \text{if } x \leq 0 \end{cases}$$

  with α typically set to 0.01, allowing a small gradient when $x \leq 0$.

This project will use ReLU, sigmoid, and softplus, and softmax. Tanh was tested during the prototyping of the model but showed less promise than ReLU for the application.

4.2.5 Pooling Layers

Convolutional layers add significant computational complexity and dimensionality to the model which has significant drawbacks for complex model accessibility as well as excessive dimensionality reducing generalisation [42]. To counter this, each convolution layer is then wrapped by a pooling layer. Pooling layers reduce the spatial dimensions of the feature maps through a pooling operation, but retain the channel dimension.

The most common pooling operation is max pooling, defined as:

$$y_{i,j}^{(k)} = \max_{\substack{0 \leq m < F \\ 0 \leq n < F}} x_{S \cdot i + m,\, S \cdot j + n}^{(k)}$$

where:

- F is the pooling window size, similar to the convolutional layer kernel size.
- S is the stride, determining the step size the window moves after each pool.
- $y_{i,j}^{(k)}$ is the pooled output.
- $x^{(k)}$ is the input feature map.

Pooling layers introduce a form of translational invariance, making the network less susceptible to small shifts and distortions in the input. In Figure 6, a $F = 2 \times 2, S = 2$ pooling operation is applied to convolutional layer 1 which reduced the spatial dimensions from 24×24 to 12×12. Each value in the output layer would represent the maximum value in the 2×2 window of the input map.

4.2.6 Fully Connected and Output Layers

After convolutional and pooling layers, the high-level feature representations are flattened into a one-dimensional vector and fed into fully connected layers.

Each neuron in a fully connected layer is connected to every neuron in the previous layer. The operation is defined as:

$$\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

where:

- $\mathbf{y}$ is the output vector.
- $\mathbf{W}$ is the weight matrix.
- $\mathbf{x}$ is the input vector.
- $\mathbf{b}$ is the bias vector.

The final layer produces the network's predictions. These predictions can have a choice of activation function depending on the application as discussed in Section 4.2.4.

4.2.7 Loss Function Calculation

The loss function quantifies the discrepancy between the predicted outputs and the true labels, guiding the optimisation process. In a multi-task learning setting, the model is trained to perform multiple related tasks simultaneously by sharing representations across these tasks. The overall loss function in a multi-task CNN is typically formulated as the summation of individual loss functions corresponding to each task. This aggregated loss function allows the network to learn shared features that are beneficial for all tasks while also capturing task-specific nuances.

Mathematically, the total loss L_{total} is expressed as:

$$L_{\text{total}} = \sum_{t=1}^{T} \lambda_t L_t$$

where:

- T is the total number of tasks.
- L_t is the loss function associated with task t.
- λ_t is a weighting factor that balances the contribution of each task to the total loss.

Each individual loss L_t is computed based on the nature of the specific task.
Individual loss functions include:

- Mean Squared Error (MSE) for Regression: MSE is used for predicting results where targets can take on real real value $\hat{y}_i \in \mathbb{R}$ such as the energy of a resonance.

  $$L_{\text{MSE}} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

  where:

  - N is the number of samples.
  - y_i is the true value.
  - $\hat{y}_i$ is the predicted value.

- Cross-Entropy Loss for Classification:

  Binary Classification is used for predicting results where the targets are limited to binary options $\hat{y}_i \in \{0, 1\}$. A prediction is typically activated by a sigmoid function that converts the regression outputs to a value between that range. An example of this is predicting the confidences of an resonance being present within a bounds.

  $$L_{\text{BCE}} = -\frac{1}{N} \sum_{i=1}^{N} \left[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right]$$

  Multi-class Classification is similar to binary, but targets are limited to a defined set of integers representing classes $\hat{y}_i \in \{0, 1, 2, ..., N\}$

  $$L_{\text{CCE}} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{C} y_{i,c} \log(\hat{y}_{i,c})$$

  where $y_{i,c}$ is a binary indicator for class c.

4.2.8 Backpropagation and Gradient Descent

Backpropagation computes the gradient of the loss function with respect to each weight by applying the chain rule of calculus, propagating errors backward through the network. This is effectively how the model learns, as it seeks to minimise the error by adjusting the weights in the direction of the gradient descent. After the model has taken a batch of input images and fed them forward through the model to calculate the error and loss , the backwards loss function is called to update the weights of the model in the direction of the gradient descent.

For a weight $w_{ij}^{(l)}$ connecting neuron j in layer $l-1$ to neuron i in layer l:

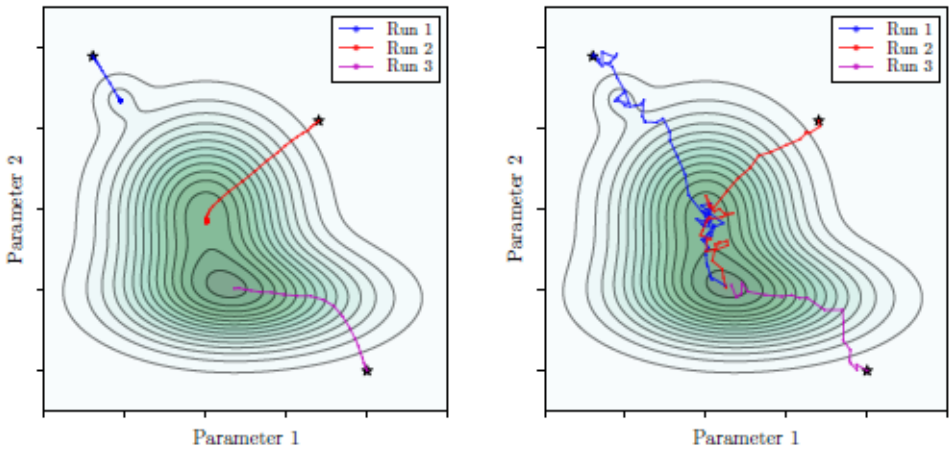


Figure 10: Limitations of backpropagation gradient descent [43]

$$\frac{\partial L}{\partial w_{ij}^{(l)}} = \delta_i^{(l)} a_j^{(l-1)}$$

where:

- L is the loss function.
- $\delta_i^{(l)}$ is the error term for neuron i in layer l.
- $a_j^{(l-1)}$ is the activation of neuron j in layer $l-1$.

The error term $\delta_i^{(l)}$ is calculated as:

$$\delta_i^{(l)} = f'(z_i^{(l)}) \sum_k \delta_k^{(l+1)} w_{ik}^{(l+1)}$$

where f' is the derivative of the activation function.
Weights are updated using the gradients to minimise the loss:

$$w_{ij}^{(l)} \leftarrow w_{ij}^{(l)} - \eta \frac{\partial L}{\partial w_{ij}^{(l)}}$$

where η is the learning rate.

This iterative process continues until convergence. However, convergence does not necessarily dictate the global minima of the gradient descent. There are a several considerations that need to be including the optimiser limitations and awareness of the vanishing and exploding gradient problems.

4.2.9 Optimisers and Learning Rate

The main limitation of gradient descent with backpropagation is its inability to guarantee finding the global minimum, as it can become trapped in local minima and affected by noisy data due to the finite input set.

To visualise this and the role of optimisers and their learning rates, assume two model parameters and a contour field representing the weights values, as seen in Figure 10. The global minimum of that contour field represents the optimal solution, or the lowest value loss function. STarting from three points, the optimiser seeks the largest negative partial derivative of $\frac{\partial L}{\partial w_{ij}}$, with magnitude proportional to the learning rate. n the left image, a smooth descent shows two paths getting trapped in local minima, while the right image shows a more chaotic descent where the higher learning rate helps all paths reach the global minimum despite validation loss increasing at times. The takeaway is to set a sufficiently large learning rate (potentially adjusted by a scheduler) and use a large validation set to avoid trapping parameters in local minima.

4.2.10 Vanishing and Exploding Gradient Problem

The vanishing and exploding gradient problems are challenges encountered during the training of deep neural networks, particularly those with many layers. Remembering that the each neuron receives an updated proportional to the loss function with respect to the current weight[44], the vanishing gradient problem arises when gradients become exceedingly small as they are backpropagated through the network layers. This diminishes the updates to the weights in the earlier layers, impeding the network's ability to learn long-range dependencies and causing slow convergence or stagnation during training. Conversely, the exploding gradient problem occurs when gradients grow exponentially during backpropagation, leading to extremely large updates to the network weights. This can cause numerical instability, resulting in oscillating weights or divergence, and prevents the network from converging to an optimal solution.

To address these issues, several techniques have been developed and integrated into neural network training methodologies. The use of activation functions like ReLU mitigates the vanishing gradient problem by providing a constant gradient for positive input values. By contrast the sigmoid and tanh activation function that compress the scores to a small set of values is far more susceptible to these problems. Weight initialisation strategies such as Xavier initialisation and He initialisation are designed to keep the scale of the input and output variances the same across layers, which helps in preventing both vanishing and exploding gradients. Gradient clipping is employed to handle exploding gradients by capping the gradients to a predefined maximum norm and remain within a stable range.

4.3 Model Performance and Overfitting

This section aims to describe how a ML model performance is measured beyond the loss function and optimised. It introduces some of the fallacies tied to model sizing. It should address:

- Measuring model performance beyond loss minimisation.
- Overfitting and generalisation of models.
- Improving model generalisation performance through regularisation and normalisation techniques.
- Improving model through input pre-processing techniques.
- Evaluating model performance through sensitivity analysis and hyperparameter adjustment.

4.3.1 Measuring Model Performance

Evaluating a model's performance involves assessing its predictive accuracy and generalisation ability for each specific feature. Appropriate metrics provide quantitative measures to compare models and guide improvements. It is important to track the performance of these metrics per epoch for both the training and validation sets. There is overlap with model performance evaluation and loss functions, but the distinction between them is that model performance is concerned with providing a normalised or standardised set of results for each feature, while the loss function is there to assist with back propagation and gradient descent. Typical performance for regression metrics include:

- Coefficient of Determination (R^2 Score):

$$R^2 = 1 - \frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{N} (y_i - \bar{y})^2}$$

  where:

  - y_i is the true value.
  - $\hat{y}_i$ is the predicted value.
  - $\bar{y} = \frac{1}{N} \sum_{i=1}^{N} y_i$ is the mean of the true values.

- Mean Absolute Error (MAE):

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|$$

- Mean Squared Error (MSE):

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

- Root Mean Squared Error (RMSE):

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

The R^2 score measures the proportion of variance in the dependent variable that is predictable from the independent variables. It ranges from $-\infty$ to 1, with higher values indicating better model performance. An R^2 score of 1 implies perfect prediction, while a score of 0 indicates that the model does no better than the mean of the data. Negative values suggest that the model performs worse than simply using the mean.

Classification tasks instead assess discrete outputs against discrete labels, often to determine the models predictions against binary outcomes. Metrics include:

- Accuracy:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{N}$$

- Precision, Recall, F1-Score:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- Confusion Matrix:

  A table that summarises the performance by showing true vs. predicted classifications.

4.3.2 Overfitting and empirical risk minimisation

Overfitting is a fundamental challenge in machine learning where a model learns the training data too well, including its noise and outliers, to the extent that it negatively impacts the model's performance on new, unseen data. This occurs when a model is excessively complex relative to the amount of training data, allowing it to capture spurious patterns that do not generalise beyond the training set. This also introduces the concepts of generalisation which is a key characteristic of the model to handle its novel applications.

Mathematically, overfitting can be characterised by a model that minimises the empirical risk on the training data but fails to minimise the expected risk on the overall data distribution. The empirical risk R_{emp} is defined as:

$$R_{\text{emp}} = \frac{1}{N_{\text{train}}}\sum_{i=1}^{N_{\text{train}}} L\left(f(x_i;\theta), y_i\right)$$

where:

- N_{train} is the number of training samples.
- x_i and y_i are the input and true output of the i-th training sample.
- $f(x_i;\theta)$ is the model's prediction with parameters θ.
- $L(\cdot)$ is the loss function.

An overfitted model exhibits low training loss but high loss on validation or test data, indicating poor generalisation.

4.3.3 Training and Validation Sets

An effective strategy to mitigate overfitting involves dividing the available dataset into distinct subsets: the training set and the validation set. This separation allows the model to be trained and evaluated on different data, providing a clear assessment of its ability to generalise to unseen inputs.

Overfitting can be detected by monitoring the training and validation losses over epochs. Typically, during the initial phase of training both R_{train} and R_{val} decrease, indicating that the model is learning generalised patterns. However, as training continues R_{train} may continue to decrease, approaching zero as the model fits the training data perfectly. R_{val} may start to increase after a certain point, signaling that the model is beginning to overfit the training data and is not generalising well to unseen data.

This divergence between training and validation losses is a hallmark of overfitting. Validation sets can identify the epoch at which R_{val} is minimised and apply early stopping to halt training at this optimal point.

Consider the following example:

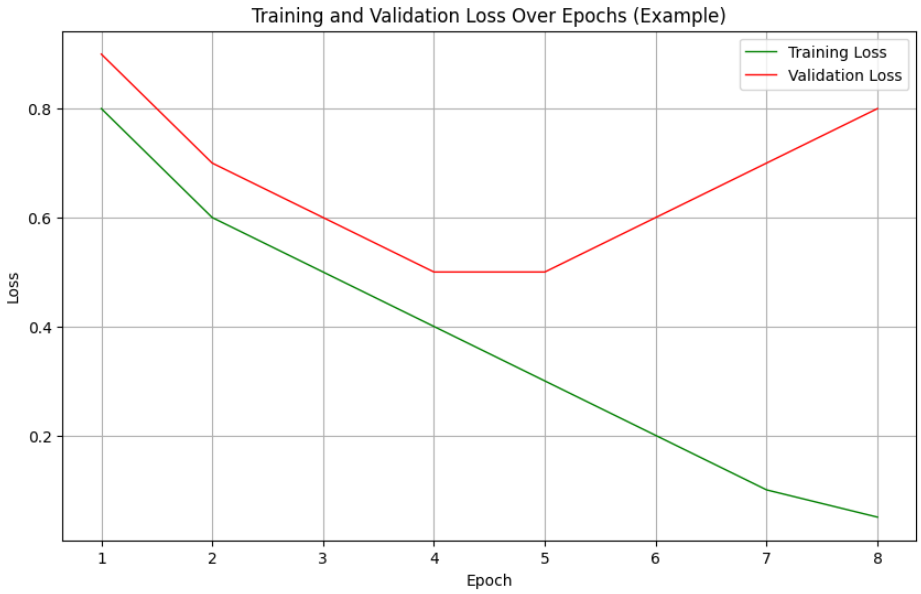


Figure 11: Example of overfitting from training and validation loss divergence

In this example, the validation loss R_{val} reaches its minimum at epoch 4 and starts increasing afterward, while the training loss R_{train} continues to decrease. This indicates that the model begins to overfit after epoch 4.

4.3.4 Dataset Segmentation

Given a dataset with known labels (also referred to as targets), there are various practices for data segmentation into training and validation sets. Notably, the training set data will see the loss function backpropagated through the model's weights while the validation set will not to measure the models ability to generalise. As such, data segmentation techniques are important to ensure that the model sees the appropriate amount of training data to learn complex features while retaining a suitable partition for generalisation, depending on the application. For example, some novel applications may not require generalisation as the area of interest is wholly contained within the training data.

Some common methods for data segmentation are as follows:

- Random Sampling: Split the data randomly to ensure that both sets are representative of the overall data distribution.
- Stratification: If the data is imbalanced across classes or labels, stratified sampling ensures that both sets maintain the same class proportions.
- K-fold cross-validation: Dataset is divided into k equal parts, and the model is trained and validated k times, each time using a different part as the validation set and the remaining parts as the training set.

Both training and validation sets have ground truth labels attached to the inputs to allow for the loss calculations. Experimental tests sets, such as the application of the IBANDL datasets, would then only output predicted values, with human inferencing needed to interpret result performance.

4.3.5 Regularisation

Regularisation introduces additional constraints or penalties on the model parameters to prevent overfitting by discouraging complex models that are overly flexible to meet the training data. A new loss function is calculated based on the addition of the empirical loss L_{emp} and a penalisation term from either a L1 or L2 application.

- L1 Regularisation (Lasso): Adds a penalty proportional to the absolute value of the weights:

$$L_{\text{total}} = L_{\text{emp}} + \alpha \sum_j |\theta_j|$$

- L2 Regularisation (Ridge): Adds a penalty proportional to the squared value of the weights:

$$L_{\text{total}} = L_{\text{emp}} + \alpha \sum_j \theta_j^2$$

  where α is the regularisation strength hyperparameter.

In this project, L2 regularisation was applied to the models through a weight decay in the optimiser where α was set to 1e-6.

4.3.6 Batch Normalisation

Batch normalisation normalises the inputs to each layer so that they have zero mean and unit variance. This helps stabilise and accelerate the training process by reducing internal covariate shift—the change in the distribution of network activations due to changes in network parameters during training.

For a mini-batch $\mathcal{B}$, batch normalization is applied as:

$$\mu_{\mathcal{B}} = \frac{1}{m} \sum_{i=1}^{m} x_i$$

$$\sigma_{\mathcal{B}}^2 = \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2$$

$$\hat{x}_i = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$$

$$y_i = \gamma \hat{x}_i + \beta$$

where:

- x_i is the input to the layer.
- $\mu_{\mathcal{B}}$ and $\sigma_{\mathcal{B}}^2$ are the mean and variance of the mini-batch.
- ϵ is a small constant for numerical stability.
- γ and β are learnable parameters that allow the model to restore the original distribution if necessary.

4.3.7 Dropout

Dropout is another regularisation technique that reduces overfitting by preventing complex co-adaptations on training data. During each training iteration, a random subset of neurons is temporarily "dropped out," meaning their activation is set to zero. This forces the network to learn redundant representations, as no single neuron can rely on the presence of other specific neurons.

Mathematically, for each neuron in the network, a Bernoulli random variable r_i is sampled:

$$r_i \sim \text{Bernoulli}(p)$$

where:

- p is the probability of keeping a neuron active, usually set between 0.8 and 0.5 during training.

The output of the neuron y_i is then scaled:

$$\tilde{y}_i = r_i y_i$$

In the model, this dropout function takes the input of (1 - p) to represent the fraction of neurons to drop in that layer. Dropout techniques can be applied to one or many of the fully connected layers. Recent developments have also looked at convolution layer dropout methods. During inference, no neurons are dropped, and the weights are scaled appropriately to account for the dropout during training.

4.3.8 Data Augmentation

Data augmentation artificially increases the diversity of the training dataset by applying transformations to the input data. These transformations should preserve the underlying label while introducing variability that helps the model generalise better.

Common augmentation techniques for image data include:

- Rotating images by small angles.
- Scaling by zooming in or out.
- Translation shifting images horizontally or vertically.
- Mirroring images horizontally or vertically.
- Introducing random noise to images.

4.3.9 Model Simplification and Sensitivity Analysis

Model simplification involves reducing the complexity of a neural network. Simplifying a model can be achieved by adjusting its architecture to better align with the complexity inherent in the data. In machine learning models, increasing model complexity does not always improve performance and can often have the opposite effect.

Key strategies for model simplification include:

- Reducing Network Depth: Decreasing the number of layers in the network limits the model's capacity to learn intricate patterns.
- Reducing Layer Width: Decreasing the number of neurons in each layer reduces the total number of parameters.
- Pruning: Removing weights or neurons that contribute little to the model's output can streamline the network without significantly impacting performance.
- Using Simpler Architectures: Choosing less complex model architectures or switching to models with fewer hyperparameters (e.g., from a deep CNN to a shallow network).

Simplifying the model helps in matching its capacity to the complexity of the problem and reducing the computational cost of training the model and evaluating data through it. To understand where a model can be simplified, a sensitivity analysis is often performed to see how the model performs under an iterative condition sweep of the input, hyperparameters, or parameters tuning.

In the context of neural networks, sensitivity analysis can identify which inputs (features) or parameters (weights) have the most significant impact on the model's predictions. Mathematically, sensitivity analysis involves computing the partial derivatives of the model output with respect to its input, hyperparameters, or parameters.

An example of this is seen in Han et al. (2015) [45], where a sensitivity analysis is applied to simplify neural networks by identifying and pruning weights deemed redundant. The authors first train the network fully, then evaluate each weight's importance based on the sensitivity $S_{ij} = |w_{ij}|$ of the loss function L. Weights below a set threshold are pruned, reducing network parameters and complexity without significantly affecting performance. Their results show that up to 90% of weights can be removed with minimal impact on accuracy, suggesting that many weights in large networks are unnecessary. While not implemented here, this approach could optimise model performance for larger datasets in future studies.

4.3.10 Data Pre-processing

Data pre-processing is essential in preparing datasets for machine learning models, with tailored techniques significantly enhancing model performance. Methods like normalisation and standardisation balance input variables and preventing dominant features from skewing the training process. Feature scaling (e.g., min-max normalisation or z-score standardisation) and encoding of categorical variables (e.g., one-hot or label encoding) make the data model-ready, while dimensionality reduction techniques, such as Principal Component Analysis, help limit redundancy and mitigate the curse of dimensionality [42]. Cleaning data through imputation methods can address missing, corrupt, or error-prone points.

The key takeaway is that pre-processing techniques must suit the specific model and dataset, as no single method guarantees performance improvements. This project initially focused on simplified models to prototype pre-processing methods effectively.

4.4 Simple Single Resonance Model

This section aims to detail the network architecture and performance of the single resonance image model. It aims to address:

- Rationale for simplified prototyping model.
- Dataset creation and understanding the targets.
- Pre-processing techniques used for this model.
- Model Architecture.
- Model Performance.

4.4.1 Prototyping Model

Developing complex machine learning models is a substantial feat as the various levers reserved for algorithm practitioners and developers are seldom predictable in their response. The neural network acts as a black box, so optimising model performance can be difficult to implement as the full weight of any action taken is not readily understood. As such, it is desirable to have a compact version of the model ready to test and prototype feature development. Prototyping models allow for rapid iteration of model hyperparameters and sensitivity analysis to model size. It offers the best chance of providing visualisation techniques of the data processing, which can lead to better operator diagnostics.

This project uses this strategy by creating a dataset of images that are limited to a single resonance. Notably, this takes away many of the compounding errors that proliferate through the model such as assignment cost and resonance overlap. This approach allowed direct observation of the model's response to feature implementation.

For the simplified model, the following techniques were implemented from an observed increase in model performance in validation sets:

- Sobel filter edge detection processing.

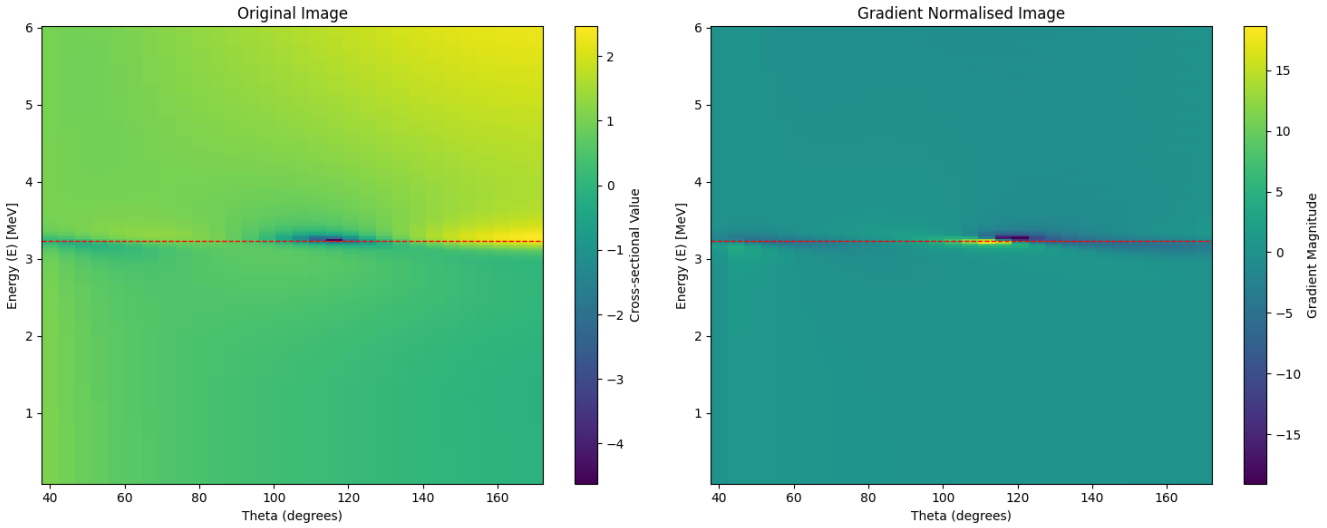


Figure 12: Image before and after Sobel filter & input normalisation.

- Input normalisation and standardisation.
- Cross entropy evaluation of angular momentum.
- Loss normalisation factoring.

4.4.2 Dataset Creation

The dataset was created using the Python twobody module [16]. The cross-sectional manifold was developed from Equation 1 with a hard sphere phase shift for δ_l. Then, for each resonance, the energy E, the width Γ, and angular momentum l properties were randomly determined. The Γ values were chose to (on average) increase with the resonance energy to mimic expected experimental characteristics. The resonant phase shift was then added to the hard sphere phase shift resulted in the corresponding cross-section. The angular dependence of the manifold was reduced by dividing through by the Rutherford cross-section (θ dependent) and $sin(\frac{\theta}{2})$. Finally, the resultant cross-section field was compressed through the natural log operator.

The module created datasets of 100, 1000, and 10,000 images for training rate and generalisation analysis. Only the 10,000-image dataset was used for evaluatio.n The dataset used an 80-20 training and validation random split respectively. Each model configuration was run multiple times to account for sensitivity with sampling bias. The sensitivity results for dataset size variations and sampling bias have not been included in this report.

4.4.3 Sobel Filter

As part of the pre-processing pipeline, the Sobel filter was applied to the scattering amplitude images to enhance edge detection and emphasize significant features. The Sobel filter computes the gradient of the image intensity, highlighting regions with abrupt changes in amplitude corresponding to resonance peaks and troughs. Mathematically, the Sobel operator uses convolutional kernels G_x and G_y to approximate the derivatives along the horizontal and vertical axes:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}, \quad G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}.$$

Typically the magnitude of the gradient $G = \sqrt{G_x^2 + G_y^2}$ would be computed as the new input channel. However, as this loses information regarding the resonance fissure characteristics (incline or decline), this project opted for an additive gradient $G = G_x + G_y$. This pre-processing step enhances the model's sensitivity to the resonance features while flattening the manifold.

4.4.4 Input Normalisation

Input normalisation was performed on the image to improve numerical stability during training. Each image was normalised by:

$$x_{\text{norm}} = \frac{x - \mu}{\sigma},$$

Input normalisation remains important even when using batch normalisation in the CNN, as batch normalisation is only applied after a layer in the network to address the outputs. Alongside this, input normalisation allows for faster training and convergence.

4.4.5 Loss Function: MSE vs Cross-Entropy

Evaluating model predictions needs to consider the subset of possible values the label can take, and apply the appropriate criterion for loss calculation and model evaluation as discussed in Section 4.3.1. In this scenario, the labels $E_{\text{target}}, \Gamma_{\text{target}} \in \mathbb{R}$ and as such, the appropriate loss function is derived from a regression evaluation, in this case Mean Squared Error (MSE). However, the angular momentum can only take on discrete positive values $l_{\text{target}} \in \mathbb{Z}^+$ which can benefit from categorical assessment through classification techniques. An ideal choice for classification tasks is the cross-entropy function. However, as the variable is discrete instead of ordinal, it is worth evaluating the variable under both a regression and classification schema. This will be included in the model architecture below and results below.

4.4.6 Loss Normalisation

Given the multi-task learning objective of predicting both continuous energy E_{target} and width Γ_{target} of sizeable magnitude differences, and discrete angular momentum l_{target} features, loss normalisation was implemented to balance the contributions of each task to the total loss function. For each feature, the normalisation factor is:

$$\lambda = \frac{1}{\sigma_y}$$

where σ_y is the mean of that feature within the loaded training set.

The total loss L_{total} was computed as a weighted sum of the individual losses:

$$L_{\text{total}} = \sum \lambda_i L_{\text{i}}$$

Adjusting these scaling factors allows the model to avoid domination of one loss component over others and optimise such that all tasks are learned equitably. This approach improves the convergence behaviour during training and leads to better generalisation performance across all predicted features.

4.4.7 Multi-task Learning

The previous Section 4.4.6 introduces the concept of multi-task learning, where the model is trying to simultaneous train and predict on multiple tasks (E, Γ, l). A natural intuition would suggest that separate models for each metric would lead to more optimised results as the loss function is no longer constrained to find the minimum summation of all tasks. However, as presented by the initial paper by Caruana [46] and overview of multi-task learning in neural networks by Ruder [47], shared representation can, but not always, lead to inductive transfer that improves the generalisation of the model. Learning tasks in parallel can allow the model to learn discriminatory features from one task designation that is shared and used by other tasks. It does this by using domain information contained in the training signals for related tasks as a bias without the need for supervision. The benefits of multi-task learning is enhanced attention focusing, implicit data augmentation through effectively increasing the sample noise, and improved generalisation through inductive bias.

In the context of this project, it is clear that resonance parameters demonstrate a linkage of relatedness by their influence on the cross-section manifold. As such, the single resonance model will employ this multi-task learning approach by keeping the shared fully connected network cascade down to the output layers.

4.4.8 Model Architecture

The model architecture, developed in PyTorch [48], involves a 1-80-160-320-640 2D convolutional layering structure with batch normalisation and ReLU activation as seen in Figure 13.

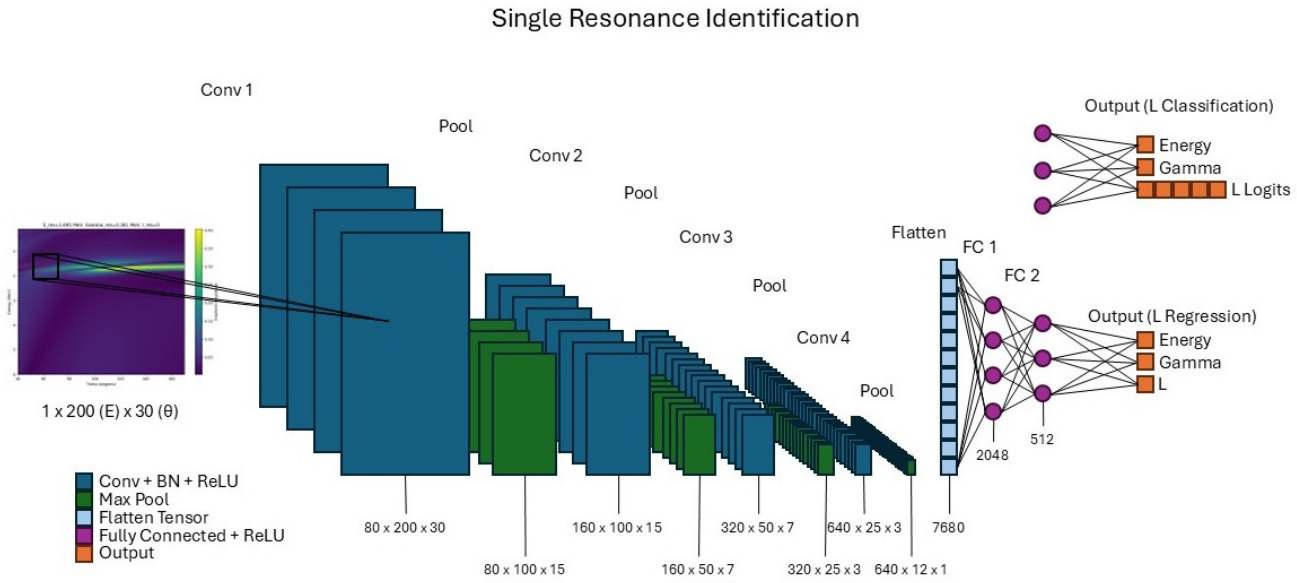


Figure 13: Single resonance model architecture.

Each convolutional layer was pooled with max pool function of size and stride 2. The outputs were then flattened to size 7680 and passed through a fully connected cascade of 7680-2048-512-Outputs. The fully connected layers used ReLU activation. The outputs reserved two indexes for E_{pred} and Γ_{pred}. An additional index was reserved for l_{pred} under regression analysis while 5 indexes were reserved for cross-entropy analysis, polling the argmax in its final prediction. There was a dropout layer between the 2048-512 fully connect layers with a 30% dropout.

The model implemented the Adam optimiser [49] with a learning rate of 1e-5 and weight decay of 1e-6. Alongside this, the ReduceLROnPlateau scheduler was implemented on the optimiser with a factor of 0.1 and patience of 10.

The model ran for 100 epochs with an 80-20 training and validation random split run in parallel. Loss was calculated from the summed loss function of the three outputs. Energy and width were evaluated through MSE for regression analysis. Angular momentum was calculated with MSE under the regression schema, and cross-entropy for the classification schema. The model collected the loss function for both training and validation sets per epoch. Additionally, it collected the validation set's R^2 score for regression tasks and the accuracy the classification task. A rounding function was applied to the l regression outputs to assess their accuracy for comparision between the model schema.

4.4.9 Model Results

The model demonstrated strong predictive capabilities for the resonance energy and width, with moderate but promising predictions for angular momentum under both the regression and classification schema. The models demonstrated convergence within 10 epochs of the loss function as illustrated in Figure 14 and 15. This indicates that the model efficiently learned the underlying patterns in the data. The subsequent divergence between the training and validation loss suggests overfitting, although the slight downward trend in the validation loss implies that extending the training duration might yield further improvements.

Figure 14: Single resonance model loss (L Regression). A generalisation gap appeared from overfitting. Implications to implant heavier regularisation in the advanced model.

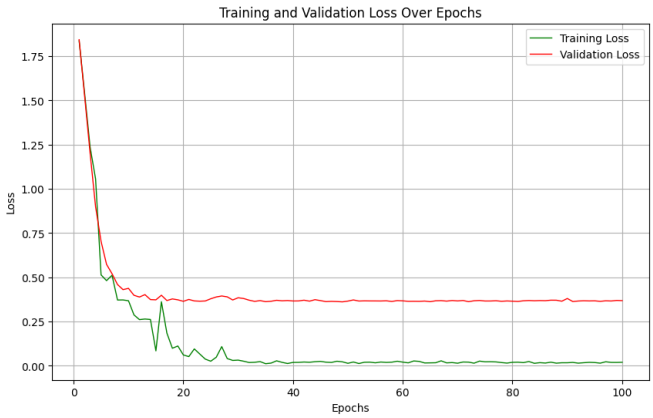


Figure 15: Single resonance model loss (L Classification). Similar characteristics to the regression schema with a generalisation gap.

The feature performance of the model was evaluated solely on the validation dataset that was excluded from model training as seen in Figures 16, 17, 18, 19, 20. Using the parameters at epoch 100, E achieved an R^2 score of 0.9764 indicating the model could explain over 97% of the variance of the labels could be explained by the models interpretation of the inputs. Resonance width Γ achieved a similar R^2 score of 0.9798, matching that of energy. However, as observed it had longer learning rate indicating the higher need for training data. These two features did not differ significantly across the model architecture for the two l schema. This indicates that the CNN architecture effectively captures the relationships between the input images and the resonance parameters E and Γ.

Angular momentum l performed worse than E and Γ and achieved an R^2 score of 0.7687 indicating capacity of the model to predict l but not at a suitability relevant to the application. When the regression outputs were rounded to the nearest integer to assess classification accuracy, the model achieved an accuracy of 67.15%. However, under the classification schema using cross-entropy, the model performed significantly better with an accuracy score of 91.00%. This substantial improvement suggests that treating l as a categorical variable and using classification techniques is more effective than regression for predicting discrete properties like angular momentum.

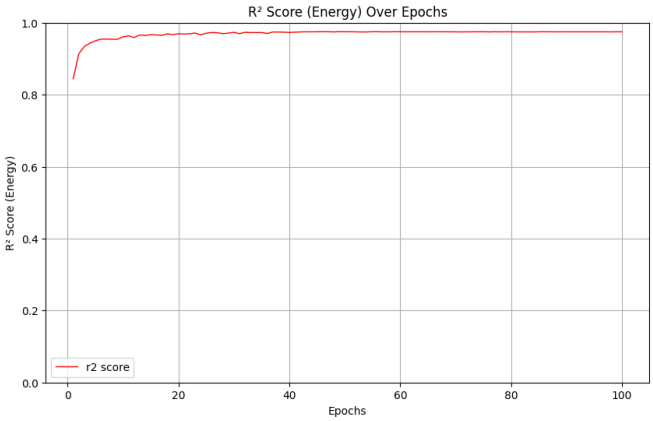


Figure 16: R^2 score for E predictions. The model could account for nearly all the variance in the input data with a quick training rate.

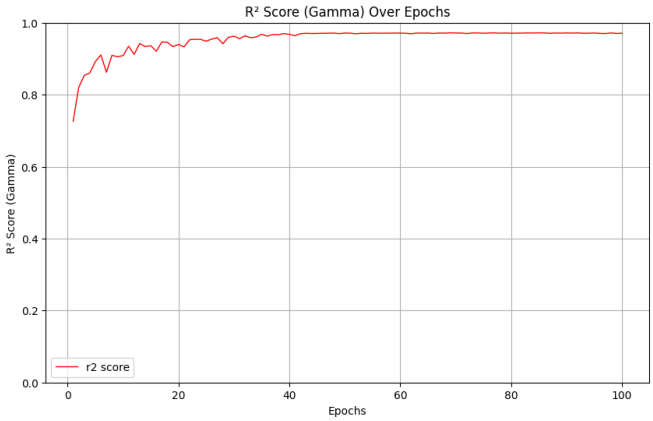


Figure 17: R^2 score for Γ predictions. Similar scores to energy but with a lower training rate indicating difficulty in appropriate feature extraction.

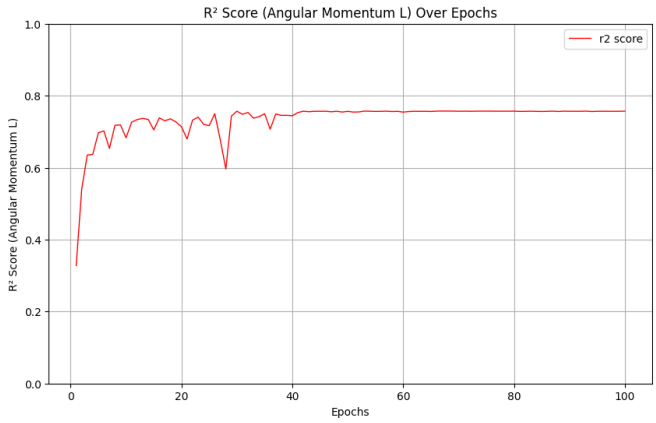


Figure 18: R^2 score for l predictions in regression schema. Moderate performance compared to the other parameters while the model still was able to capture some insight into the value.

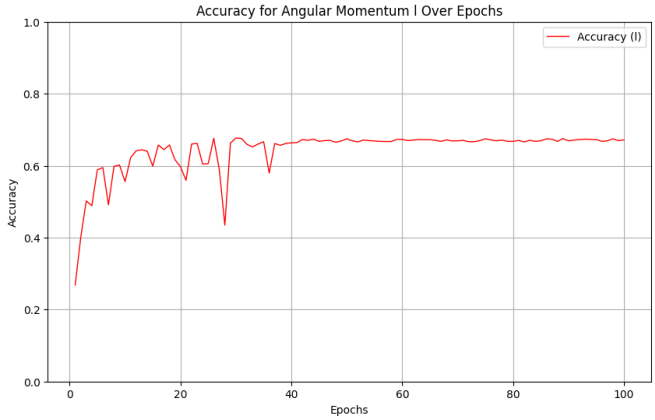


Figure 19: Accuracy score for l predictions from rounded regression values. Once again, a moderate performance that eludes insight into the models capacity but not to the standards needed in nuclear data analysis.

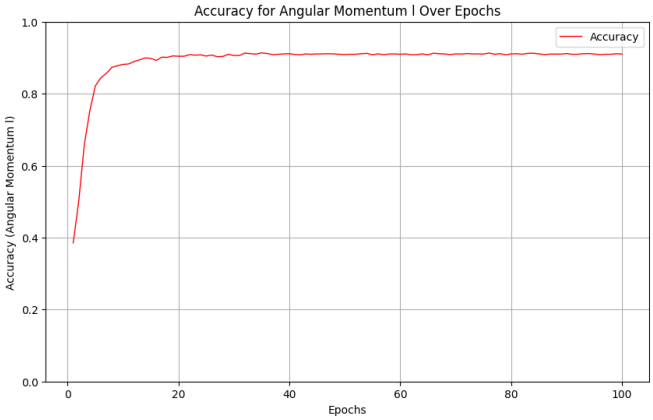


Figure 20: Accuracy score for l predictions in classification schema. Significant improvement over regression with a fast training rate.

These finding imply that the while the model can learn the dataset and predict continuous resonance parameters with minimal error, it struggles with discrete features with regression methods. The superior performance of the classification approach for l indicates that adapting the model to the nature of the target variable led to an increase in predictive accuracy, a key finding that will be implemented in multi-resonance detection model.

5 Multi-stage Resonance Identification Model Design and Performance

Finally, with the fundamentals of Machine Learning established and a simplified model demonstrating a proof of concept, an advanced model architecture can be constructed to handle the case of multiple resonances within the same task. Introducing multiple resonances into the same image and attempting to predict their properties adds a large order of magnitude in complexity and difficulty for model construction and evaluation. This area is one of the many frontiers of Machine Learning models and emerging developments are frequently being discovered. This chapter intends to detail the challenges arising from multiple resonance processing, the proposed CNN model architecture and performance, and some additional discussion regarding future direction.

5.1 Accounting for Multiple Resonances Imaging

The single resonance model demonstrated strong results at predicting the defined resonance parameters. However, in light of the reality of cross-sectional data as seen in Figure 1, the ideal model will need to be able to predict the

resonance parameters for a variable amount of resonances within a given energy-theta bound. Achieving this will allow the model to analyse real experimental data from the IBANDL library and discern between closely linked resonances that interfere with each other in the scattering amplitude. Doing so introduces a significant step in model complexity as the prediction model must be prefaced by a object detection candidacy model. This section aims to explore the challenges with variable detection systems combined with feature extraction. It addresses:

- Discriminating image object detection from feature prediction.
- Error propagation from detection-prediction inferencing.
- Various object detection frameworks applicable to this problem.
- Introducing masking and padding to retain tensor dimensionality.

5.1.1 Object Detection vs Feature Prediction

Detection refers to the identification and localisation of an object of interest within an input, whereas prediction involves estimating the specific attributes of properties of those objects. For resonance analysis, detection involves identifying the presence and positions of multiple resonances within the energy-theta scattering amplitude images. Thus, the first step of the model is to determine how many resonances exist within an image, and then predict the parameters of each resonance $(E_{\text{pred}}, \Gamma_{\text{pred}}, l_{\text{pred}})$. The key distinction is that detection must first localise the resonances before their specific properties can be predicted. In an image with multiple resonances, these tasks become more complex due to overlapping features, interference patterns, and error propagation.

5.1.2 Issues arising from Detection-Prediction Inferencing

Several challenges arise when extending the model to handle multiple resonances. Overlapping resonances may occur in both the energy and angular domains, leading to complex interference patterns that are difficult to disentangle. The number of resonances present in each image is not fixed, necessitating a model that can handle a variable number of objects. Associating predicted resonance parameters with the correct detected resonance becomes non-trivial when multiple resonances are present, posing an assignment problem. This matching is crucial because the loss function depends on the correspondence between predicted and actual resonances. If the model predicts resonances that do not align with the ground truth, it becomes challenging to compute an accurate loss, especially when the number of predicted resonances differs from the number of actual resonances. This necessitates a need for cost optimisation implementation, where the loss criterion between the predicted outputs and the target labels is assessed at the lowest cost.

Consider the example where the energy-theta image contains three ground truth resonances at energies $y = \{2.1\,\text{MeV}, 2.2\,\text{MeV}, 4.9\,\text{MeV}\}$. Suppose the model fails to disentangle the closely spaced resonances at 2.1 MeV and 2.2 MeV and instead predicts resonances at $x = \{2.1\,\text{MeV}, 5.1\,\text{MeV}, 3.2\,\text{MeV}\}$. The predicted resonances at 5.1 MeV and 3.2 MeV do not correspond well with the ground truth resonances, and will exacerbate the loss function. Likewise, if the model only compares predictions it is confident in $x = \{2.1\,\text{MeV}, 5.1\,\text{MeV}$ it will still run falsely bloat its loss function without cost implementation. There are various ways to handle this such as Deep Sets [50], or the Hungarian algorithm [51]. Neither approach was implemented in this project due to time constraints and will be included in further work.

Alongside this, error propagation exists even in a cost optimised model in both a single predictive model or detection-prediction inferencing model (discussed in Section 5.1.3). In either scenario, there is shared representation of the network architecture and the error in one task can adversely affect the performance of other tasks. This is due to the errors that stem gradient interference, where the gradients computed for each task during backpropagation are interfering with each other deconstructively. Unlike the multi-task learning findings in Section 4.4.7, in this scenario the properties of each resonance are unrelated to the properties of other resonances and as such do not benefit from inductive learning.

Finally, the model must scale to accommodate the additional complexity introduced by multiple resonances, potentially leading to increased computational complexity and resource requirements. Additionally, some resonances may be more prominent than others, leading to imbalanced representations in the dataset and potential bias in the model's learning process.

5.1.3 Object Detection Frameworks

Choosing the the appropriate detection framework for an application can be more art than science as model performance is generally evaluated empirically for a given dataset. It also is bespoke to the model developers goals and constraints, such as performance metric standards, computational budget, implementation expertise and a slew of other factors.

The framework chosen for this project is a multi-stage inferencing model where threes models are trained independently on the same training set and labels before being evaluated in a final inferencing function. Stage 1 handles the logic of assessing the number of objects (resonances) in an image and their corresponding energy. Stage 2 is trained on a cropped image offset from the true energy and attempts to correct the offset, while stage 3 uses the energy corrected crops to predict its resonance features width and angular momentum. Further detail in Section 5.2.

Several detection frameworks were explored for this project, with additional approaches considered for future development.

Region Proposal Networks (RPNs) identify regions likely to contain objects of interest. In resonance detection, an RPN scans scattering amplitude images to propose areas potentially containing resonances, with each proposed region processed to predict resonance parameters. This method accommodates variable resonance counts by generating content-based proposals but requires training data with optimised bounding boxes for effective evaluation. Common RPNs include Faster R-CNN [52], YOLO [53], and SSD [54].

Multi-instance learning enables prediction based on sets of instances within the data [55], allowing the model to identify and predict multiple resonances without specific region proposals.

Clustering techniques, such as Spectral Clustering [56] and KNN [57], can also be applied to feature maps post-convolution, forming potential resonance areas that can be clustered and used as inputs for further prediction.

5.1.4 Masking and Padding

Multiple object detection frameworks often require adapting the model to handle variable-length inputs and outputs. To achieve computational efficiency and integrate smoothly with PyTorch modules, padding and masking are commonly applied. Padding adds dummy data to inputs or outputs, ensuring a consistent length across sequences or data representations. The model takes the maximum expected resonances for an image; if fewer resonances are present, it pads with placeholder values (e.g., 0 or -1) that are ignored during evaluation through masking.

Masking identifies which parts of the input or output data are meaningful and which are padding. During training and inference, the mask enables the model to focus on relevant data, effectively ignoring padding that could introduce noise. This combined approach allows the model to process images with variable resonance counts accurately.

5.2 Multi-Stage Inferencing Model Results and Discussion

This section aims to detail the network architecture and performance of the multiple resonance CNN detection model. It addresses:

- Introducing the multi-stage inferencing model.
- Model architecture of each stage.
- Stage 1 model performance.
- Stage 2 model architecture.
- Stage 3 model performance.
- Inferencing results and discussion

5.2.1 Proposed Multi-stage Inferencing Model

Moving beyond the prototyping model, the decision to adopt a multi-stage inferencing model was driven by the complexity of detecting and predicting multiple resonances within energy-theta images. In scenarios with multiple resonances, overlapping features and interference patterns made it challenging for a single model to perform both detection and feature prediction. Initial attempts to approach the problem with a multi-task solution provided poor predictive capacity even with larger, computationally expensive models. The solution is to separate the detection

framework from the feature prediction framework into different models which allows each model to be trained separately with error propagation from unrelated multi-task learning.

The advantage of this multi-stage approach lies in the prioritisation of resonance detection accuracy which is the primary objective in this exploratory work. The detection stage focuses on detecting the number of resonances N and their predicted energy $E_{i,\text{pred}}$ without the added constraints of feature detection in predicted width Γ and angular momentum l, which as demonstrated in the prototyping model, was a far greater contributor to longer training rates and the loss function. It assesses the number of resonances by evaluating the confidences $\sigma \ \forall \ E_i$ where the *sigma* exceeds a defined threshold t.

A feature extraction model model is trained on segmented regions of the image, conceptually similar to a RPN model (Section 5.1.3). The regions proposed are extracted slices of the original image at each target resonance that capture the entire angular distribution. The images are centred at the target resonance energy with a slice width and evaluated on Γ, l predictions. This allowed the feature extraction model to focus on one particular resonance at a time and eliminate unrelated information in the training process.

However, noting that the detection model is going to contain residual error in its energy predictions, an offset was introduced in the feature extraction model that would augment the data by centreing the energy slice by an offset. Using

$$\sigma_{E,error} = \text{StdDev}\left(E_{\text{pred}} - E_{\text{true}}\right)$$

produced by the detection model to create an energy offset such that

$$\Delta E \sim \mathcal{N}\left(0, \sigma^2_{E,error}\right)$$

allowed it to mimic the uncertainty of the detection model's output.

Notably, the feature extraction model saw extremely quick performance degradation with increased $\sigma_{E,error}$ as seen in Table 3. Given the initial tests of the detection model produced a $\sigma_{E,error} = 0.154$, an interim model between the detection and feature extraction models was developed to improve the energy predictions by reducing $\sigma_{E,error}$. This model was fed the extracted slices of the original image at each of its resonances with an offset ΔE and trained to predict that offset. This allows the feature extraction model to run with a significantly smaller $\sigma_{E,error}$ and drastically increase its predictive capacity.

$\sigma_{E,error}$	Γ R^2 Score	l Accuracy Score
0.01	0.92	0.97
0.05	0.88	0.94
0.15	0.72	0.82

Table 3: Feature prediction performance degradation with increased $\sigma_{E,error}$. This creates imperative for energy correction mechanisms.

Finally, and inferencing function combines the three pre-trained models that can be used to interpret experimental test results where labels are undefined. An image will be passed into Stage 1 where the outputs propose $\{E_i, ..., E_n\}$ where $\sigma_i > t$. A sliced image centred at the predicted energy levels will then be fed to the Stage 2 model for energy correction. After energy correction, the image slice is re-extracted at the new energy level and evaluated for Γ, l features.

The multi-stage model architecture can be summarised in Figure 21 where each stage is pre-trained.

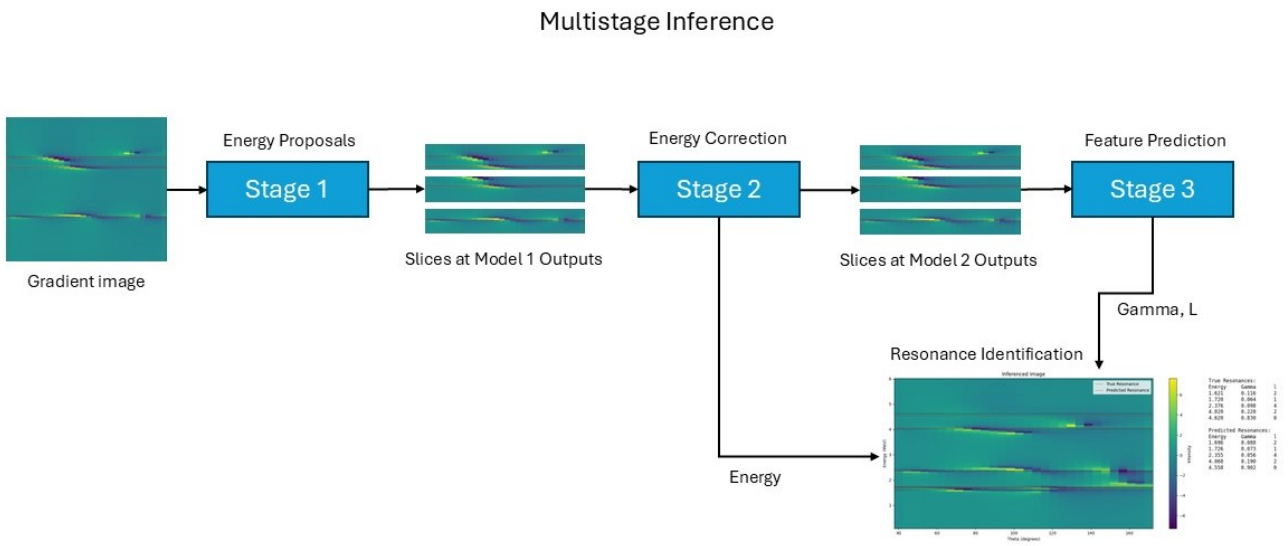


Figure 21: Multi-stage network architecture where a pre-processed image is passed through the inference pipeline to produce detection-prediction outputs.

5.2.2 Multi-stage Model Architecture Training Process

The specifications of each stage model is listed below. All models were derived from the same base architecture of Stage 1. If a hyperparameter is not listed in Stage 2 or 3, refer to Stage 1.

Stage 1: The Stage 1 model architecture as seen in Figure 22, involves a 1-80-160-320-640 2D convolutional layering structure with batch normalisation and ReLU activation. Each convolutional layer was pooled with a max pool function of size and stride 2. The outputs were then flattened to size 7680 and passed through a fully connected cascade of 7680-4096-1024-206-Outputs. ReLU activation was applied to the fully connected layers. The outputs produced a single output for number of resonances which was activated through the sigmoid function and multiplied by the max resonances. For energies and confidences, the output was an array of size max resonances. There was a dropout layer of 30% for both the 2048 and 1024 fully connected layers.

The model implemented the Adam optimiser with a learning rate of 1e-4 and weight decay of 1e-6. Alongside this, the ReduceLROnPlateau scheduler was implemented on the optimiser with a factor of 0.1 and patience of 5.

The model ran for 100 epochs, with an 80-20 training and validation split ran in parallel on 10000 generated images. Loss was calculated through the summed normalised loss function of the three output types. Number of resonances and energies were evaluated through MSE, while confidences were evaluated by binary cross entropy. Number of resonances and energies were then normalised by dividing through by the mean of the respective labels within the training set. For both training and validation sets, the model collected the R^2 scores for regression tasks $\{E, N\}$ and the accuracy, precision, recall, and F_1 score for the confidences. Lastly, the the model also collected the MAE of the nearest rounded number of resonances, an important factor in determining the models capability to inference to subsequent stages and comparison against the confidences.

Stage 2: The Stage 2 energy correction model required the simplest task analysis and was more prone to over-fitting. As such the network has been scaled down compared to Stage 1 and involves 1-80-160-320 2D convolutional layering structure with the same batch normalisation, ReLU activation, and pooling structure as seen in 23. Fully Connected layer 1 was reduced to 2048 and the dropout between each fully connected layers was increased to 50%. The weight decay in the optimiser was increased to 1e-4. $\sigma_{E,error}$ was set to 0.15 MeV and slice width was 16 bins, equivalent to 0.5 MeV.

The model evaluated its loss function over 50 epochs on the single regression output against ΔE with MSE. For both training and validation sets, the model collected the R^2 scores for energy offset, as well as the prediction error for comparative analysis against the detection model.

Stage 3: The Stage 3 feature extraction model mirrored the architecture of the Stage 1 detection model bar the outputs as seen in Figure 24 and a shorter epoch duration of 50. The outputs included a single regression prediction of Γ with softplus activation to prevent negative values and decrease the sensitivity of the smaller scaled values. Five logits were assigned to l prediction with a softmax activation to normalise the contribution to the loss function. Predicted l was taken from the argmax of the logits. A 30% dropout was used after fully connected layer

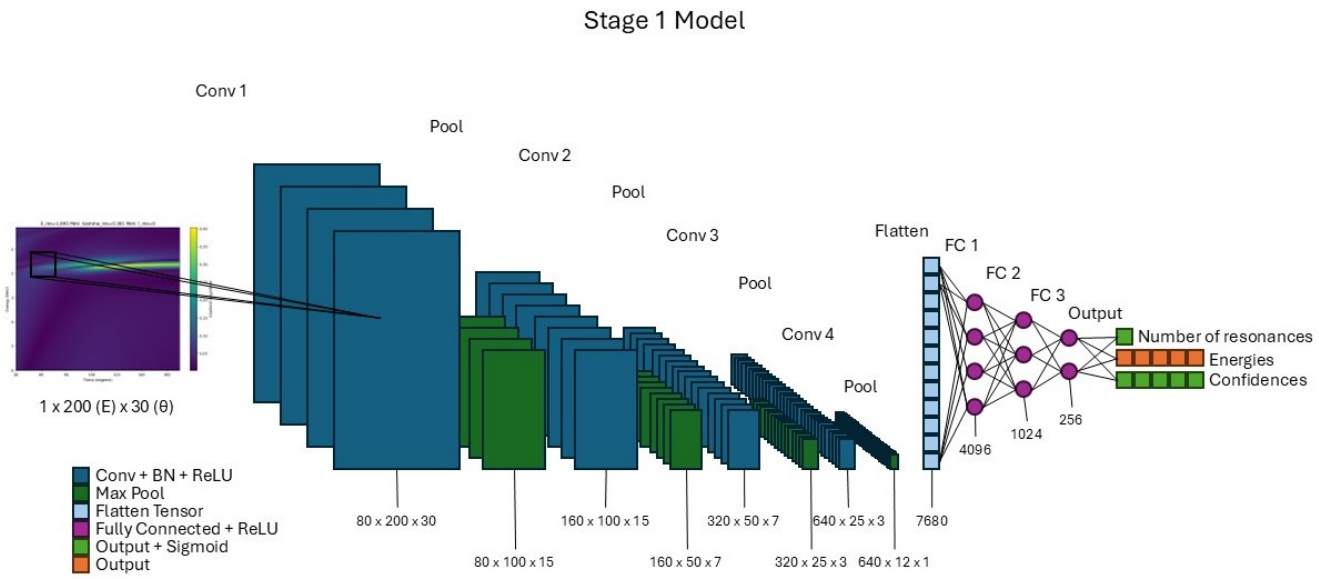


Figure 22: Stage 1 resonance detection model architecture. The convolution layers scan over the predecessor feature map while the fully connected layers are there to predict the regressional and logit outputs.

1 and 2. The weight decay in the optimiser was increased to 1e-4. $\sigma_{E,error}$ was set to 0.05 MeV and slice width was 16 bins. The $\sigma_{E,error}$ is chosen underneath the performance values of the Stage 2 model to preference accurate energy predictions over broad generalisation.

The model evaluated its loss function with MSE for Γ predictions and cross entropy for l logits. For both training and validation sets, the model collected the R^2 scores for Γ, and the accuracy, precision, recall, and F_1 score for l predictions.

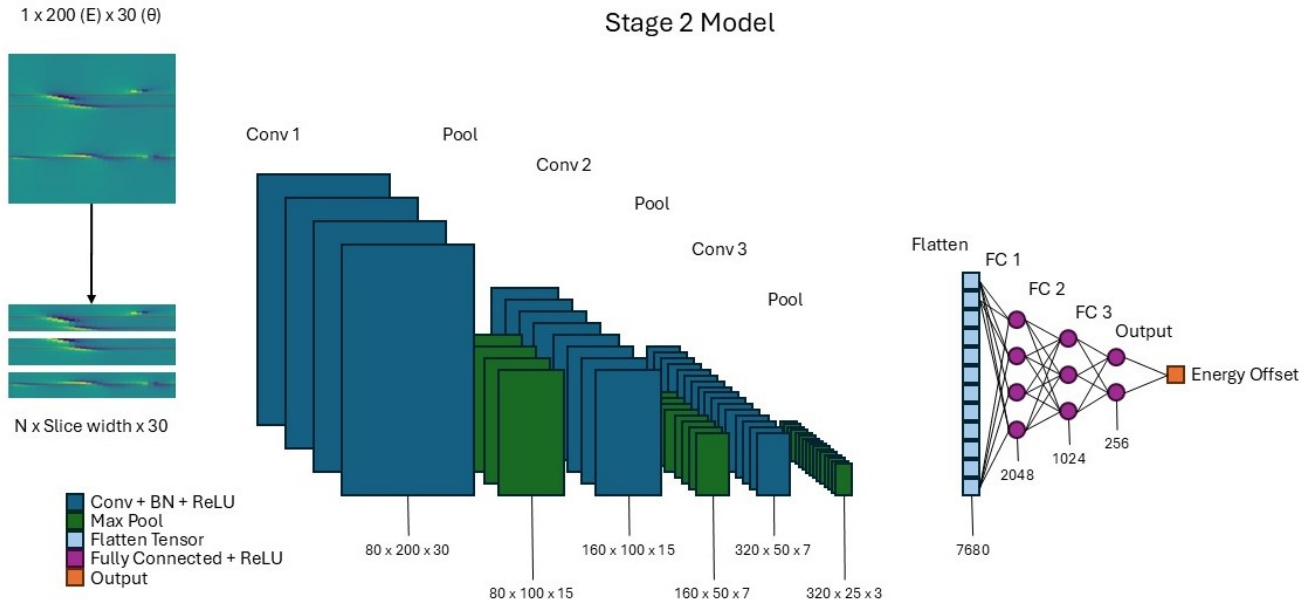


Figure 23: Stage 2 energy correction model architecture. The 4th convolutional layer has been dropped and FC1 has been scaled down to limited to diminish overfitting. The model has one primary task which is to decrease the energy error prediction.

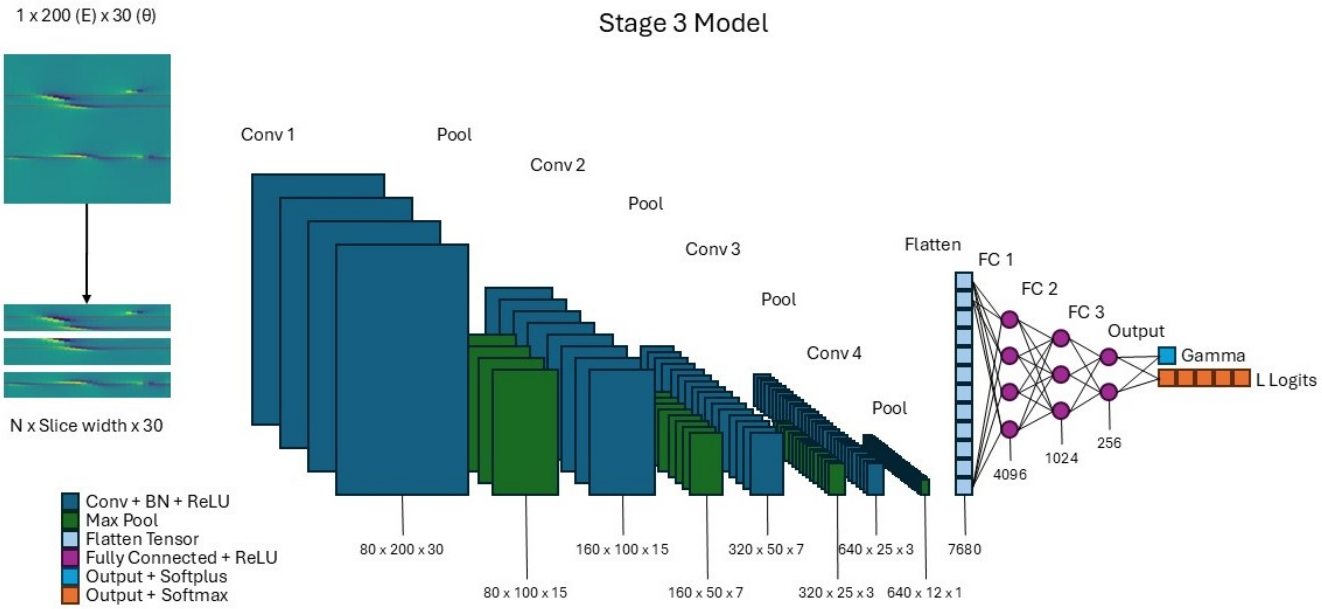


Figure 24: Stage 3 feature prediction model architecture. Γ produces a regression output while l provides logits representing the confidences of each guess.

5.2.3 Stage 1 Model Training and Validation Performance

The Stage 1 resonance detection model demonstrated strong performance across all evaluated metrics, indicating its effectiveness in accurately identifying the number of resonances and their approximate energies. Throughout the training process, the model showed consistent convergence without signs of overfitting, as evidenced by the training and validation loss curves in Figure 25.

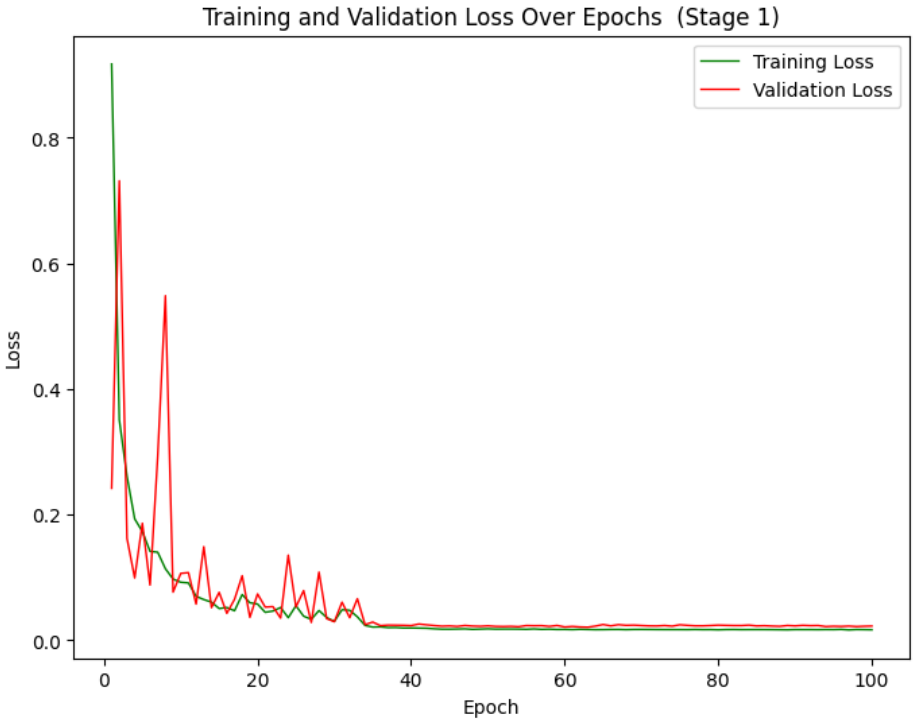


Figure 25: Training and validation loss curves for the Stage 1 model over 100 epochs. Both loss functions remain converged with no sign of overfitting.

The model achieved a high R^2 score for predicting the number of resonances, reaching a value of 0.9955 on the validation set, as shown in Figure 26. This indicates a strong correlation between the predicted and true number of resonances. The MAE for the energy predictions was 0.010, demonstrating the model's precision in estimating the number of resonances.

Similarly, the energy predictions exhibited high accuracy, with an R^2 score of 0.9908 on the validation set (Figure 27).

The confidence predictions, representing the model's certainty in the presence of resonances that can be used in downstream inferencing, achieved high performance metrics with an accuracy of 0.9977, precision of 0.9994, recall of 0.9967, and F_1-score of 0.9980 on the validation set. The confusion matrix in Figure 28 illustrates the predictive power of the model with all prediction-label combinations.

The histogram of prediction errors for energies (Figure 29) illustrates a small under-predictive bias with mean and median error of -0.08. The prediction errors for energies had a standard deviation of $\sigma_{E,error} = 0.11$ which falls within the current training bounds of the energy correction model at $\sigma_{E,error} = 0.15$.

Overall, the Stage 1 model effectively detects resonances with extremely high accuracy and reliability, providing a solid foundation for the subsequent stages in the multi-stage inferencing framework.

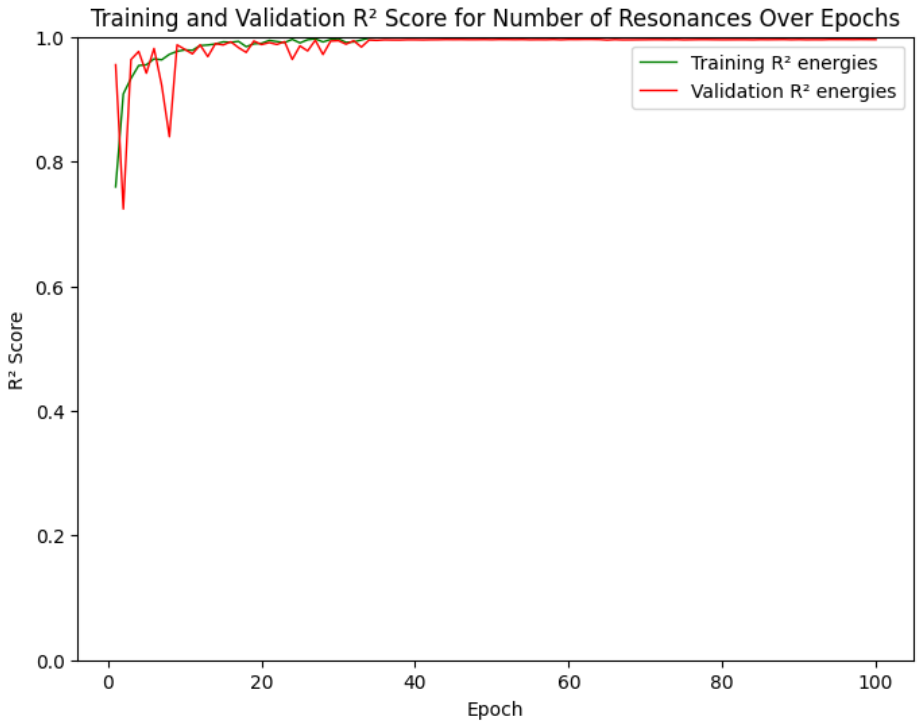


Figure 26: R^2 score for the prediction of the number of resonances during training and validation. No sign of loss divergence indicating an absence of overfitting. Model performs remarkably well.

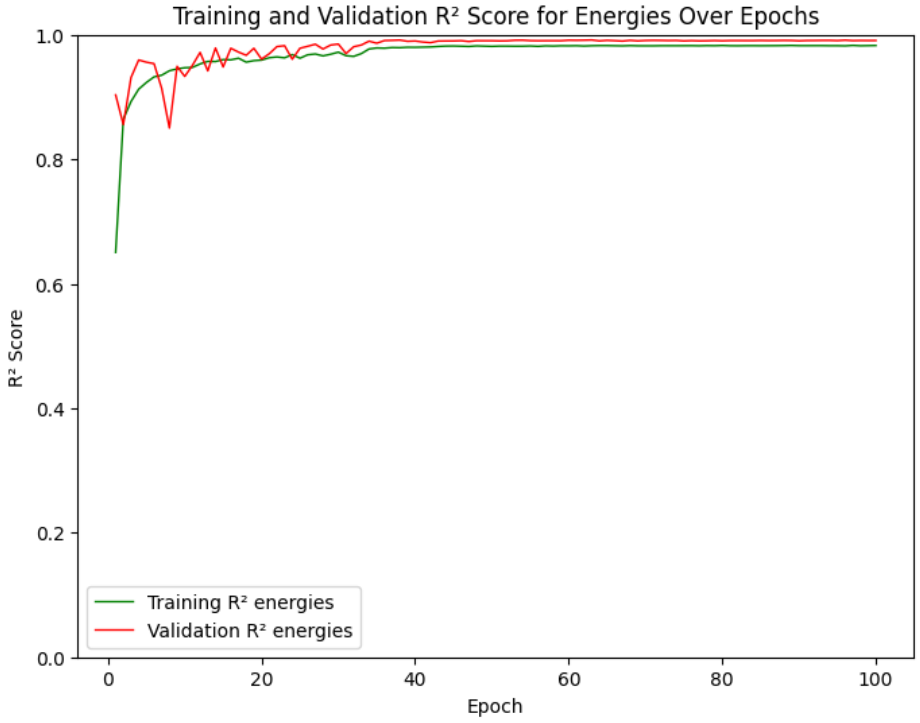


Figure 27: R^2 score for the energy predictions during training and validation. No overfitting was observed with a minimal generalisation gap between the training and validation sets.

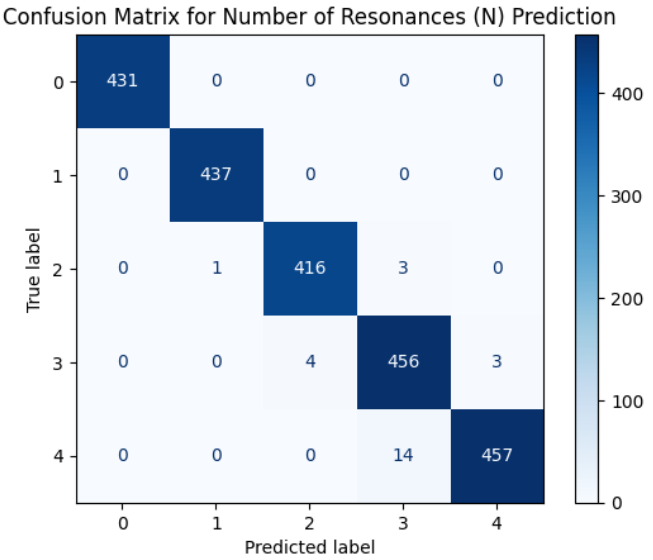


Figure 28: Confusion matrix of predicted and true labels for number of resonances during validation. The vast majority of cases predict the correct number of resonances. In those cases where the prediction is incorrect, the maximum prediction error is 1.

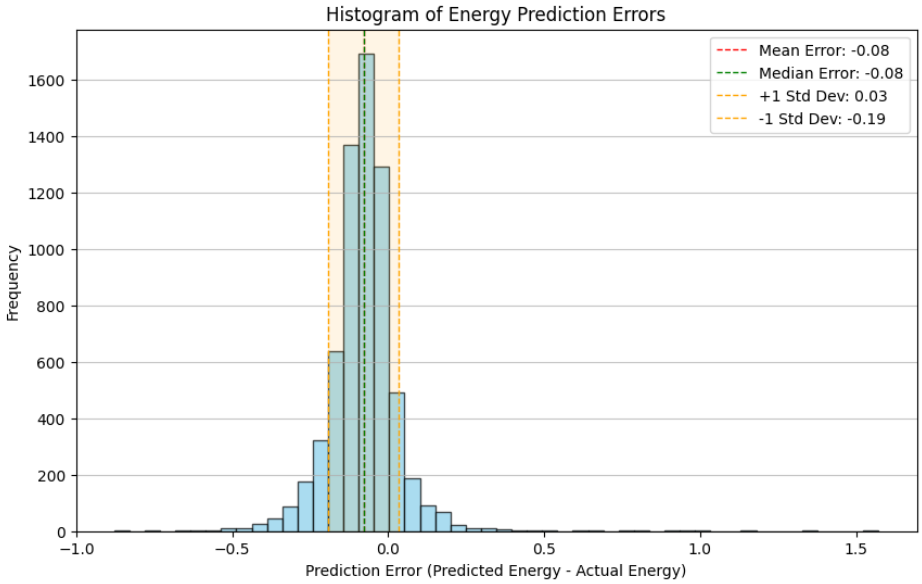


Figure 29: Histogram of Stage 1 prediction error of Energy outputs. Ideally, this would have a mean and medium of 0, with the smallest std dev possible to reduce error propagation.

5.2.4 Stage 2 Model Training and Validation Performance

The Stage 2 energy correction model focused on emulating residual error correction from the Stage 1 resonance detection prediction through image slice extracts offset by the normal distribution characteristics of Stage 1 error. Despite its reduced complexity compared to Stage 1, the model achieved solid performance by reducing the energy prediction error. The model rapidly converge during training. There are signs of overfitting so earlier model parameters at epoch 20 was selected, as shown in Figure 30.

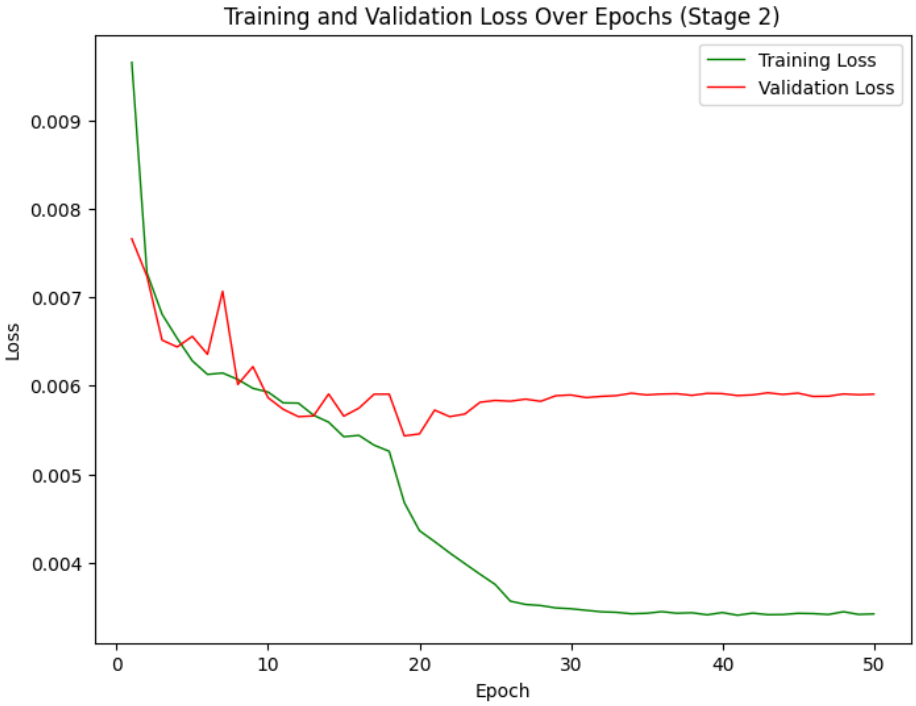


Figure 30: Training and validation loss curves for the Stage 2 model over 50 epochs. The model does overfit towards the latter epochs, so an early iteration (circa epoch 20) of the model parameters were saved and used in the multi-stage inferencing function.

The model achieved an R^2 score of 0.7411 for the energy offset predictions on the validation set (Figure 31), indicating a strong ability to predict the corrections to the initial energy estimates.

The regression plot in Figure 32 shows a close alignment between the predicted energy offsets and the true offsets, with data points clustering around the ideal diagonal line. While the R^2 score is lower than the other models regression performances, achieving any positive R^2 score constitutes a gain in the inferencing function which is the ultimate objective of the project.

The histogram of prediction errors for energies (Figure 33) illustrates a shift in the mean and median down to near 0 from Figure 29. Importantly, the standard deviation of errors dropped from $\sigma_{E,error} = 0.11$ to $\sigma_{E,error} = 0.08$ which improves model inputs for stage 3 training that lead to greater performance in feature detection as denoted by Table 3.

By reducing the residual error in energy predictions, the Stage 2 model has improved the performance for the Stage 3 feature extraction model, thereby improving the overall performance of the multi-stage inferencing framework.

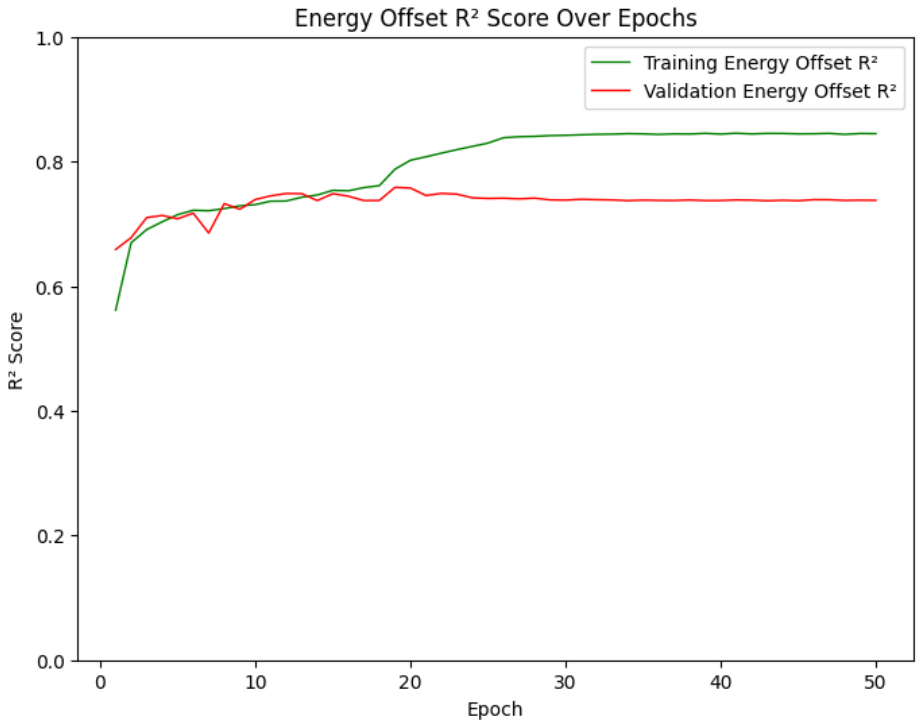


Figure 31: R^2 score for energy offset predictions. Similar to the loss function, the model display some overfitting towards the latter epochs so an earlier version was used in the multi-stage inferencing function.

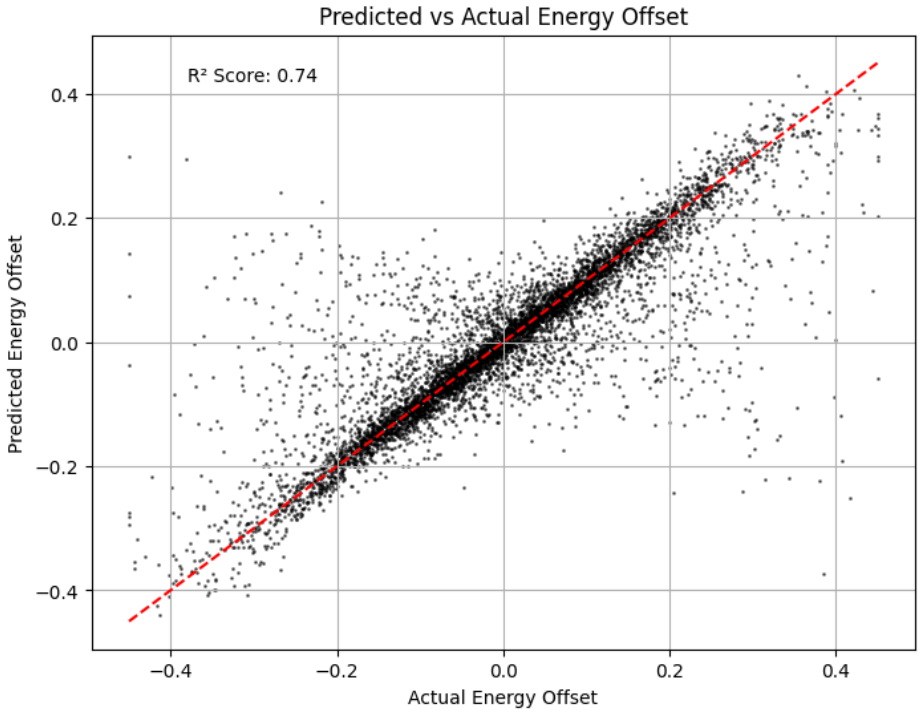


Figure 32: Regression plot comparing predicted energy offsets to true energy offsets. A weaker R^2 score than the other regression tasks in this paper, but the model still manages to improve the energy predictions to allow for better training regime for Stage 3.

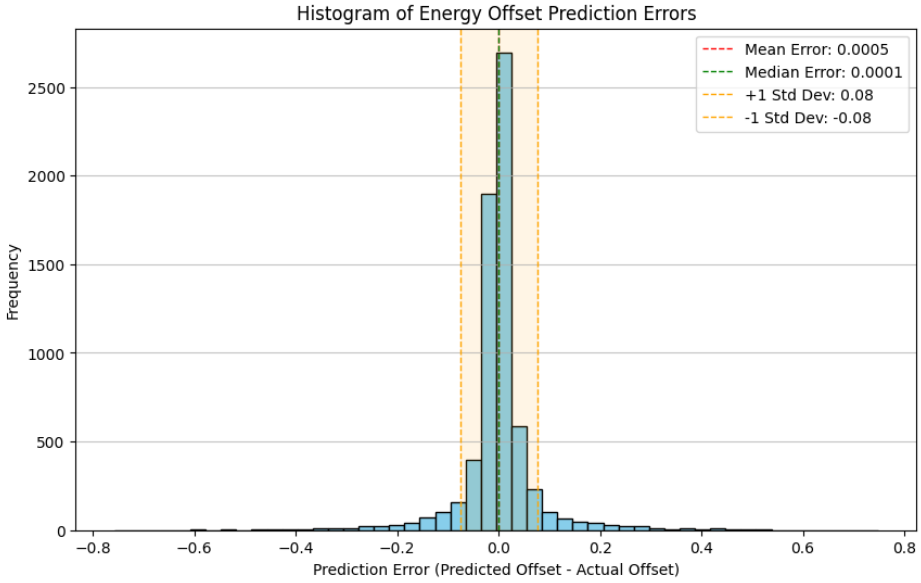


Figure 33: Histogram of prediction errors for the energy offset predictions. In contrast to the histogram of Stage 1, the mean and median are effectively 0 while the std dev has dropped from 0.11 to 0.08.

5.2.5 Stage 3 Model Training and Validation Performance

The Stage 3 feature extraction model was tasked with predicting the resonance width (Γ) and angular momentum (l) based on image slices similar to Stage 2, but with a significantly lower energy offset $\sigma_{E,error} = 0.05$. The model demonstrated strong performance in both regression and classification tasks. As there was signs of overfitting in the latter iterations (Figure 34), epoch 20 was chosen for for validation and inferencing.

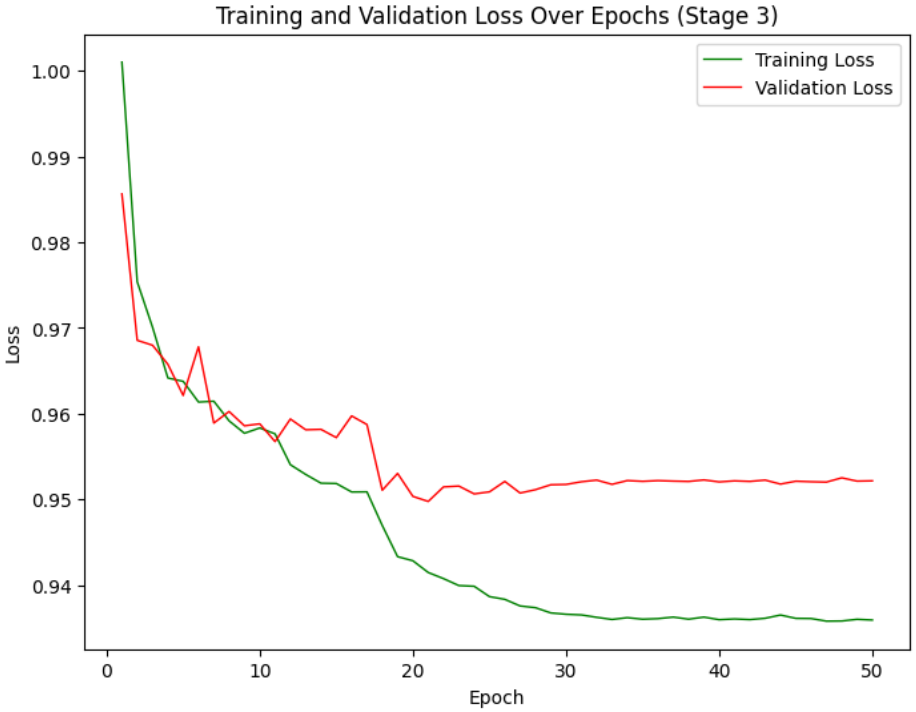


Figure 34: Stage 3 model loss over 50 epochs. The model shows parallel training-validation loss improvement until circa epoch 20 where the two functions diverge and indicate overfitting.

For the Γ predictions, the model achieved an R^2 score of 0.8680 on the validation set, as depicted in Figure 35. This high R^2 score indicates that the model captures the relationship between the input data and the resonance widths, particularly with the poor resolution of the image compared to some of the Γ values.

The angular momentum (l) classification achieved an accuracy of 0.9560 (Figure 37), with precision, recall, and F_1-score values of 0.9564, 0.9560, and 0.9559 respectively on the validation set. The confusion matrix in Figure 38 illustrates the model's performance across different l classes, showing high true positive rates and low misclassification rates.

The performance of the Stage 3 model validates the effectiveness of the multi-stage approach and outperformed all other approaches tested during development including the single resonance model. This demonstrates that accurate feature extraction is achievable when the model is provided with precise energy-corrected extracted inputs.

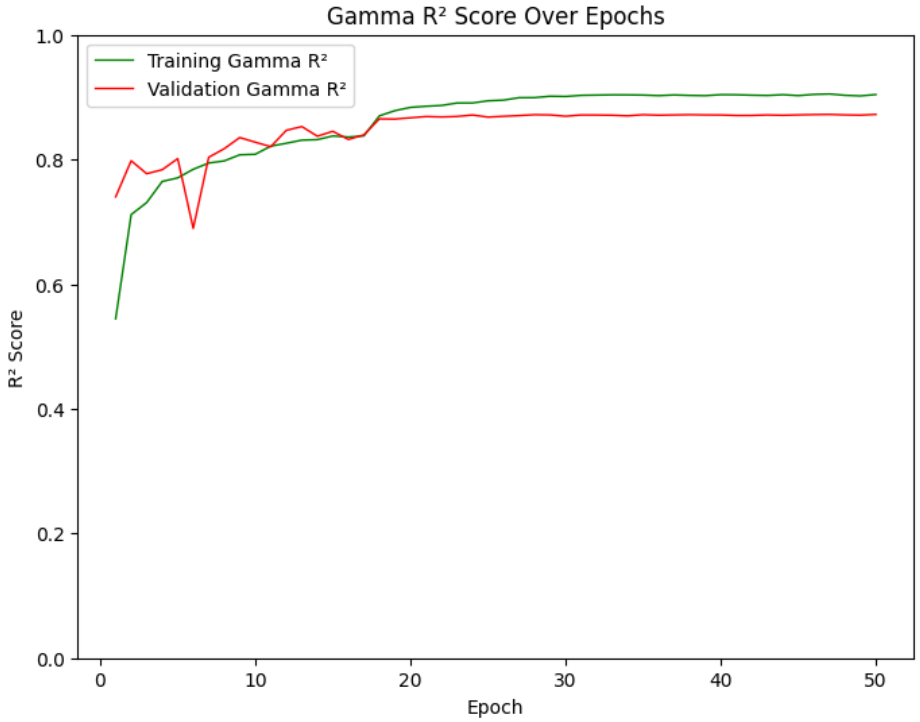


Figure 35: R^2 score for Γ predictions. Model shows a strong linkage with training and validation performance, indicating generalisation.

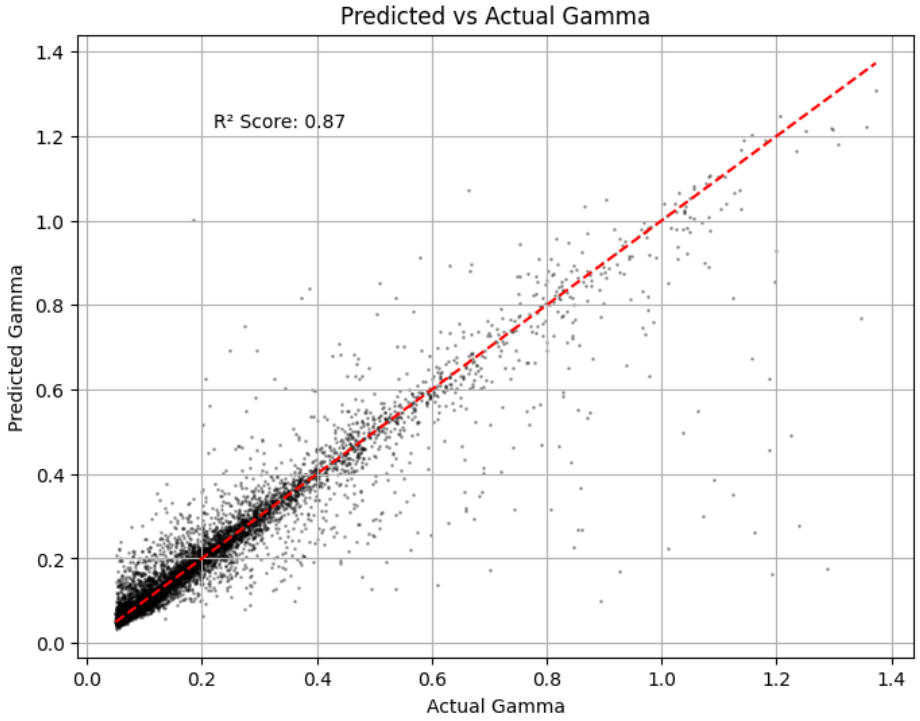


Figure 36: Regression plot for Γ predictions. Notice the scale of distribution with this plot with a heavy bias towards low energy widths, many of which are sensitive to the energy resolution of 0.03 MeV.

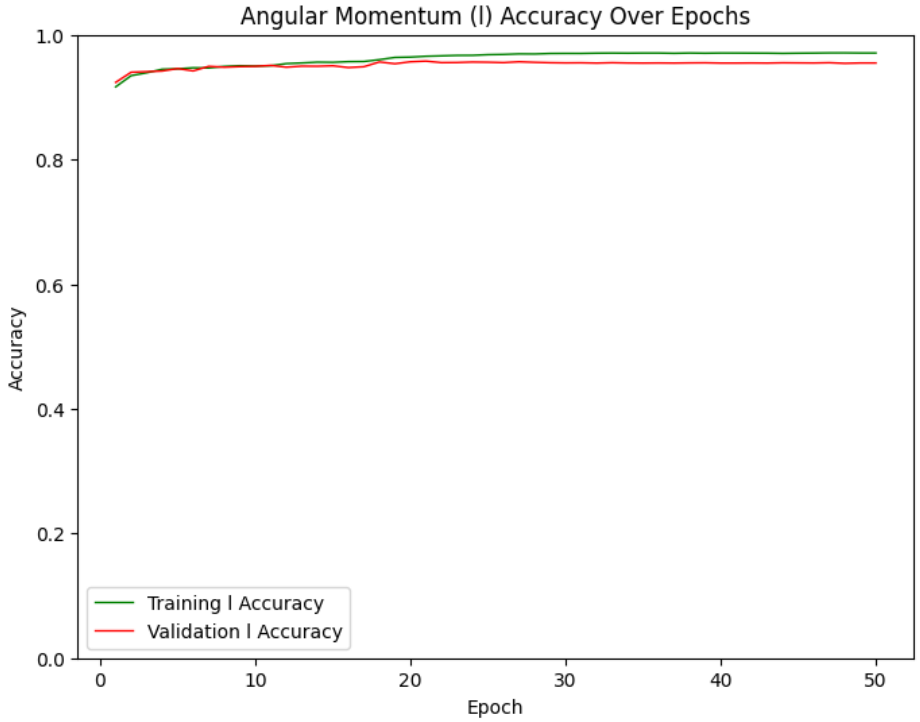


Figure 37: Accuracy score for l predictions. The classification schema performed remarkably well with a ¿90% after the first epoch.

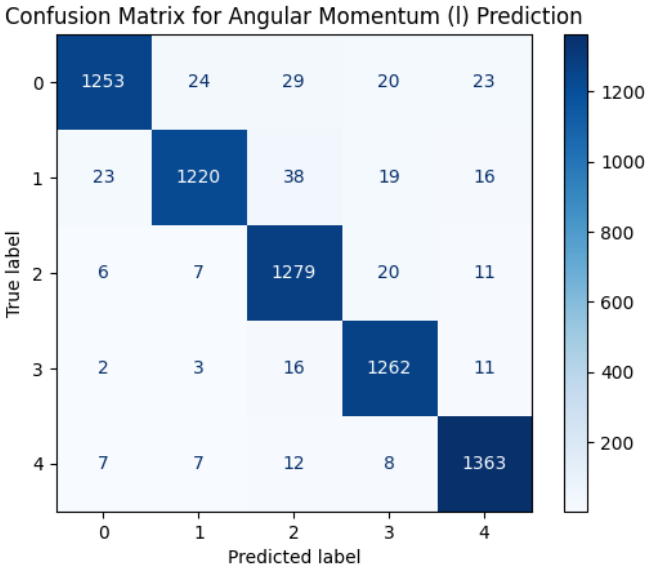
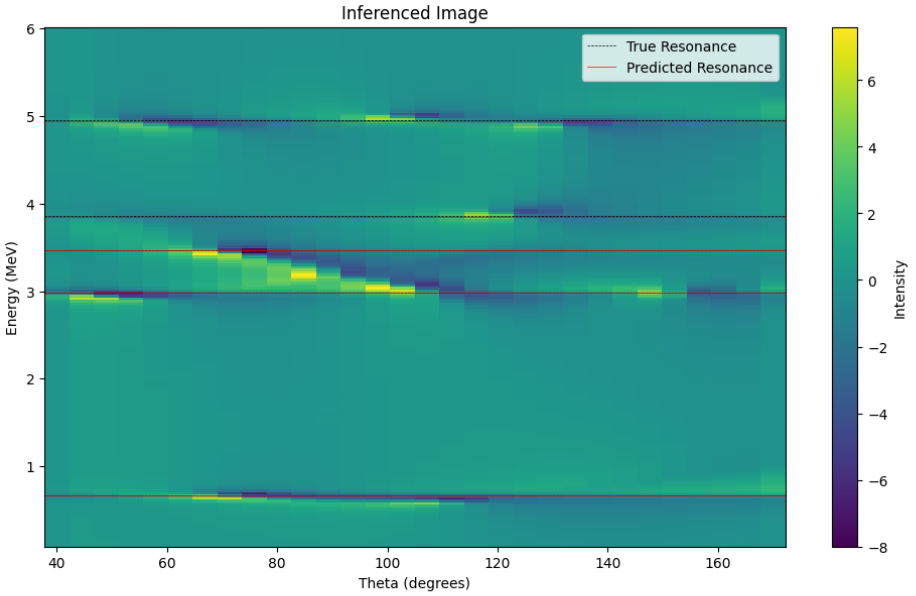


Figure 38: Confusion Matrix of predicted and true labels for angular momentum during validation. Notably this could improve with cost optimisation.

5.2.6 Multi-stage Inferencing Results and Discussion

After integrating the three stages into a cohesive inferencing pipeline, the multi-stage model was evaluated in a qualitative assessment through sampling as the culmination of the 3 different stage models performance has been discussed. As such, the purpose of this function is to observe that the multi-stage inferencing is working as intended, and highlight any impediments that may affect the model when testing on experimental data in the future.

Figure 39 demonstrates the model can handle high resonance features, particularly when two or more closely overlap and reduce the imperative for a cost function.



Resonance	True Energy	Gamma	l	Pred Energy	Gamma	l
1	0.669	0.103	2	0.670	0.078	2
2	2.981	0.108	4	2.982	0.063	4
3	3.462	0.251	2	3.464	0.220	2
4	3.855	0.335	1	3.846	0.390	1
5	4.956	0.156	3	4.938	0.123	3

Figure 39: Multi-stage inferencing of a 5 resonance image illustrating the models capacity to very accurately detect and predict various resonances across the energy spectrum.

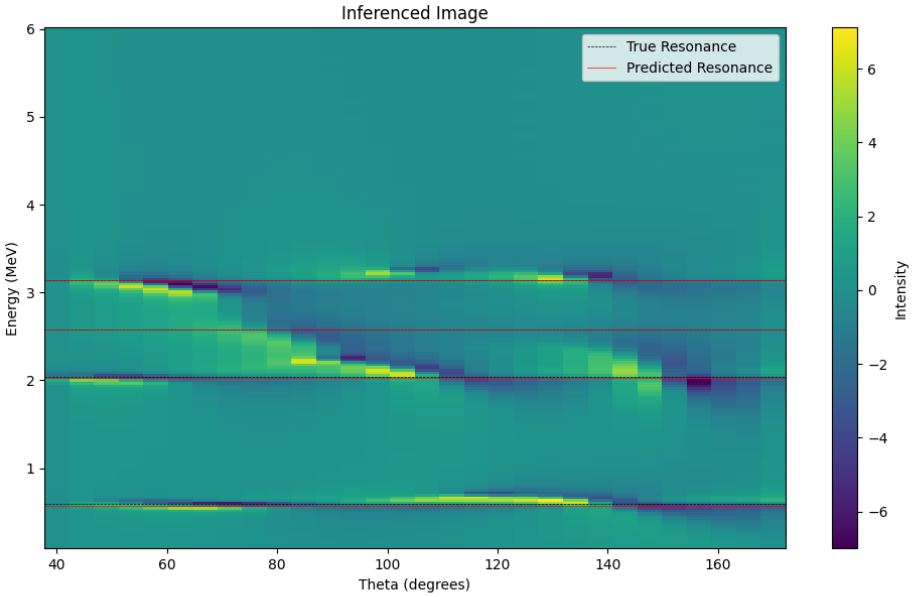
Figure 40 demonstrates the models capacity to accurately manage cases there is significant overlap of wide Γ resonances. The wider resonances are harder to detect with the Sobel filter as their slower incline or decline is more easily washed in the manifold.

Figure 41 shows the model can handle sparse resonances with very little interaction or overlap.

Figure 42 illustrates the challenges with assignment ambiguity as resonances 2 and 3 have both been correctly identified for the high performing metrics of E and l but do not present the cost optimal solution, signalling the need for cost optimisation.

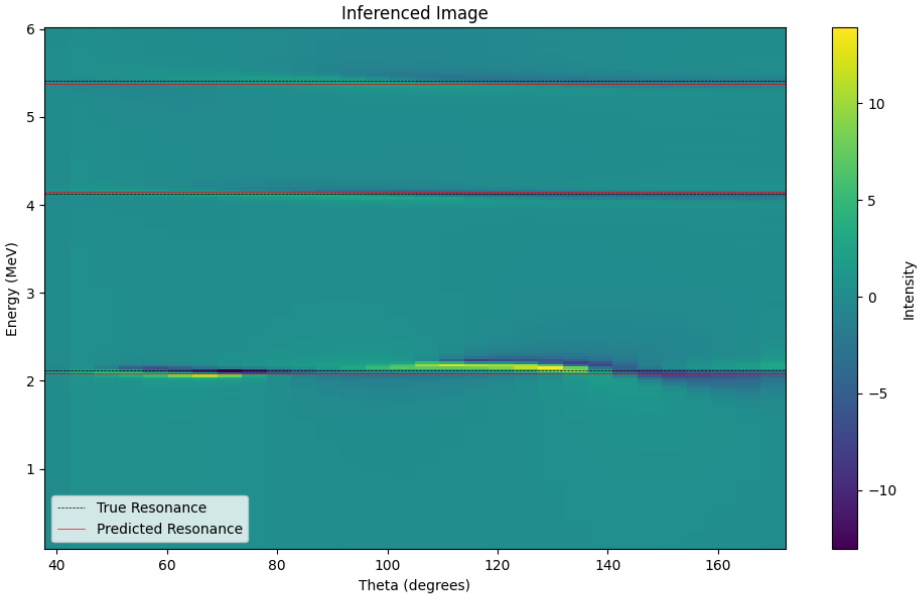
Figure 43 shows the model performing on a particularly challenge set where human observation could likely struggle to disentangle or correctly identify the resonance energies. Here, the largest energy error is 43 keV, an impressive feat contrasting the energy pixel is parsed over 30 keV steps. All l values are correctly predicted and Γ does follow the orders of magnitude expected.

As such, the model has performed excellently in rising to the challenge set by the novel application.



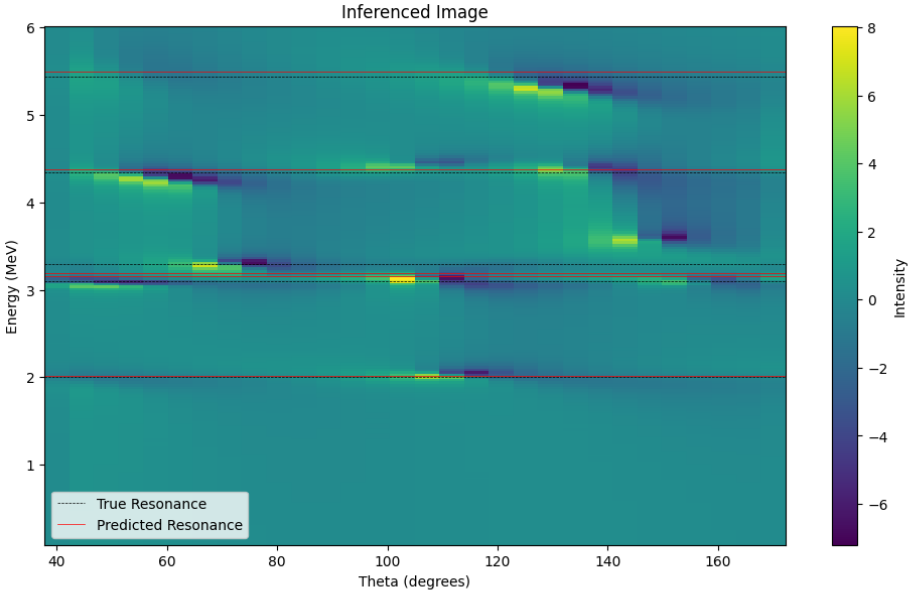
Resonance	True Energy	Gamma	l	Pred Energy	Gamma	l
1	0.598	0.054	3	0.575	0.034	3
2	2.042	0.096	4	2.032	0.069	4
3	2.581	0.583	0	2.583	0.671	0
4	3.139	0.193	3	3.142	0.131	3

Figure 40: Multi-stage inferencing of a 4 resonance image with wide overlaps. Resonance 3 has a peak at 2.581 MeV that cascades into the peak at 2.042 MeV, adding complexity to disentangle after Sobel filtering.



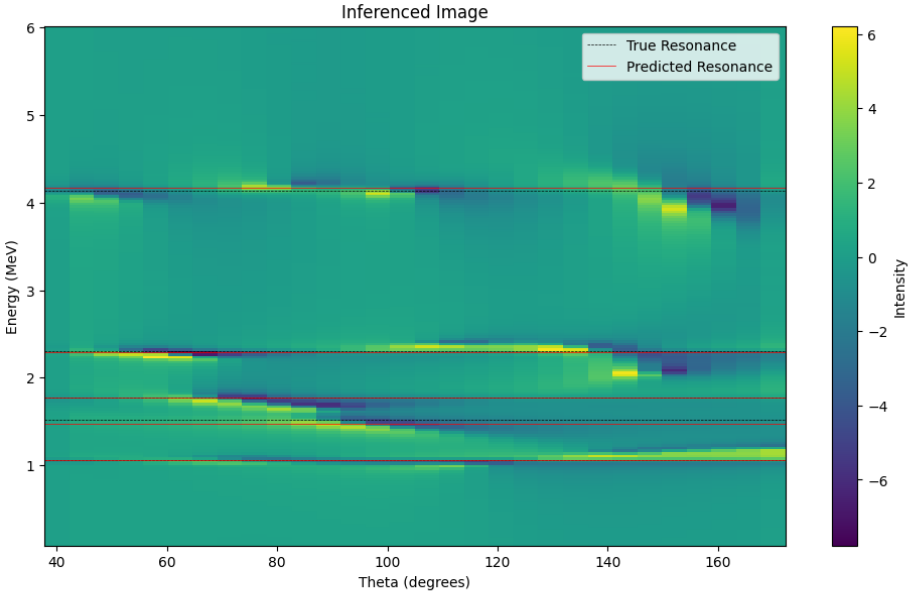
Resonance	True Energy	Gamma	l	Pred Energy	Gamma	l
1	2.117	0.060	3	2.091	0.011	3
2	4.127	0.097	0	4.144	0.072	0
3	5.412	0.179	0	5.375	0.169	0

Figure 41: Multi-stage inferencing of a 3 resonance image with sparse resonances, model is adaptable to both high and low frequency fluctuation of the cross-section amplitude.



Resonance	True Energy	Gamma	l	Pred Energy	Gamma	l
1	2.008	0.255	1	2.020	0.277	1
2	3.099	0.087	4	3.194	0.064	2
3	3.297	0.221	2	3.158	0.035	4
4	4.343	0.240	3	4.374	0.154	3
5	5.431	0.701	1	5.496	0.539	1

Figure 42: Multi-stage inferencing of a 5 resonance image where the model has been punished by an assignment error between resonance 2 and 3.



Resonance	True Energy	Gamma	l	Pred Energy	Gamma	l
1	1.062	0.052	2	1.058	0.085	2
2	1.523	0.318	0	1.480	0.237	0
3	1.772	0.178	2	1.776	0.150	2
4	2.304	0.092	3	2.290	0.039	3
5	4.133	0.179	4	4.169	0.136	4

Figure 43: Multi-stage inferencing of a 5 resonance image with a topology that would be challenge to discern by human observation

6 Conclusion and Futre direction

This work demonstrates the potential of convolutional neural networks for resonance identification in nuclear scattering events with predictive features of energy, width, and angular momentum. The multi-stage inferencing model that combine three CNNs that have been trained on synthesised datasets, shows high accuracy in predicting the number of resonances in an image as well as their feature properties. This multi-stage architecture approach, when combined with various pre-processing steps, outperforms single multi-task CNN model. It also more closely mimics the typical approach that human evaluators take, in using the cross-section to identify peaks, then angular distribution and neighbouring data to identify l and Γ suggesting that the approach can be further generalised to deal with more complex reactions. Moving forward, a natural extension involves incorporating real experimental data to evaluate the model's practical application in field studies.

The future direction of this project has four major avenues of improvement for the next stage of development. Firstly, constructing a test suite to benchmark the inferencing pipeline performance with weighted criterion by the research objective. This will allow for a more thorough sensitivity analysis of all the model hyperparameters and network architecture for computational cost minimisation. Alongside this, it will open the door for comparison against other neural networks models that may be applicable to this project as suggestion in Section 5.1.3.

Secondly, mechanisms to expand the training data to include other collisions. Currently, the dataset is only comprised of $^{12}C(\alpha, \alpha_0)^{12}C$ simulated data and ideally this will need to expand. Apart of this includes other quantum properties of the nuclei such as reaction channels and spin.

Thirdly, implementing the cost minimisation algorithm on predictions. This function will come with great importance when disentangling neighbouring resonances for better feature extraction.

Lastly and perhaps most importantly, evaluating experimental data from IBANDL with the model. Ideally this would be included in this report, however there are stringent challenges with sparse data interpolation and collection to create suitable inputs for the model.

This project has demonstrated the promise of such approach.

References

[1] John M. Blatt and Victor F. Weisskopf. *Theoretical Nuclear Physics*. John Wiley & Sons, 1952.

[2] Gregory Breit and Eugene P. Wigner. Capture of slow neutrons. *Physical Review*, 49(7):519–531, 1936.

[3] Herman Feshbach. Unified theory of nuclear reactions. *Annals of Physics*, 5(4):357–390, 1958.

[4] N. F. Mott and H. S. W. Massey. *The Theory of Atomic Collisions*. Oxford University Press, London, 3rd edition, 1965.

[5] Roger G. Newton. *Scattering Theory of Waves and Particles*. Springer-Verlag, New York, 2nd edition, 1982.

[6] Walter Greiner. *Quantum Mechanics: An Introduction*. Springer Science & Business Media, 2012.

[7] C. J. Joachain. *Quantum Collision Theory*. North-Holland, Amsterdam, 1975.

[8] J. Knoll. Lecture notes on scattering theory. `https://theory.gsi.de/~knoll/Lecture-notes/3-scattering.pdf`, 2024. Accessed: 2024-10-13.

[9] Kenneth S. Krane. *Introductory Nuclear Physics*. Wiley, 1987.

[10] John S. Lilley. *Nuclear Physics: Principles and Applications*. Wiley, 2001.

[11] Claus E. Rolfs and William S. Rodney. *Cauldrons in the Cosmos: Nuclear Astrophysics*. University of Chicago Press, 1988.

[12] A. M. Lane and R. G. Thomas. R-matrix theory of nuclear reactions. *Reviews of Modern Physics*, 30(2):257–353, 1958.

[13] P Descouvemont and D Baye. Ther-matrix theory. *Reports on Progress in Physics*, 73(3):036301, February 2010.

[14] Ion beam analysis nuclear data library (ibandl). `https://www-nds.iaea.org/ibandl/`. Accessed: October 13, 2023.

[15] Sigmacalc: Web calculator of nuclear reaction cross sections. `http://sigmacalc.iate.obninsk.ru/`. Accessed: October 13, 2023.

[16] twobody Developers. twobody: A python library for two-body scattering calculations. `https://twobody.readthedocs.io/en/stable/`, 2023. Accessed: October 13, 2023.

[17] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006.

[18] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, 2016.

[19] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, Cambridge, MA, 2012.

[20] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York, 2nd edition, 2009.

[21] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[22] Johan Schmidt, Miguel R. G. Marques, Silvana Botti, and Micael A. L. Marques. Recent advances and applications of machine learning in solid-state materials science. *npj Computational Materials*, 5:83, 2019.

[23] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, Upper Saddle River, NJ, 3rd edition, 2016.

[24] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[25] Riza Utama, Jorge Piekarewicz, and Harrison B. Prosper. Nuclear mass predictions for the crustal composition of neutron stars: A bayesian neural network approach. *Physical Review C*, 93(1):014311, 2016.

[26] Léo Neufcourt, Yuchen Cao, Witold Nazarewicz, and Frederi Veins. Bayesian approach to model-based extrapolation of nuclear observables. *Physical Review Letters*, 122(6):062502, 2019.

[27] Enrico Zio and Francesco Di Maio. A data-driven fuzzy approach for predicting the remaining useful life in dynamic failure scenarios of a nuclear system. *Reliability Engineering & System Safety*, 96(7):935–946, 2011.

[28] Dan Gabriel Cacuci, editor. *Handbook of Nuclear Engineering*. Springer, Boston, MA, 2010.

[29] Abdullah Abdulaziz, Jianxin Zhou, Angela Di Fulvio, Yoann Altmann, and Stephen McLaughlin. Semi-supervised gaussian mixture variational autoencoder for pulse shape discrimination. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3538–3542. IEEE, 2022.

[30] X Fabian, G Baulieu, L Ducroux, O Stézowski, A Boujrad, E Clément, S Coudert, G de France, N Erduran, S Ertürk, et al. Artificial neural networks for neutron/γ discrimination in the neutron detectors of neda. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 986:164750, 2021.

[31] Changsong Jin, Tiejun Li, Jianmin Zhang, Wei Zhang, Bo Yang, Ruixuan Ren, and Cunhao Cui. Fecsg-ml: Feature engineering for nuclear reaction cross sections generation using machine learning. *Applied Radiation and Isotopes*, page 111545, 2024.

[32] Amber Boehnlein, Markus Diefenthaler, Nobuo Sato, Malachi Schram, Veronique Ziegler, Cristiano Fanelli, Morten Hjorth-Jensen, Tanja Horn, Michelle P. Kuchera, Dean Lee, Witold Nazarewicz, Peter Ostroumov, Kostas Orginos, Alan Poon, Xin-Nian Wang, Alexander Scheinker, Michael S. Smith, and Long-Gang Pang. Colloquium: Machine learning in nuclear physics. *Rev. Mod. Phys.*, 94:031003, Sep 2022.

[33] Abbas J. Jinia, Shaun D. Clarke, Jean M. Moran, and Sara A. Pozzi. Intelligent radiation: A review of machine learning applications in nuclear and radiological sciences. *Annals of Nuclear Energy*, 201:110444, 2024.

[34] Yong Hyun Kim, Dong Geon Kim, Kihong Pak, Jae Young Jeong, Jae Chang Kim, Han Cheol Yang, Seung Beom Goh, and Yong Kyun Kim. Identification of multiple radioisotopes through convolutional neural networks trained on 2-d transformed gamma spectral data from csi(tl) spectrometer. *Radiation Physics and Chemistry*, 210:111054, 2023.

[35] Andy Rivas, Gregory Kyriakos Delipei, Ian Davis, Satyan Bhongale, and Jason Hou. A system diagnostic and prognostic framework based on deep learning for advanced reactors. *Progress in Nuclear Energy*, 170:105114, 2024.

[36] Huasong Cao, Peiwei Sun, and Liang Zhao. Pca-svm method with sliding window for online fault diagnosis of a small pressurized water reactor. *Annals of Nuclear Energy*, 171:109036, 2022.

[37] Michela Paganini, Luke de Oliveira, and Benjamin Philip Nachman. Accelerating science with generative adversarial networks: An application to 3d particle showers in multilayer calorimeters. *Physical review letters*, 120 4:042003, 2017.

[38] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[39] Wikipedia contributors. Hadamard product (matrices) — Wikipedia, the free encyclopedia, 2024. [Online; accessed 22-October-2024].

[40] Wikipedia contributors. Kernel (image processing) — Wikipedia, the free encyclopedia, 2024. [Online; accessed 22-October-2024].

[41] Papers with Code. He initialization, 2024. [Online; accessed 22-October-2024].

[42] Wikipedia contributors. Curse of dimensionality — Wikipedia, The Free Encyclopedia. `https://en.wikipedia.org/wiki/Curse_of_dimensionality`, 2023. [Online; accessed 1-October-2023].

[43] Stack Overflow. Parameter contour plot, 2018. Image, Accessed: 2024-10-23.

[44] Sunitha Basodi, Chunyan Ji, Haiping Zhang, and Yi Pan. Gradient amplification: An efficient way to train deep neural networks. *Big Data Mining and Analytics*, 3(3):196–207, 2020.

[45] Song Han, Jeff Pool, John Tran, and William J Dally. Learning both weights and connections for efficient neural networks. In *Advances in Neural Information Processing Systems*, volume 28, pages 1135–1143, 2015.

[46] Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

[47] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[48] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[49] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[50] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alex Smola. Deep sets. *arXiv preprint arXiv:1703.06114*, 2017.

[51] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[52] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015.

[53] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 779–788, 2016.

[54] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot multibox detector. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 21–37. Springer, 2016.

[55] Gioconda Campanella, Michael G Hanna, Lee Geneslaw, Amrita Miraflor, Viviane W Silva, Klaus J Busam, Edi Brogi, Victor E Reuter, David S Klimstra, and Thomas J Fuchs. Monte-carlo sampling applied to multiple instance learning for histological image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[56] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems (NeurIPS)*, 14:849–856, 2002.

[57] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.