AUTOMATED ISOTOPE IDENTIFICATION ALGORITHM USING ARTIFICIAL NEURAL NETWORKS

BY

MARK KAMUDA

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Nuclear Plasma and Radiological Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2017

Urbana, Illinois

Master's Committee:

Assistant Professor Clair J. Sullivan, Adviser
Assistant Professor Kathryn Huff

ABSTRACT

There is a need to develop an algorithm that can determine the relative activities of a mixture of many isotopes in a low-resolution gamma-ray spectrum. While techniques for this task exist, they require a human operator and are too slow to use on very large datasets of spectra. Pattern recognition algorithms such as neural networks are prime candidates for automated isotope identification using low-resolution gamma-ray spectra. While algorithms based on feature extraction such as peak finding or ROI algorithms work well for well calibrated high resolution detectors, for low-resolution detectors it may be more beneficial to use algorithms that incorporate more abstract features of the spectrum. This is especially true when analyzing a mixture of isotopes where peak overlap and Compton continuum effects occlude features of interest. To solve this, an artificial neural network (ANN) was trained to predict the presence and relative activities of isotopes from a mixture of many isotopes. The ANN is trained with simulated gamma-ray spectra, allowing easy expansion of the library of target isotopes. In this thesis, an algorithm based on an ANN is presented and evaluated against a series of measured spectra.

ACKNOWLEDGMENTS

First, I would like to thank my adviser Professor Clair Sullivan for her patience and support with my project. I'm grateful for all the work Clair has done to motivate and challenge me as a student and person.

I would also like to thank Professor Kathryn Huff and Professor James F. Stubbins for their feedback and comments on my thesis.

I would also like to greatly thank Jacob Stinnett for inspiring the work presented here. Our discussions were immeasurably helpful.

I would also like to thank my family, friends, and loved ones for their continued support.

TABLE OF CONTENTS

LIST OF TABLES v

LIST OF FIGURES vi

CHAPTER 1 INTRODUCTION 1
1.1 Introduction and Motivation 1
1.2 Neural Network History 2
1.3 Existing Methods for Automated Isotope Identification and Quantification . 3
1.4 Existing Neural Network Applications to Isotope Identification and Quantification 4
1.5 Proposed Solution 5

CHAPTER 2 THEORY 7
2.1 Neural Networks 7
2.2 Training Set Creation 21
2.3 Chosen Network Architecture 27
2.4 Training Details 27
2.5 Random Hyperparameter Search Results 28
2.6 Summary 29

CHAPTER 3 RESULTS AND DISCUSSION 30
3.1 Identification Performance 30
3.2 Measurement of sensitivity to gain shift 42

CHAPTER 4 CONCLUSIONS AND FUTURE WORK 45

REFERENCES 47

LIST OF TABLES

2.1 Range of hyperparameter values tested in a random search optimization . . . 28

3.1 ANN output from Figure 3.2 representing relative gross count contribution . 32
3.2 ANN output from Figure 3.3 representing relative gross-count contribution . 33
3.3 ANN output from Figure 3.4 representing relative gross-count contribution . 34
3.4 ANN output from Figure 3.5 representing relative gross-count contribution . 35
3.5 ANN output from a 5 second gamma-ray spectrum of a 0.288 μCi ^{60}Co source measured from a distance of 7.5 cm from the detector face and a 0.890 μCi ^{137}Cs source. 36
3.6 ANN output from a 60 second gamma-ray spectrum of a 0.288 μCi ^{60}Co source measured from a distance of 7.5 cm from the detector face and a ^{137}Cs source. 37
3.7 ANN output from Figure 3.8 representing relative gross-count contribution . 38
3.8 ANN output from the BeRP ball. 39
3.9 ANN output from the three ^{60}Co spectra in Figure 3.10. 41
3.10 ANN output from the three ^{152}Eu spectra in Figure 3.11. 42
3.11 Summary of the top 4 ANN outputs from a ^{60}Co source using different PMT voltage settings. 44

LIST OF FIGURES

2.1 Example ANN with input neurons A_n, hidden neurons B_j, and output neurons C_k. 7
2.2 Summary of the operation of a single neuron. 8
2.3 Example of a single-layer neural network with two inputs (x_1 and x_2), three classes (y_1, y_2, y_3), and a bias neuron set to one. 10
2.4 One possible dataset describing a three class function. Each class is represented by a different color. 10
2.5 Ideal training and testing error curves. 14
2.6 Example training paths for a large learning rate, red, and a small learning rate, green. 15
2.7 Two examples of data augmentation using an image of a cat. The image to the left is the original. The top right image is augmented using a horizontal flip. The bottom right image is augmented using blur. 19
2.8 A comparison between a grid search and a random search for hyperparameter optimization when performance is strongly tied to one hyperparameter. The green function represents the effect of an important hyperparameter on a cost function while the yellow function represents the effect for an unimportant hyperparameter. Figure reproduced from [42]. 21
2.9 Normalized converged MCNP6 simulation of a ^{60}Co spectrum. 23
2.10 Two sampled ^{60}Co spectra using the converged ^{60}Co spectrum shown in Figure 2.9. The top spectrum was simulated with a total of 10^3 counts and the bottom spectrum was simulated with a total of 10^5 counts. 24
2.11 A 6 hour spectrum of natural background. This background spectrum was used to generate the background isotope used in the ANN training set. . . . 25
2.12 In black, a ^{60}Co spectrum recorded using a NaI(Tl) detector for 60 seconds with the detector PMT set at 740V. In red, a simulated ^{60}Co spectrum with 70000 total counts shifted by 0.825. 26
2.13 In black, a ^{60}Co spectrum recorded using a NaI(Tl) detector for 60 seconds with the detector PMT set at 760V. In red, a simulated ^{60}Co spectrum with 55000 total counts and no shifting. 26
2.14 Training error curves using optimal hyperparameters found in the random search. The training dataset error is shown in blue and the testing dataset error is shown in red. 29

3.1 False alarm rate given a minimum relative count contribution. 31
3.2 A 6 hour background spectrum. Total number of counts in this spectrum is 1742651. 31
3.3 A 500 second gamma-ray spectrum of a 0.288 μCi ^{60}Co source measured from a distance of 7.5 cm from the detector face. There are 90003 gross counts in this spectrum. 32
3.4 A 500 second gamma-ray spectrum of a 8.84 μCi ^{152}Eu source measured from a distance of 74 cm from the detector face. There are 76710 gross counts in this spectrum. 34
3.5 A 10 second gamma-ray spectrum of a 8.84 μCi ^{152}Eu source measured from a distance of 74 cm from the detector face. There are 1569 gross counts in this spectrum. 35
3.6 A 5 second gamma-ray spectrum of a 0.288 μCi ^{60}Co source measured from a distance of 7.5 cm from the detector face and a 0.890 μCi ^{137}Cs source. There are 1550 gross counts in this spectrum. 36
3.7 A 60 second gamma-ray spectrum of a 0.288 μCi ^{60}Co source measured from a distance of 7.5 cm from the detector face and a 0.890 μCi ^{137}Cs source. There are 18087 gross counts in this spectrum 37
3.8 A 60 second gamma-ray spectrum of a 0.750 μCi ^{133}Ba source measured from a distance of 17 cm from the detector face and a 0.288 μCi ^{60}Co source measured from a distance of 7.5 cm from the detector face. There are 16787 gross counts in this spectrum. 38
3.9 60s spectrum of the BeRP ball. 39
3.10 Three spectra of a ^{60}Co measured 10 cm from the detector face for 480 seconds. The green spectrum is a bare ^{60}Co source, the red spectrum is the ^{60}Co source attenuated by a 2.27 mm sheet of lead, and the blue spectrum is the ^{60}Co source attenuated by a 5.33 mm sheet of lead. 40
3.11 Three spectra of a ^{152}Eu measured 15 cm from the detector face for 60 seconds. The green spectrum is a bare ^{152}Eu source, the red spectrum is the ^{152}Eu source attenuated by a 2.27 mm sheet of lead, and the blue spectrum is the ^{152}Eu source attenuated by a 5.33 mm sheet of lead. 42
3.12 Two 60 second ^{60}Co spectra. The PMT was biased at 730 V for the black spectrum and 765 V for the grey spectrum. 43

CHAPTER 1

INTRODUCTION

1.1 Introduction and Motivation

Immediately after a nuclear detonation, first responders may be collecting a large number of gamma-ray spectra using handheld low-resolution detectors. These detectors may have an unknown or poor calibration and each spectrum may be measured for a short amount of time. Despite these drawbacks, the data is still valuable because it can be used to determine isotopics of the debris generated by the explosion [1]. The isotopic analysis from these spectra can then be used to characterize the nuclear explosive device and inform nuclear forensics before more precise chemical analysis can be performed. Trained experts can determine the isotopics of the debris by analyzing the photopeaks in a spectrum. Due to the complicated gamma-ray spectrum produced by a large number of radioactive fission products, many photopeaks and spectral features will overlap in a spectrum. This feature overlap increases the difficulty of and slows photopeak analysis. Cheaper gamma-ray detectors typically have lower-resolution than their more expensive counterparts, increasing the chances of overlapping photopeaks and difficulty of analysis as the number of photopeaks increases. Despite this, due to the potential ubiquity of low-resolution detectors over higher-resolution detectors (due to portability and price), the ability to find useful information from low-resolution detectors may aid nuclear forensics efforts.

Traditional automated gamma-ray spectroscopy techniques are not optimized for identifying mixtures of isotopes in low-resolution detectors. Furthermore, isotope quantification typically requires an expert to manually identify and fit photopeaks in a spectrum to calculate the amount of each constituent isotope present in a spectrum. Machine learning algorithms such as artificial neural network (ANNs) have proven their ability to solve a large

variety of complicated problems. For problems where isotopes of interest are difficult to acquire for academic research, such as isotopes important to nuclear forensics, simulated gamma-ray spectra can be used to train an ANN. While some work has been done applying machine learning algorithms to spectroscopy problems, the topic of simulated training data and an ANNs ability to perform identification and quantification on mixtures of isotopes using low-resolution detectors has not been sufficiently explored [2, 3, 4]. An ANN tailored to perform isotope identification and quantification on mixtures of isotopes using low-resolution gamma-ray spectra has the potential to operate with minimal human intervention.

1.2 Neural Network History

Artificial neural networks were first theorized in the 1940s as a model of how complex biological systems like groups of neurons learn and remember [5, 6]. These theories hypothesized that learning took place by reinforcing neural connections corresponding to correct behavior. The first implementation of an ANN came in 1958 in the form of The Perceptron [7, 8]. The Perceptron was a single layer neural network modeling a two-class image recognition problem. A class is a unique category, like dog and cat or on and off. There is no limit on the number of classes a model can attempt to learn. Another single layer neural network design was ADALINE, created in 1960 [9]. While the ADALINE algorithm shares a similar architecture with the perceptron, the learning rule is different. Unfortunately, the single-layer perceptron was proven to not work in cases where classes in the data are not linearly separable [10]. This realization led to a decline in neural network research.

While a single layer network could only solve linearly separable problems, in 1969 it was also found that a network with multiple layers could solve problems that are non-linearly separable [10]. The perceptron algorithm was limited to a single layer network due to its activation function being a non-differentiable step function. It was soon shown that multiple ADALINE neurons could be stacked on top of each other, creating a MADALINE network [11]. Due to its multilayer structure, MADALINE is able to learn non-linearly separable functions. A MADALINE network was and is still used as an adaptive filter that removes echo from phone lines [12].

Further advances in ANNs came in the form of learning improvements. Not long after the success of ADALINE and MADALINE, it was suggested that the concept of error propagation could be applied to ANNs [13, 14]. Additionally, it was shown that error backpropogation applied with a differentiable non-linear activation function was an extremely powerful method to train ANNs [15]. Algorithms based on the backpropogation of error are now the most common method to train ANNs.

Currently, ANNs can solve many diverse problems. ANNs have shown promise in everyday problems such as handwritten zip code recognition [16], fingerprint identification [17], and image recognition [18]; as well as more complicated problems such as lung cancer classification based on MRI images [19], estimating surface soil moisture from high-resolution aerial images of cropland [20], and stock market predition [21].

1.3 Existing Methods for Automated Isotope Identification and Quantification

To quantify the total number of counts from a radioisotope in a given spectrum, either the constituent isotopes must first be identified before quantification or identification and quantification must happen simultaneously. There are many radioisotope identification methods available, but few perform well given a low-resolution gamma-ray spectrum of a mixture of radioisotopes. Common methods include library comparison algorithms, region of interest (ROI) algorithms, principle component analysis (PCA), and template matching.

Library comparison algorithms attempt to match photopeak energies found in a gamma-ray spectrum with those found in a library of known isotope decay energies. Drifts and uncertainties in detector calibration can lead to misidentifying photopeaks, leading to incorrect identifications [22]. To be automated, this method needs an algorithm to extract photopeak centroids from a spectrum. Photopeak extraction algorithms face difficulties when a large number of photopeaks overlap in a spectrum, such as when a mixture of radio-isotopes are measured with a low-resolution detector [23].

ROI algorithms search for elevated counts compared to background in a region where photopeaks are expected to be for different radioisotopes. ROI algorithms may operate

poorly when photopeaks of different radioisotopes overlap [22]. For this reason, large isotope libraries will preform poorly using this method. Similarly to the library comparison algorithm, calibration drift may shift photopeaks into different neighboring ROIs, leading to incorrect identification. The ROI method has been used to differentiate normally occuring radioactive material (NORM) from special nuclear material (SNM) using plastic scintillators [24].

PCA can also be applied to radioisotope identification. The goal of PCA is to reduce the dimensionality of a dataset into uncorrelated variables [25]. Using a few of these principle components, the data may be represented in a reduced space that contains most of the information present in the original data. The transformed data can then be clustered based on isotope identity. Clustering algorithms may include K-means or Mahalanobis distance [26, 27]. PCA has been applied to isotope identification using plastic scintillators [28] and anomaly detection using both plastic scintillators and NaI detectors [29]. Despite the progress of PCA in isotope identification, there has not been significant progress in applying PCA to separating mixtures of isotopes in gamma-ray spectra.

Template matching algorithms find an example in a database of gamma-ray spectra that most closely matches a measured spectrum. [22]. The database of spectra can contain multiple detector calibration settings, shielding materials, and source-to-detector distances. Goodness of fit can be measured using a chi-squared test, euclidean distance, or Mahalanobis distance. While a sufficient amount of example spectra can be used to identify almost any measured spectrum, the drawback of this method is the time necessary to compare a measured spectrum to the library and the computer memory necessary to store said library. This method also may have difficulty when mixtures of isotopes are considered, although work is being done to correct this [30].

1.4 Existing Neural Network Applications to Isotope Identification and Quantification

There have been a number of published papers which apply ANNs to automated isotope identification. ANNs have been applied to peak fitting [31], isotope identification [2, 3], and

activity estimation [2, 32]. Many of this work rely on ROI methods [33], feature extraction [34], high-resolution gamma-ray spectra as the input to the ANN [4], small libraries of isotopes, and assume perfectly calibrated detectors. ANN training methods created for high-resolution gamma-ray spectra may not perform well when trained using low-resolution spectra given the large discrepancy in resolution. ANN training that relies on ROI methods may not perform well when ROIs overlap significantly with large libraries of isotopes.

It has been shown that an ANN may be trained to perform isotope identification and quantification using low-resolution NaI gamma-ray spectrum using a library of five isotopes [35]. While promising, this study did not include complicated source mixture analysis. This study also used a library too small to be of practical use. The US Department of Homeland Security has 31 isotopes in their minimum identification standards for radioisotope identifiers (RIIDs) [36] and many more may be needed for post-detonation nuclear forensics [1].

1.5 Proposed Solution

The solution outlined in this thesis is to perform radioisotope identification and quantification simultaneously using an ANN. Skipping automated peak-fitting routines releases the algorithm from the burden of determining proper peak-fitting subroutines for the variety of cases in which a peak may be seen. A photopeak can typically be fit using a Gaussian distribution added to a linear baseline. This baseline changes based on its location in another photopeak's Compton continuum. In spectra where peaks overlap, deconvolution techniques may be needed to resolve constituent peaks. An unknown gain or poorly calibrated detector will shift photopeaks, further complicating a spectrum. Instead of making deterministic rules for each of these cases, a machine learning algorithm can be taught to recognize and handle them automatically.

Traditionally, once photopeaks are found, another algorithm is needed to measure the locations of peak centroids and additional information (peak area, area uncertainty) to identify what isotopes are present in a spectrum. This process adds computation time and again suffers from the need to be modified to handle changes in a spectrum as described

above. Also, the algorithm that performs identification based on peak information must be tailored to the peak-fitting routine. This requires unique algorithms to be created for different detector materials and sizes, as they change the shape of the spectrum. Using an ANN with an appropriate training set, these problems can be avoided.

By allowing a machine to learn the important features of a gamma-ray spectrum, the problem of determining the best analysis technique given a spectrum with significant features overlap and unknown calibration is avoided. This method is ideal for low-resolution gamma-ray spectroscopy for mixtures of radioisotopes for a number of reasons. Because this method uses simulated gamma-ray spectra to train the ANN, there is no restriction on the number of isotopes allowed in the library or their identity. This allows the ANN training set to be cheaply generated using mixtures of exotic, dangerous, or short-lived isotopes that are not easily accessible in an academic setting.

Another benefit of using a training set of simulated spectra is that an ANN may be trained for a specific scenario using a custom isotope library. The isotope library requirements for a border patrol, which may focus on distinguishing medical isotopes and NORM from SNM, are very different from the isotope library needed to perform post-detonation nuclear forensics. A simulated training set also allows the same ANN creation process to be applied to different detector materials. This is because different gamma-ray detector materials, such as CZT or HPGe, create spectra with different features. Because spectra can be generated with different photomultiplier tube (PMT) gain values, this method can be insensitive to a range of gain shift. This insensitivity would allow isotope identification and quantification to be performed without prior knowledge of the detectors calibration.

CHAPTER 2

THEORY

2.1 Neural Networks

2.1.1 Neural Network Architecture

An ANN is a mathematical model that attempts to map an arbitrary function from $\mathbb{R}^M$ to $\mathbb{R}^N$, where M and N are positive integers. An ANN accomplishes this by mimicking biological neurons. One example of an ANN architecture is shown in Figure 2.1. This ANN has N neurons in input layer A, J neurons in hidden layer B, and K neurons in output layer C. Each neuron in adjacent layers are connected by weights, represented in Figure 2.1 by arrows connecting neurons.

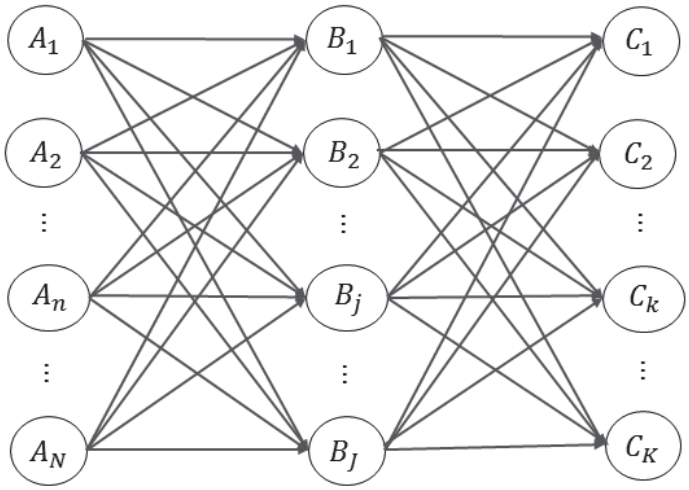

Figure 2.1: Example ANN with input neurons A_n, hidden neurons B_j, and output neurons C_k.

Similar to a biological neuron, the ANN neuron receives input stimuli, performs an oper-

ation using it, and outputs the resulting signal. The structure and equation governing the operation of an individual neuron is shown in Figure 2.2.

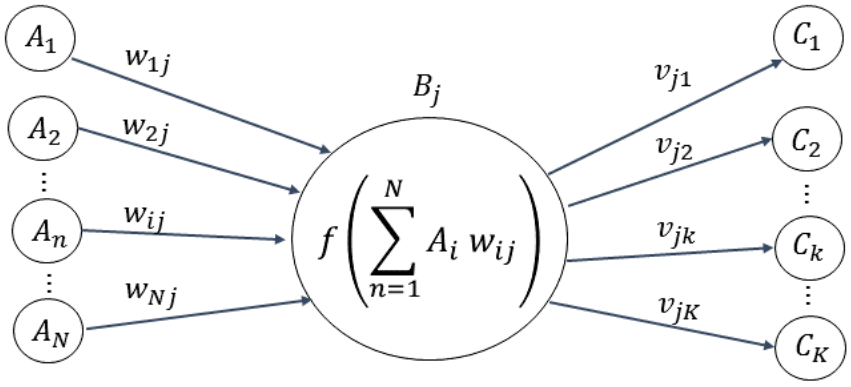

Figure 2.2: Summary of the operation of a single neuron.

As seen in Figure 2.2, each neuron operates by summing the products of the previous layers values (A_1, A_2, ... A_N) and each individual weight (w_{1j}, w_{2j}, ... w_{nj}) connecting the neurons. This summation is analogous to the stimulus a biologic neuron receives from its dendrites. The stimulus is then operated on by an activation function f, typically a non-linear function. The output signal is then passed to the next layer of the ANN where the process repeats. An ANN may be trained by setting the network weights, $\mathbf{W}$, in such a way that they minimize the error between target values, $\mathbf{T}$, in a training dataset, $\mathbf{Y}$, and the ANN output given that training dataset, $f(\mathbf{Y};\mathbf{W})$,

$$\underset{\mathbf{W}}{\operatorname{argmin}}\ \text{Error}(f(\mathbf{Y};\mathbf{W}),\mathbf{T}). \tag{2.1}$$

Except under simple cases, Equation 2.1 cannot be solved analytically. Numerical methods for solving this equation include gradient descent through the back-propagation of errors [14], genetic learning algorithms [37], and Newtons method [38].

2.1.2 Simple Neural Network Example

An example of a very simple one-layer ANN is shown in Figure 2.3. This ANN takes two inputs (x_1 and x_2) and performs the operation shown in Figure 2.2. The weights in the

hidden layer connecting the i^{th} input neuron to the j^{th} output neuron will be represented by w_{ij}. The bias is set to a constant value of 1. This allows the bias to be effectively trained by changing the weights connecting the bias to the next layer. Using the hyperbolic tangent function, the network outputs for each class y_1, y_2, and y_3 range from -1 to 1. Using these outputs, any input can be classified into a given class if the class' respective output neurons value is above zero, or not in a class if the output neurons value is below zero. The equation for the output of the j^{th} output is given in Equation 2.2, where x_i is the i^{th} input from the previous layer, and b_i is the value of the weight connecting the j^{th} output to the bias neuron.

$$y_j = \tanh(\sum_i x_i w_{ij} + b_j), \tag{2.2}$$

To more clearly understand the geometry of the network's operation, consider the dataset in Figure 2.4. This dataset is composed of three classes: red, green, and blue. The axes that define this dataset are the inputs to the single layer network in Figure 2.3, (x_1,x_2). If $\mathbf{W}$ is defined to be a vector with elements (w_{11}, w_{21}), a line can be defined perpendicular to $\mathbf{W}$ and shifted by $\frac{b_1}{||\mathbf{W}||}$ away from the origin in the direction of $\mathbf{W}$. Given appropriate values for w_{11}, w_{21}, and b_1, a line that separates the blue class from the non-blue class can be created. Any point on the -$\mathbf{W}$ side of the line will have $y_1 < 0$, allowing for classification. Similarly, a separating line for the red class using w_{12}, w_{22}, and b_2 and a separating line for the green class using w_{13}, w_{23}, and b_3 can be constructed.

The classes in this example dataset are linearly separable, meaning lines can be drawn completely separating each class. If the classes were not linearly separable, additional hidden layers would be necessary to compute the function. It has been shown that additional hidden layers allow the creation of arbitrary decision boundaries [39].

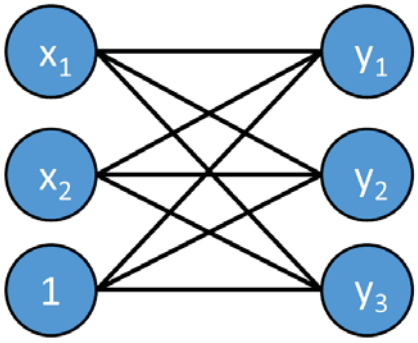

Figure 2.3: Example of a single-layer neural network with two inputs (x_1 and x_2), three classes (y_1, y_2, y_3), and a bias neuron set to one.

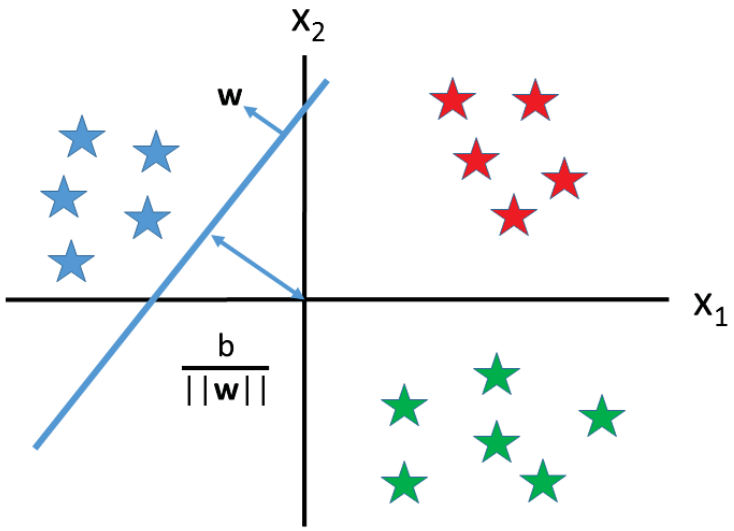

Figure 2.4: One possible dataset describing a three class function. Each class is represented by a different color.

2.1.3 Neural Network Training

One of the most common methods of training an ANN is through error backpropagation. Error backpropagation is a method to minimizes an error function with respect to the weights connecting neurons as seen in Equation 2.3.

Training the simple, one-layer ANN requires finding an expression for 2.3. For the following section, let y_i be defined as in Equation 2.2, E_{MSE} be defined as the mean squared error function, and training data be defined as in Equation 2.4 where $\mathbf{x_n}$ represents the n^{th} training

token and t_n represents the n^{th} binary training target [40]. Note, this derivation would need to be repeated for a different error function. By the chain rule,

$$\frac{dE_{MSE}}{dw_j} = \frac{dE_{MSE}}{dy_i}\frac{dy_i}{dw_j}. \tag{2.3}$$

$$D = (\boldsymbol{x}_1, t_1), ..., (\boldsymbol{x}_n, t_n) \tag{2.4}$$

Equation 2.3 can be solved by first evaluating $\frac{dE_{MSE}}{dy_i}$,

$$\frac{dE_{MSE}}{dy_i} = \frac{d}{dy_i}\sum_i (t_i - y_i)^2 \tag{2.5}$$

$$= \sum_i \frac{d}{dy_i}(t_i - y_i)^2 \tag{2.6}$$

$$= -2\sum_i (t_i - y_i). \tag{2.7}$$

Evaluating $\frac{dy_i}{dw_j}$,

$$\frac{dy_i}{dw_j} = \frac{d}{dw_j}\tanh(\boldsymbol{w}'\boldsymbol{x}_i + b) \tag{2.8}$$

$$= \tanh'(\boldsymbol{w}'\boldsymbol{x}_i + b)\frac{d}{dw_j}(\boldsymbol{w}'\boldsymbol{x}_i + b) \tag{2.9}$$

$$= \tanh'(\boldsymbol{w}'\boldsymbol{x}_i + b)x_{ij}. \tag{2.10}$$

where x_{ij} is the j^{th} feature of the i^{th} training vector. The update rule for the bias vector is found using

$$\frac{dE_{MSE}}{db_j} = \frac{dE_{MSE}}{dy_i}\frac{dy_i}{db_j}. \tag{2.11}$$

The expression for $\frac{dE_{MSE}}{dy_i}$ is known from Equation 2.7. Evaluating the other derivative in 2.11,

$$\frac{dy_i}{db_j} = \frac{d}{db_j}\tanh(\boldsymbol{w}'\boldsymbol{x}_i + b), \tag{2.12}$$

$$= \tanh'(\boldsymbol{w}'\boldsymbol{x}_i + b). \tag{2.13}$$

Finally, the gradient of the error function with respect to a single weight can be computed,

$$\frac{dE_{MSE}}{dw_j} = -2\sum_i (t_i - y_i)\tanh'(\boldsymbol{w}'\boldsymbol{x}_i + b)x_{ij}, \tag{2.14}$$

and a single bias,

$$\frac{dE_{MSE}}{db_j} = -2\sum_i (t_i - y_i)\tanh'(\boldsymbol{w}'\boldsymbol{x}_i + b). \tag{2.15}$$

The derivative in Equations 2.14 and 2.15 can used to update each weight and bias in the network as defined by

$$\Delta w_j = -\eta\frac{dE_{MSE}}{dw_j} \tag{2.16}$$

and

$$\Delta b_j = -\eta\frac{dE_{MSE}}{db_j} \tag{2.17}$$

where η in Equations 2.16 and 2.17 represents the learning rate of the neural network. The learning rate and its effect on training will be discussed more thoroughly in a later section.

Gradient descent can also be applied to multi-layer ANN. Given an L-layer network where the input stimuli to the l^{th} layer training example x is represented by

$$\boldsymbol{z}^{x,l} = \boldsymbol{w}^l\boldsymbol{a}^{x,l-1} + \boldsymbol{b}^l \tag{2.18}$$

where $\boldsymbol{w}^l$ is the l^{th} layer's weight matrix, $\boldsymbol{b}^l$ is the l^{th} layer's bias vector, and the output activation vector from the $(l-1)^{th}$ layer is

$$\boldsymbol{a}^{x,l-1} = f^{l-1}(\boldsymbol{z}^{x,l-1}) \tag{2.19}$$

where f^{l-1} represents the non-linear activation function used in the $(l-1)^{th}$-layer. Defining the output error as

$$\delta^{x,L} = \frac{\partial C}{\partial a^{x,L}} \odot \frac{\partial f^{l-1}(\boldsymbol{z}^{x,l-1})}{\partial \boldsymbol{z}^{x,l-1}}, \tag{2.20}$$

where $\odot$ is the Hadamard product, the output error can backpropagate to previous layers.

For each $l = L-1, L-2, ..., 2$ an error can be defined by

$$\delta^{x,l} = ((\boldsymbol{w}^{l+1})^T \delta^{l+1}) \odot \frac{\partial f^l(\boldsymbol{z}^{x,l})}{\partial \boldsymbol{z}^{x,l}}. \tag{2.21}$$

The gradient of the cost function as a function of each individual weight and bias can now be defined as

$$\frac{\partial E}{\partial w_{jk}} = a_k^{l-1} \delta_j^l \tag{2.22}$$

and

$$\frac{\partial E}{\partial b_j} = \delta_j^l. \tag{2.23}$$

Using Equations 2.16 and 2.17, the weights can be updated for a single iteration of backpropogation.

ANNs require some stopping condition to conclude training. One simple stopping condition is ending training when a threshold in network accuracy on a testing set is reached. Where this simple accuracy test is not applicable, such as for regression training sets, a condition based on the ANN training error metric can be used.

Early stopping has the benefit of preventing overfitting and encouraging generalization. Early stopping works by removing a small portion of the training data and defining it as testing data. The ANN is trained using the new training data while some error metric for both the training and testing set are recorded. As training progresses, the ANN likely will overfit to the training data, leading to an increase in error for the testing dataset. Early stopping ends training before overfitting occurs, as illustrated in Figure 2.5. Because training is stopped before the error in a dataset unknown to the model increases, generalization, or the ability for the ANN to correctly identify patterns outside of the training set, is also improved.

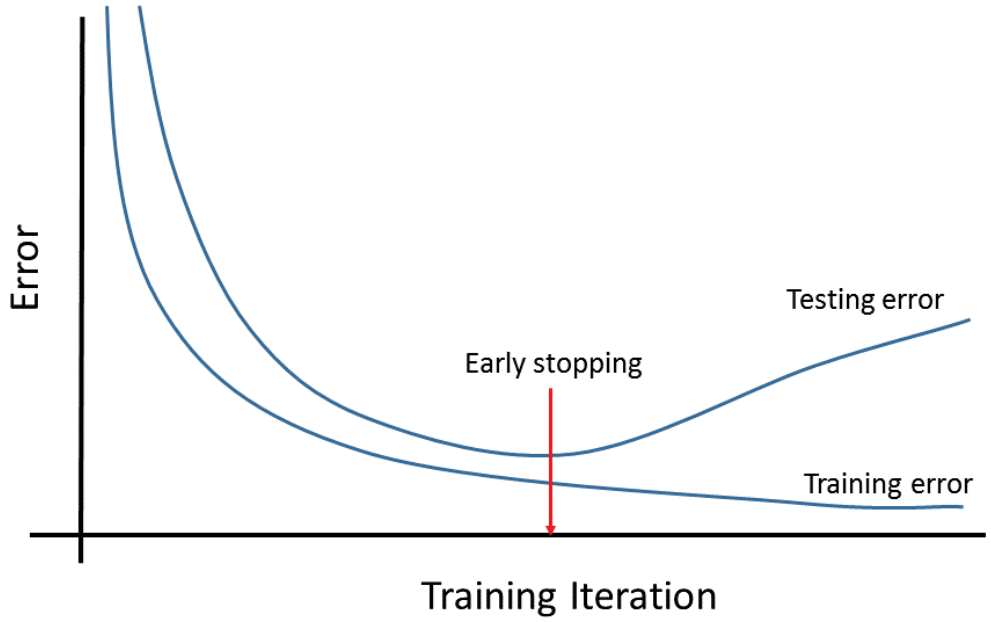

Figure 2.5: Ideal training and testing error curves.

2.1.4 Hyperparameters

In addition to the weights connecting neurons, ANNs can have additional hyperparameters. Hyperparameters determine both the networks structure (number of layers, number of nodes in each layer, activation function for each layer) and how the model learns (learning rate, momentum, loss function). In the following section, various hyperparameters and their effects on ANN learning are discussed.

2.1.4.1 Learning Rate

The learning rate is a tunable parameter that affects the magnitude of each weight update. If η is too small, the network will learn slowly and training will be inefficient. If η is too large, the network will fail to learn, either by converging to a non-extremum or by diverging. An example of a small and large learning rate are shown in Figure 2.6

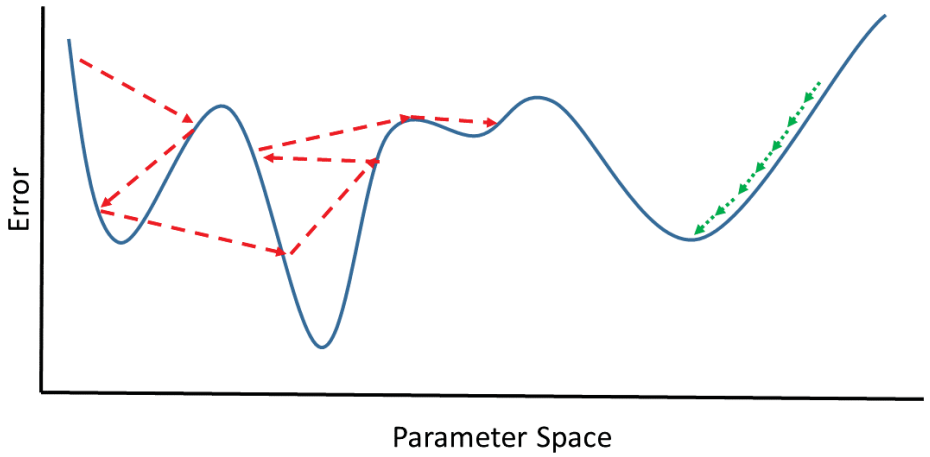

Figure 2.6: Example training paths for a large learning rate, red, and a small learning rate, green.

There are many methods to modify η to encourage more efficient learning. One method to increase the speed of learning is to start with a large η and decrease η as a function of iteration number. Ideally, this method would lead to quick initial learning when far from an optima and slower learning near an optima to more finely explore it. The difficulty with this method is the requirement for a function that slows the learning rate efficiently for a given problem.

2.1.4.2 Learning Momentum

Another method to speed up learning is to add a momentum hyperparameter, μ, to the weight update algorithm [41],

$$\Delta w_{ij}(n) = -\eta \frac{dE_{MSE}}{dw_j} + \mu \Delta w_{ij}(n-1). \tag{2.24}$$

Similar to the goal of slowing learning over time described above, the momentum term attempts to slow learning near optima. The momentum will be large when the weights are updated at large steps, far from an optima, but will decrease near an optima, allowing slower learning near an optimum.

2.1.4.3 Training Algorithms

There are many ANN training algorithms that employ clever learning rate schedules and momentum functions. These algorithms include but are not limited to: Nesterov's accelerated gradient [42], simulated annealing [43], ADADELTA [44], and ADAM [45].

The ADAM optimizer was chosen as the training algorithm for the work presented in this thesis due to its incorporation of parts of other successful optimization algorithms and its reported superior performance over these algorithms. Another benefit of the ADAM optimizer is introduction of only one hyperparameter, the learning rate.

The ADAM optimizer update rule is described below. For the following, g_t is the gradient of the error function with respect to the network parameters at iteration t,

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \tag{2.25}$$

is the estimate of the mean of the gradient at iteration t and

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{2.26}$$

is the estimate of the variance of the gradient at iteration t. For the following, the variables β_1 and β_2 are parameters called decay rates, ϵ is another parameter, and θ_t represents the network parameters at iteration t. As described by Kingma and Lei Ba, The default values for β_1, β_2, and ϵ are 0.9, 0.999, and 10^{-8} respectively [45]. These values were seen to work well for a variety of problems. While these hyperparameters can also be tuned, it has been shown that the default values work well for a variety of network architectures and datasets [45]. The bias-corrected first moment estimate is given by

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.27}$$

and the bias-corrected second moment estimate is given by

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}. \tag{2.28}$$

Finally, the weight update equation is computed as

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t. \tag{2.29}$$

2.1.4.4 Cost Function

The choice of cost function to train against depends on the targets the network is attempting to learn. One of the simplest error functions is the binary accuracy for classification,

$$E_{Binary} = \frac{1}{N}\sum_{n=1}^{N}[\hat{y}_n \neq y_n], \tag{2.30}$$

where N is the total number of output neurons, y_n is the ground truth of the nth output, and $\hat{y}_n$ is the ANN output of the nth output neuron. While this is a simple function, it penalizes the model for being close to the answer. Because there is no incremental indication that a model is improving, this error function is not typically used for gradient descent algorithms.

A simple differentiable cost function is the mean squared error (MSE) function shown in Equation 2.31. This function is differentiable, so gradient descent algorithms can be applied to it. The MSE function is appropriate when targets are any real number, as in a regression problem.

$$E_{Binary} = \frac{1}{N}\sum_{n=1}^{N}(\hat{y}_n - y_n)^2, \tag{2.31}$$

For classification problems, the average cross entropy, shown in Equation 2.32 can be used. Cross entropy measures how different the probability distributions y_n and $\hat{y}_n$ are from each other. Because the cross entropy treats y_n and $\hat{y}_n$ as probability distributions, they are required to exist in the range [0,1].

$$E = -\frac{1}{N}\sum_{n=1}^{N} y_n \log(\hat{y}_n) + (1 - y_n)\log(1 - \hat{y}_n), \tag{2.32}$$

Because the softmax function can be used for classification, it is traditionally used as the output function for the model when using the cross entropy cost function. The softmax

function is used in binary classification problems to calculate posterior class probabilities [46].

$$softmax(z_j) = \frac{\exp(z_j)}{\sum_{k=1}^{K} \exp(z_k)}. \tag{2.33}$$

2.1.4.5 Weight Regularization

Weight regularization is a hyperparameter that penalizes the ANN when the magnitude of the weights increases. Because the magnitude of the weights is tied to the complexity of the model, adding weight regularization attempts to limit complexity and the probability of overfitting.

A common method of incorporating weight regularization is by adding an L_n regularization term to the error function, as seen in Equation 2.34. Common values for n are 1 and 2. Adding weight regularization allows the magnitude of the weights to increase only when there is a comparable reduction in the unmodified error function.

$$\tilde{E} = E + \sum_{i} \lambda w_i^n. \tag{2.34}$$

In Equation 2.34, w_i is the weight between each neuron in the ANN and λ is the regularization strength hyperparameter. A larger λ will force the ANN to prefer smaller weights connecting the neurons. If the parameter λ is too small, the unbounded model complexity may fit only the training data. If the parameter λ is too large, the ANN will only minimize the L_n error, failing to learn.

2.1.4.6 Neuron Dropout

Another method to reduce model complexity is neuron dropout. Neuron dropout is the process of temporarily removing a neuron from the ANN architecture [47]. By randomly removing neurons from an ANN during training, heavy local codependency between neurons that could lead to the ANN becoming stuck in a local minimum in the error function, and

thus overtraining, is discouraged. The frequency with which neurons are removed is called the neuron dropout rate, which is a hyperparameter.

Almost always, taking the average output of more than one separately trained ANN improves the performance of the ANN [47]. By applying dropout at each neuron with the same probability throughout training, the ANN's architecture changes every iteration. The makes neuron dropout a cost efficient way to effectively average many different ANN architectures, improving performance.

2.1.4.7 Data Augmentation

Data augmentation changes the data during training using physically realistic transformations. For example, a training dataset of images can be rotated, flipped, blurred, or color augmented during training. An example of horizontal flip and a blur augmentation are seen in Figure 2.7. This cheaply expands the training dataset and discourages overtraining, as the ANN never observes the exact same image. Both the augmentation method and strength of said method are hyperparameters.

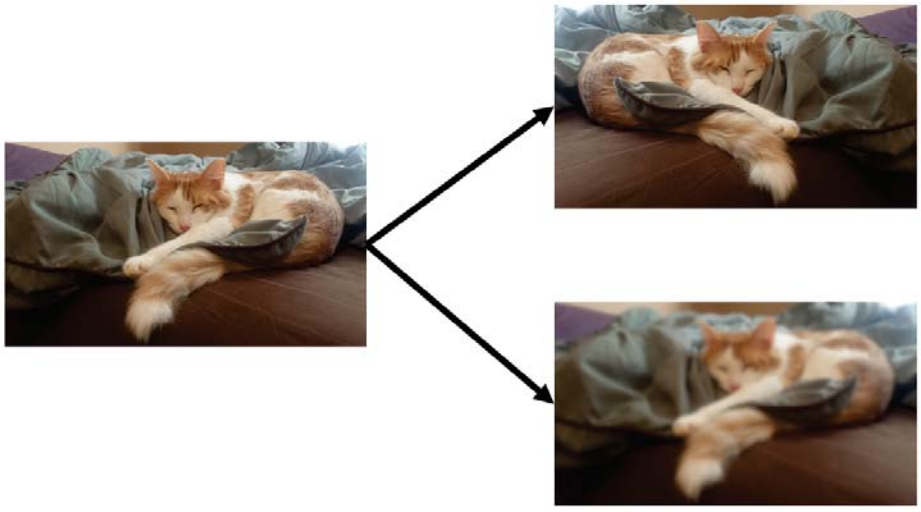

Figure 2.7: Two examples of data augmentation using an image of a cat. The image to the left is the original. The top right image is augmented using a horizontal flip. The bottom right image is augmented using blur.

2.1.5 Hyperparameter Optimization

In general, ANNs have many hyperparameters that require optimization. Optimizing these hyperparameters will lead to more efficient training and more accurate performance when training is concluded. There are several different methods to perform hyperparameter optimization for an ANN. These methods may include manual optimization, exhaustive grid search, and random parameter search.

Manually optimizing parameters is necessary when developing a novel algorithm. This involves changing hyperparameters and observing how the ANN trains and the final error on a validation dataset. Ideally, the ANN should train quickly and have a low error on a validation dataset. For many parameters 'rule of thumb' values exist that can be used to find parameters that work to some degree. Due to the large hyperparameter space, a manual search is cost prohibitive if further optimization is desired.

Once a range of parameters is determined through a manual search, multiple methods are available to explore the parameter space for an optimal solution. One method is an exhaustive grid search. In a grid search the parameter space is divided into a uniform grid and the joint performance of all parameters is tested. The grid search method is ineffective for two reasons. First, neural networks may have a large number of hyperparameters that need to be explored, and the computational requirement to explore the hyperparameter space increases exponentially with increasing hyperparamters. Second, in practice only a few hyperparameters dominate performance, but the dominating hyperparameters are different for different applications. A grid search may under represent the importance of key hyperparamters, as seen in Figure 2.8. While this method works, it has been shown that a random search in the hyperparameter target domain finds better hyperparameters quicker than testing equally distributed points in the chosen range [48]. It can also be shown that given 60 random samples over some space with a finite minimum, the minimum of those 60 random samples is within 5% of the true minimum with 95% probability [49]. This means that given a range of hyperparameters, the best performing hyperparameter combination out of 60 randomly sampled points is very likely to be close to optimal.

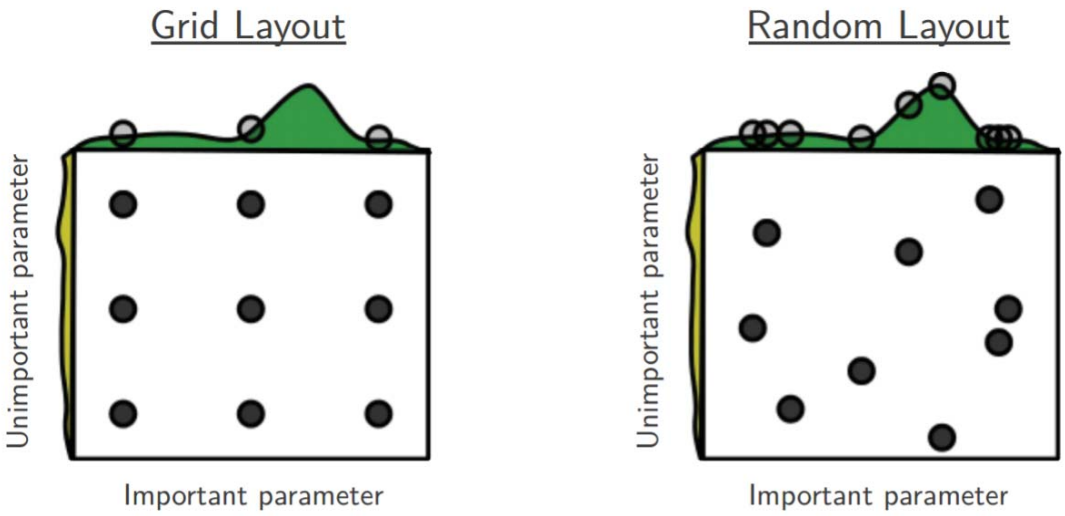

Figure 2.8: A comparison between a grid search and a random search for hyperparameter optimization when performance is strongly tied to one hyperparameter. The green function represents the effect of an important hyperparameter on a cost function while the yellow function represents the effect for an unimportant hyperparameter. Figure reproduced from [42].

The ability for an ANN to solve a problem depends on the network structure, teaching method, and the training set to be learned. In the following section, a method for generating a training set for isotope identification and quantification is described.

2.2 Training Set Creation

In order to train an ANN, a training dataset is required. For the specific problem of isotope identification and quantification, a few datasets may be useful. Given N isotopes in a desired library, one possible dataset is N smooth spectra with a training key which consists of an N-element vector of all zeros except one index which is set to one. This training key is also known as an N-element one-hot vector. Because it is simple, this dataset would fail to teach the ANN how a spectrum changes when isotopes are mixed or when count rate changes. A better solution is to create a dataset of spectra containing randomly mixed isotopes. Using an N-element training key where the n^{th} element represents the relative total gross-count contribution encourages the ANN to identify the photopeak, Compton

continuum, and possible pair production peaks from each isotope. This dataset would be incredibly difficult and expensive to create by physically measuring many isotope mixtures. A cheaper, simpler, and more modular method to creating the training set is to simulate it.

The gamma-ray spectra used to train the ANN were simulated by sampling spectra simulated in MCNP6 using a model of an Ortec 905-3 NaI detector [50]. The library of isotopes used are those identified by the US Department of Homeland Security as the minimum identification standards for RIIDs [36]:

^{241}Am, ^{133}Ba, ^{57}Co, ^{60}Co, ^{57}Cr, ^{137}Cs, ^{152}Eu, ^{67}Ga, ^{123}I, ^{125}I, ^{131}I, ^{111}In, ^{192}Ir, ^{40}K, ^{177m}Lu, ^{99}Mo, ^{237}Np, ^{103}Pd, ^{239}Pu, ^{240}Pu, ^{226}Ra, ^{75}Se, ^{153}Sm, ^{89}Sr, ^{99m}Tc, ^{232}Th, ^{201}Tl, ^{204}Tl, ^{233}U, ^{235}U, ^{238}U, and ^{133}Xe.

In addition to these, a background isotope was also included in the library. The background isotope was created by recording the background in the laboratory for 24 hours. The background was sampled using the technique described below.

Each spectrum contains between one and five isotopes from the above set. The range of isotopes mixed in each spectrum, the library of isotopes, and the detector simulated can be easily changed. This allows an ANN to be tailor made for a variety of specific scenarios.

The simulation process used in this work is based on work previously done in our research group [51]. The process begins with a converged MCNP simulation of an isotope's gamma-rays and branching ratios as found in Browne and Firestone [52]. An example of a converged spectrum is shown in Figure 2.9. This spectrum was generated based on a detector's PMT set at 760 V.

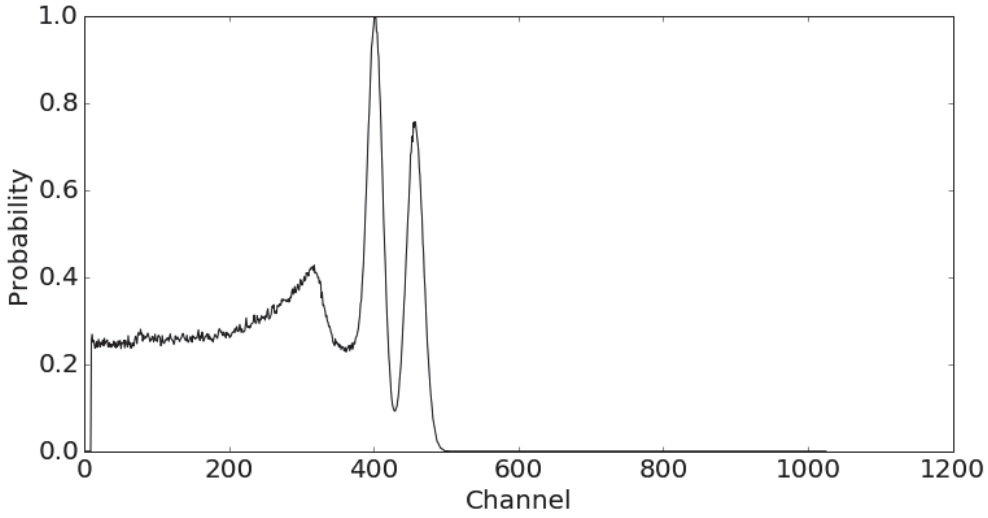

Figure 2.9: Normalized converged MCNP6 simulation of a ^{60}Co spectrum.

Treating each isotope's converged spectrum as a probability distribution function (PDF) and integrating over channel number, a cumulative distribution function (CDF) for each isotope can be created. This CDF can then be sampled using a uniform distribution with the inverse transform sampling method [53]. Using this method, spectra of any count number can be quickly generated. An example of the converged spectrum shown in Figure 2.9 sampled with 10^3 and 10^5 counts are shown in Figure 2.10. Using this method both low and high count scenarios can be simulated.

There are a few limitations to the spectrum simulation process described above. The main limitation is the lack of scattering geometry and point source without self-attenuation. This will result in simulated spectra that underestimate true Compton continua. Another limitation is that count-rate effects are not included. Count-rate effects include sum peaks and detector dead time. Sum peaks occur when two photons interact with the detector in a sufficiently small time window. When this occurs, the energy of the two photons are recorded as one. When the count rate increases the probability that two photons simultaneously interact in the detector, and thus the magnitude of the sum peak, increases. An increase in count rate also increases the detector's dead time. The detector's dead time is the period after a count is recorded that the detector cannot record additional pulses. For NaI(Tl) based detectors, the dead time is caused by limitations in the detector's electronics [54].

These effects cause changes in the spectrum that the ANN is not trained to recognize, which may lead to poor identification performance.

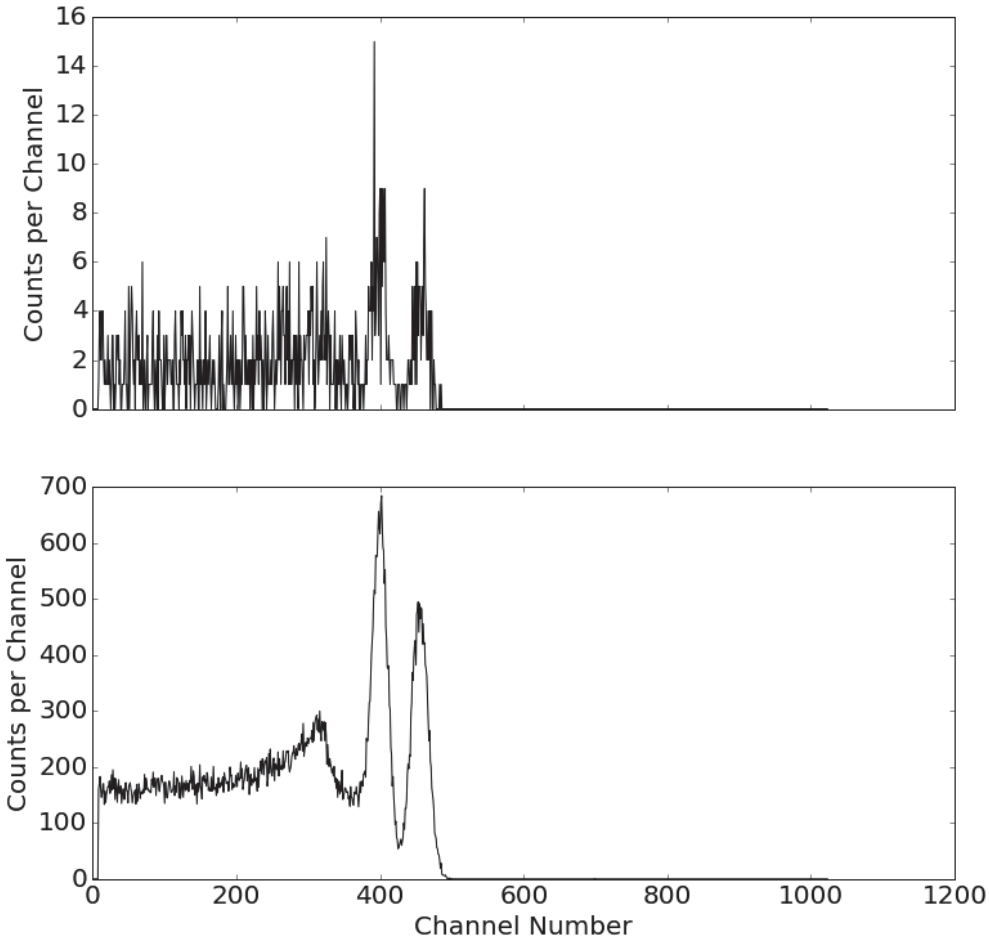

Figure 2.10: Two sampled ^{60}Co spectra using the converged ^{60}Co spectrum shown in Figure 2.9. The top spectrum was simulated with a total of 10^3 counts and the bottom spectrum was simulated with a total of 10^5 counts.

Spectra were simulated by randomly selecting the number of gross counts in each spectrum to range between 10^3 and 10^5. The total number of counts was randomly sampled between the chosen isotopes and a spectrum representing background. The background spectrum is seen in Figure 2.11. The simulated spectrum and the fraction of counts corresponding to each isotope and background are recorded. Currently, these target contributions include counts from simulated Compton scattering, so the ANN was taught to include contributions from gamma-ray photopeaks as well as the Compton continuum. To mimic gain shift due to calibration drift, a function that randomly linearly rebinned the position of each channel was

applied to each spectrum. The new spectrum was reconstructed using spline interpolation with the new bin positions. The magnitude of this shift was randomly chosen within the range (0.8,1.0). In addition to mimicking gain shift, randomizing the bins increases the generality of the neural network by inducing a form of data augmentation to the inputs. Traditionally, data augmentation methods are operated on before each training iteration. Because the ANN will see each example multiple times, this is not a true form of data augmentation.

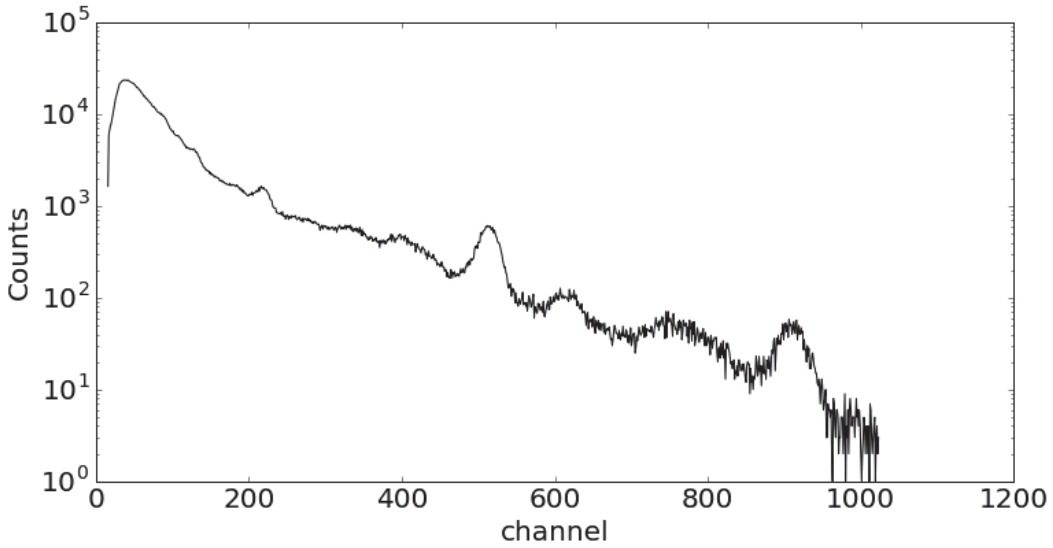

Figure 2.11: A 6 hour spectrum of natural background. This background spectrum was used to generate the background isotope used in the ANN training set.

Figures 2.12 and 2.13 show reconstructions of a ^{60}Co source using the spectrum simulation process described above. Note that with a calibration shift that moves each peak centroid about 75 channels, the rebinning function is still able to accurately reconstruct the photopeaks centroid and Compton edge location. The FWHM of the simulated peaks are slightly larger than those for the real spectra. This may be due to an overestimate of some source of error in the simulations, such as noise from the detector's electronics. Also note that the full Compton continuum is not well represented using this simulation process. This is because simulated spectra are based on a detector in vacuum. Lab conditions include scattering geometries not included in the simulations. A method to more accurately simulate spectra would include either mathematically changing the Compton region of each spectrum

based on Compton scattering physics or including a variety of scattering scenarios in the simulations. A greater variety of Compton scattering scenarios can be simulated quicker and cheaper using the former method.

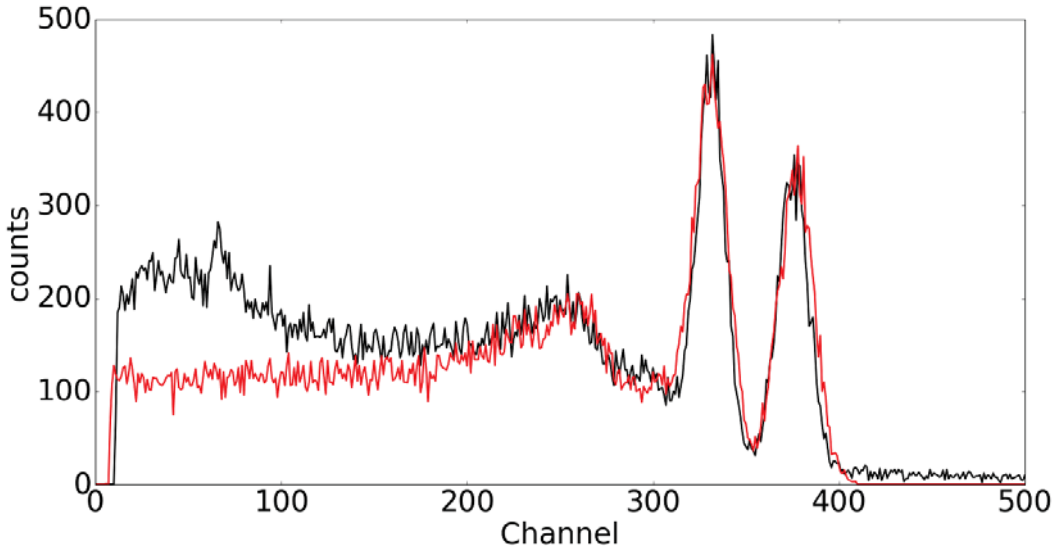

Figure 2.12: In black, a ^{60}Co spectrum recorded using a NaI(Tl) detector for 60 seconds with the detector PMT set at 740V. In red, a simulated ^{60}Co spectrum with 70000 total counts shifted by 0.825.

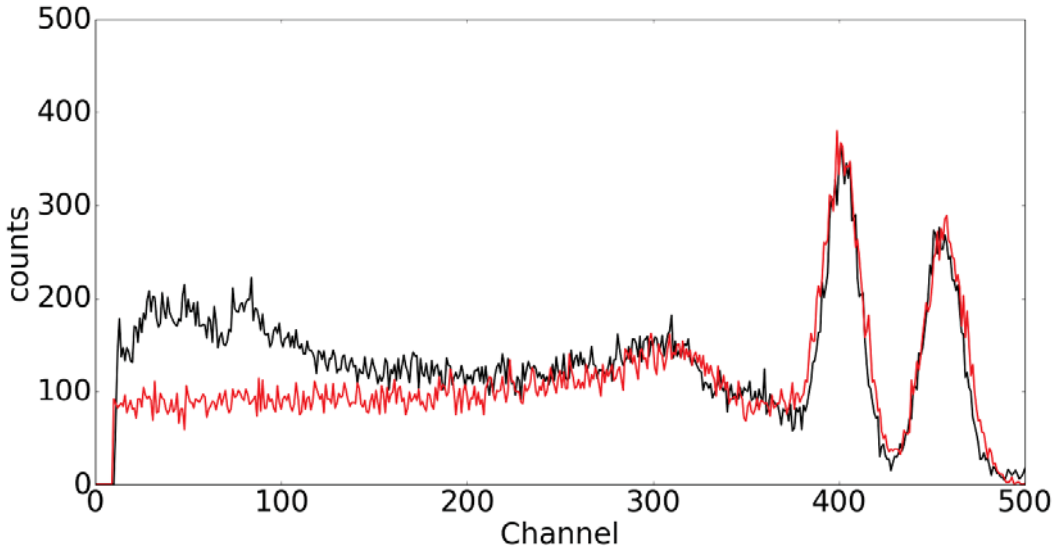

Figure 2.13: In black, a ^{60}Co spectrum recorded using a NaI(Tl) detector for 60 seconds with the detector PMT set at 760V. In red, a simulated ^{60}Co spectrum with 55000 total counts and no shifting.

In this chapter the ANN structure, training method, and a hyperparameter study are presented. Google's Python package, TensorFlow [55], was used to create and train the ANNs used in this paper. TensorFlow allowed for quicker prototyping and testing than would have been possible coding from scratch. TensorFlow also allowed more flexibility in constructing different neural network architectures than other python packages.

2.3 Chosen Network Architecture

A two-layer ANN was trained to perform relative gross count attribution from a library of 32 isotopes and background using the spectrum from a 1024 channel NaI detector as input. Deeper network architectures with additional layers are prone to long training times, overfitting issues, and learning peculiarities. Because of these issues, deeper architectures were not explored in this work.

The activation function for the hidden layers is the hyperbolic tangent function and the activation function for the output layer is the $softmax$ function, shown in Equation 2.33. While $softmax$ is traditionally used for classification, it was observed in this work that the it performed well as a regression model. Because the $softmax$ function restricts the range of output values to (0,1) and forces the output to sum to one, $softmax$ matches the conditions necessary to calculate relative class contributions.

The ANN also employs neuron dropout and L_2 regularization to combat overtraining. The choice for hyperparameter values chosen for the final ANN are discussed later.

2.4 Training Details

The ANN was trained using the ADAM optimizer. For the training, the ANN was given batches of 1024-channel spectra normalized by each spectrum's maximum value. The target vectors (33 x 1) containing the relative gross-count contribution from each isotope in the corresponding spectrum. The batch size was 1000 samples randomly picked from a training set of 10^5 simulated spectra. Randomly choosing training samples is a process called stochastic learning. Stochastic learning is usually faster than batch learning and often produces a

better ANN [56]. ANN weights were initialized with zero-mean Gaussian random variables which had a variance equal to the square root of the number of nodes in the previous layer [56]. Early stopping was used to terminate training. Training ends when the difference in mean cross entropy between the last 10 iterations and the cross entropy between the last 20 and last 10 iterations falls below 10^{-3}.

2.5 Random Hyperparameter Search Results

The ANN presented in this work has the following hyperparameters: number of neurons in layer 1, number of neurons in layer 2, initial learning rate for the ADAM optimizer, L_2 regularization strength, and neuron dropout rate. The ranges of parameters tested are shown in Table 2.1. The ranges tested were based on a manual search performed by the author. All ranges were sampled using a logarithmic distribution over their respective range except the dropout rate. The dropout rate was sampled using a uniform distribution over its range.

Out of 60 random hyperparameter combinations, the hyperparameter combination with the lowest error is shown in Table 2.1. The error during training for this ANN is shown in Figure 2.14.

Table 2.1: Range of hyperparameter values tested in a random search optimization

	Hyperparameter Range	Final Hyperparameter Values
Number of Neurons in Layer 1	10^2 - 10^4	1363
Number of Neurons in Layer 2	10^2 - 10^4	318
Initial Learning Rate	10^{-6}- 10^1	2.20 x 10^{-4}
L_2 regularization strength	10^{-5}- 10^{-2}	1.76 x 10^{-5}
Neuron Dropout Rate	0.0 - 1.0	0.655

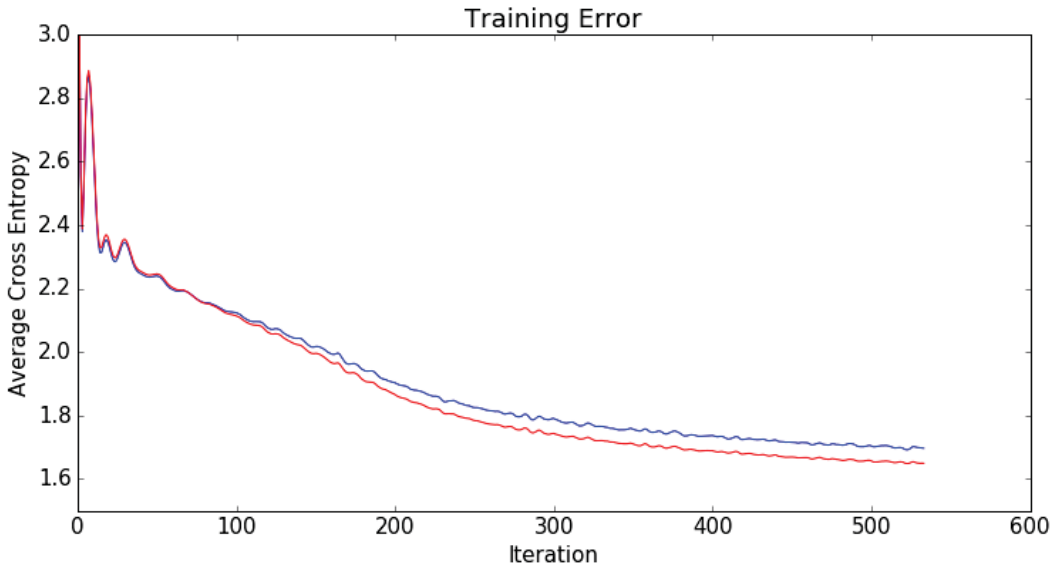

Figure 2.14: Training error curves using optimal hyperparameters found in the random search. The training dataset error is shown in blue and the testing dataset error is shown in red.

2.6 Summary

In this chapter, the structure of multi-layer ANNs, methods to train and optimize them, and a method to create a training set for isotope identification and quantification have been described. In the following section an ANN will be presented using these concepts and its performance on a number of real and simulated spectra will be discussed.

CHAPTER 3

RESULTS AND DISCUSSION

3.1 Identification Performance

The performance of this ANN is shown through a number of real spectra. All spectra in this section are real without background subtraction. For simplicity, only the first 5 largest relative gross-count contributions are shown. Because the ANN inputs are in the form of channel number and not bin energy, the spectra in this paper will be displayed as a function of channel.

Because the output of the ANN is the $softmax$ function, each output has a positive non-zero value. Because of this, the ANN calculates that each isotope is present in each spectrum. A minimum relative count contribution threshold can be used to determine what results can be ignored. The false alarm rate as a function of count contribution threshold for 1000 simulated gamma-ray spectra is shown in Figure 3.1. The 1000 spectra were simulated using the same method used to produce the training spectra. A false alarm is defined as the ANN finding a radio-isotope with a count contribution higher than the threshold that was not part of the isotopes simulated in said spectrum. Using Figure 3.1, we can determine a proper count contribution threshold given an acceptable false alarm rate. For example, given a maximum false alarm rate of 5%, all count contributions less than 0.22 can be ignored.

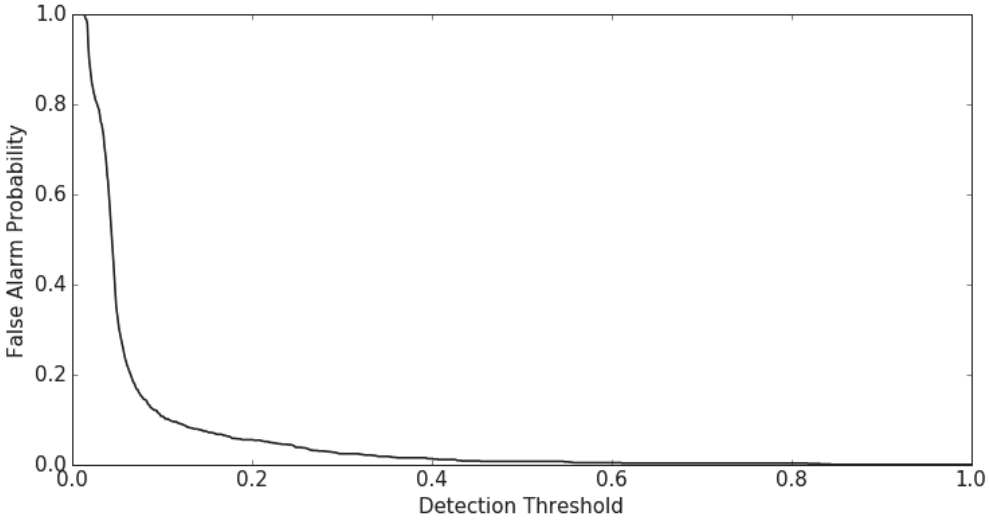

Figure 3.1: False alarm rate given a minimum relative count contribution.

3.1.1 Performance on Background

Figure 3.2 shows a 6 hour background spectrum and Table 3.1 shows the output from the ANN given the spectrum in 3.2. The ANN correctly identified a spectrum having no isotopes of interest as being predominantly background. The ANN ignores the clearly visible 1460 keV ^{40}K peak near channel 500. This demonstrated that the ANN correctly identified that the entire ^{40}K count contribution is from background and not an outside source.

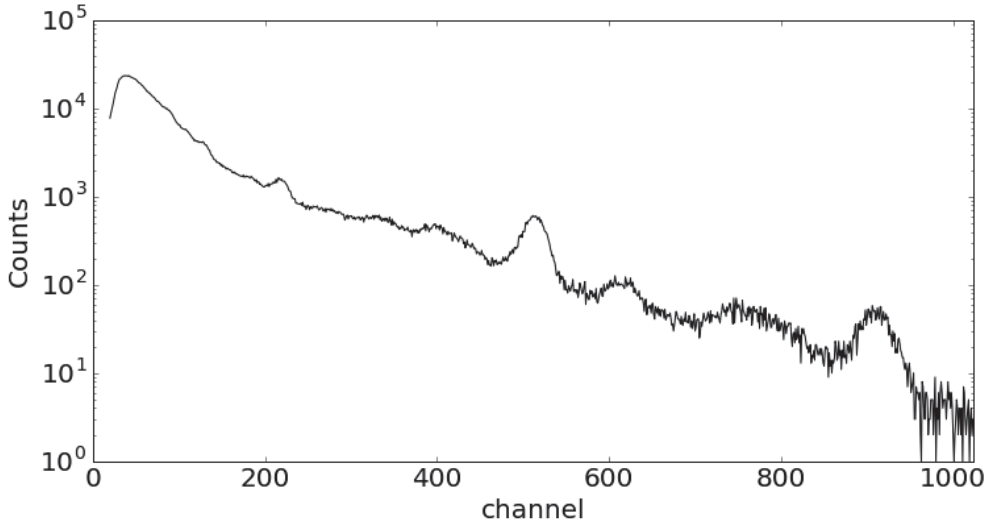

Figure 3.2: A 6 hour background spectrum. Total number of counts in this spectrum is 1742651.

Table 3.1: ANN output from Figure 3.2 representing relative gross count contribution

Isotope	Percent Contribution
Background	0.858
^{103}Pd	0.018
^{125}I	0.015
^{233}U	0.014
^{192}Ir	0.013

3.1.2 ^{60}Co

Figure 3.3 shows a ^{60}Co spectrum and Table 3.2 shows the output from the ANN given the spectrum in 3.3. The ^{60}Co source was placed at a distance that gave approximately the same count rate on the detector as background. The ANN correctly identified ^{60}Co and background as having nearly equal relative count contributions.

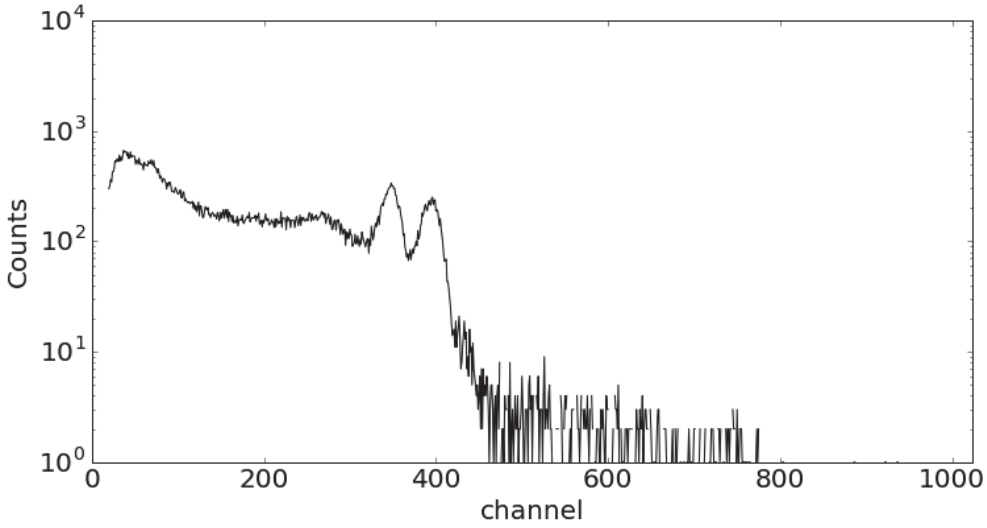

Figure 3.3: A 500 second gamma-ray spectrum of a 0.288 μCi ^{60}Co source measured from a distance of 7.5 cm from the detector face. There are 90003 gross counts in this spectrum.

Table 3.2: ANN output from Figure 3.3 representing relative gross-count contribution

Isotope	Percent Contribution
^{60}Co	0.436
Background	0.420
^{152}Eu	0.020
^{233}U	0.018
^{125}I	0.015

3.1.3 ^{152}Eu

The ANN also correctly identified the presence of a single isotope with a spectrum with a greater number of identifiable gamma-ray peaks than ^{60}Co. Figure 3.4 shows a ^{152}Eu spectrum with the output from the ANN shown in Table 3.3. Similar to the ^{60}Co example, the ^{152}Eu source had approximately the same count rate on the detector as background. While the ANN did not find a similar relative count contribution for ^{152}Eu and background, the ANN was able to correctly identify ^{152}Eu as the main non-background isotope. Because ^{152}Eu has more gamma-rays associated with it compared to ^{60}Co, more counts fall in its Compton continuum. Because the simulated Compton continuum does not match laboratory conditions, the ANN performs poorly here.

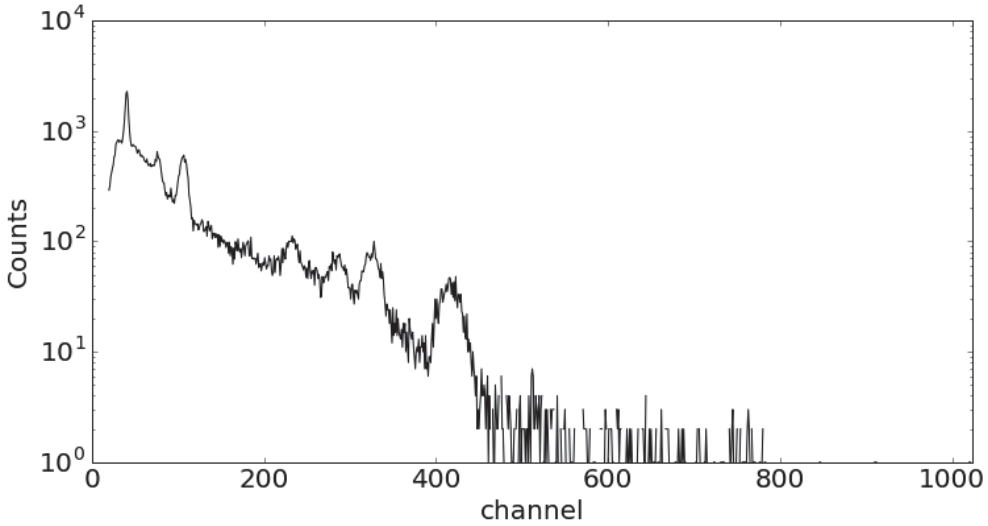

Figure 3.4: A 500 second gamma-ray spectrum of a 8.84 μCi ^{152}Eu source measured from a distance of 74 cm from the detector face. There are 76710 gross counts in this spectrum.

Table 3.3: ANN output from Figure 3.4 representing relative gross-count contribution

Isotope	Percent Contribution
Background	0.588
^{152}Eu	0.115
^{103}Pd	0.022
^{238}U	0.020
^{40}K	0.019

The above examples have clear photopeaks that can be easily identified by a feature extraction algorithm. Figure 3.5 shows a 10 second spectrum of the same ^{152}Eu source in the same position relative to the detector as Figure 3.4. This spectrum has photopeaks that are much more difficult to identify. Despite the drastic reduction in gross counts, the ANN was still able to correctly identify ^{152}Eu as the main non-background isotope present, as seen in Table 3.4. The ANN also finds a similar gross count contribution between ^{152}Eu and background in both the 500 and 10 second spectra. This shows the ANN operates similarly across a large range of gross counts.

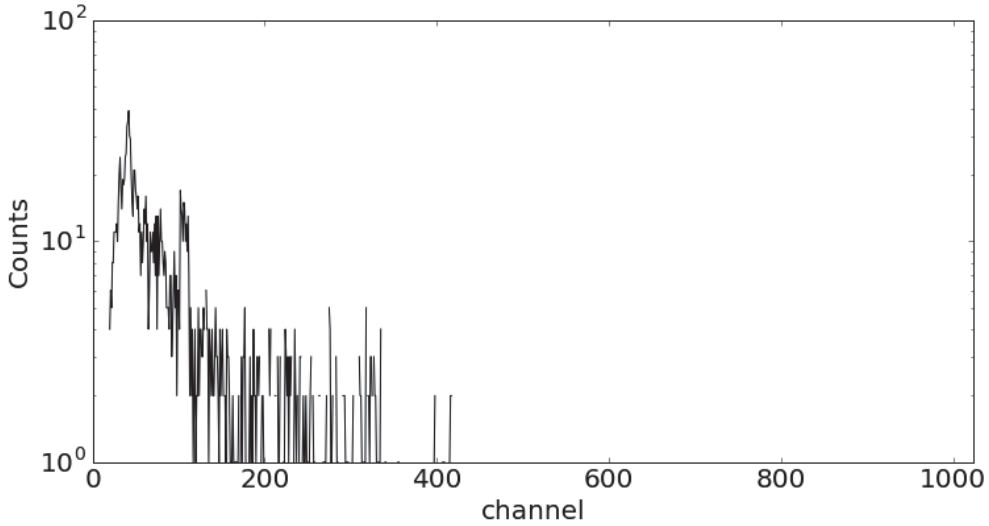

Figure 3.5: A 10 second gamma-ray spectrum of a 8.84 μCi ^{152}Eu source measured from a distance of 74 cm from the detector face. There are 1569 gross counts in this spectrum.

Table 3.4: ANN output from Figure 3.5 representing relative gross-count contribution

Isotope	Percent Contribution
Background	0.600
^{152}Eu	0.120
^{51}Cr	0.025
^{103}Pd	0.020
^{238}U	0.018

3.1.4 ^{60}Co and ^{137}Cs Mix

To investigate the ANNs ability to identify isotope mixtures, spectra with approximately equal gross count rates from ^{60}Co, ^{137}Cs, and background were tested. Table 3.5 shows the ANNs output from a ^{60}Co and ^{137}Cs source with approximately equal count rates on the detector. The spectrum for this combination is shown in Figure 3.6. The ANN is able to find both ^{60}Co and ^{137}Cs with approximately the same gross count contribution.

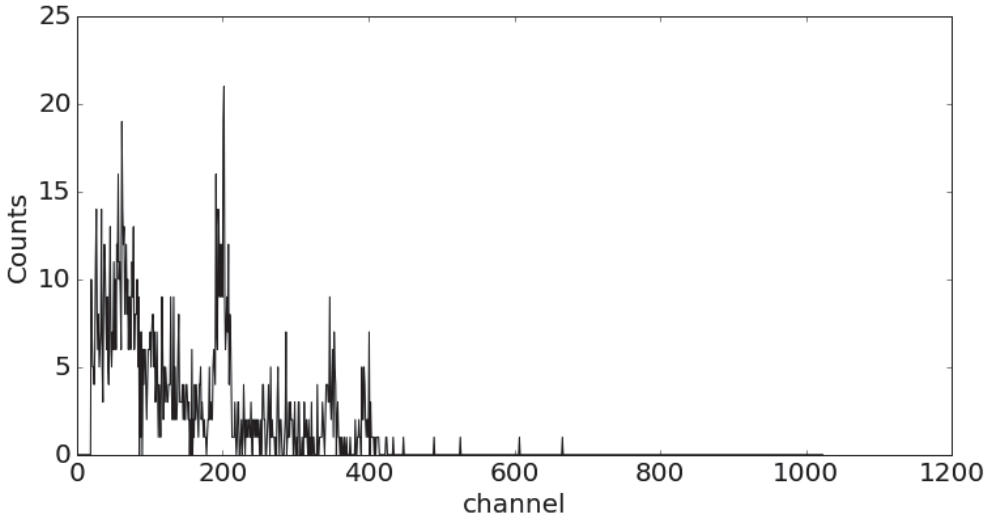

Figure 3.6: A 5 second gamma-ray spectrum of a 0.288 μCi ^{60}Co source measured from a distance of 7.5 cm from the detector face and a 0.890 μCi ^{137}Cs source. There are 1550 gross counts in this spectrum.

Table 3.5: ANN output from a 5 second gamma-ray spectrum of a 0.288 μCi ^{60}Co source measured from a distance of 7.5 cm from the detector face and a 0.890 μCi ^{137}Cs source.

Isotope	Percent Contribution
Background	0.461
^{137}Cs	0.172
^{60}Co	0.140
^{40}K	0.028
^{89}Sr	0.025

Table 3.6 shows the output from the ANN using the same isotope mix and configuration as above, but increasing the count time from 5 seconds to 60 seconds. The ANN output is similar for the 5 seconds and 60 seconds example, demonstrating the ANN can identify simple isotope mixtures through a large range of gross counts.

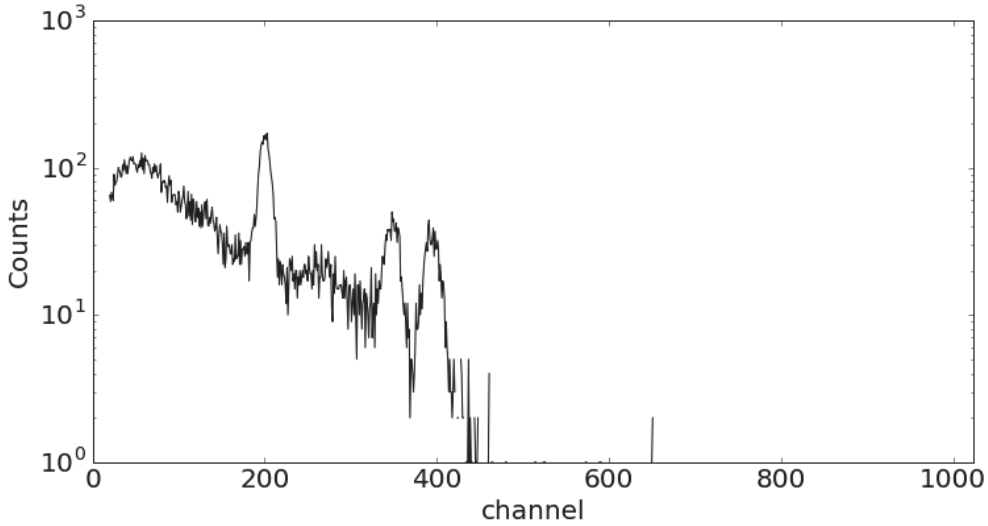

Figure 3.7: A 60 second gamma-ray spectrum of a 0.288 μCi ^{60}Co source measured from a distance of 7.5 cm from the detector face and a 0.890 μCi ^{137}Cs source. There are 18087 gross counts in this spectrum

Table 3.6: ANN output from a 60 second gamma-ray spectrum of a 0.288 μCi ^{60}Co source measured from a distance of 7.5 cm from the detector face and a ^{137}Cs source.

Isotope	Percent Contribution
Background	0.476
^{137}Cs	0.133
^{60}Co	0.114
^{152}Eu	0.042
^{125}I	0.034

3.1.5 ^{133}Ba and ^{60}Co Mix

The ANN also demonstrated promise in resolving mixtures of more complicated isotopes. Figure 3.8 shows a 60 second spectrum of both ^{133}Ba and ^{60}Co. The two isotopes were set at distances such that the relative count rate associated with each isotope was approximately the same. The ANN correctly identified both ^{133}Ba and ^{60}Co as the main isotopes present,

and correctly calculated they have approximately the same total count contribution to the detector.

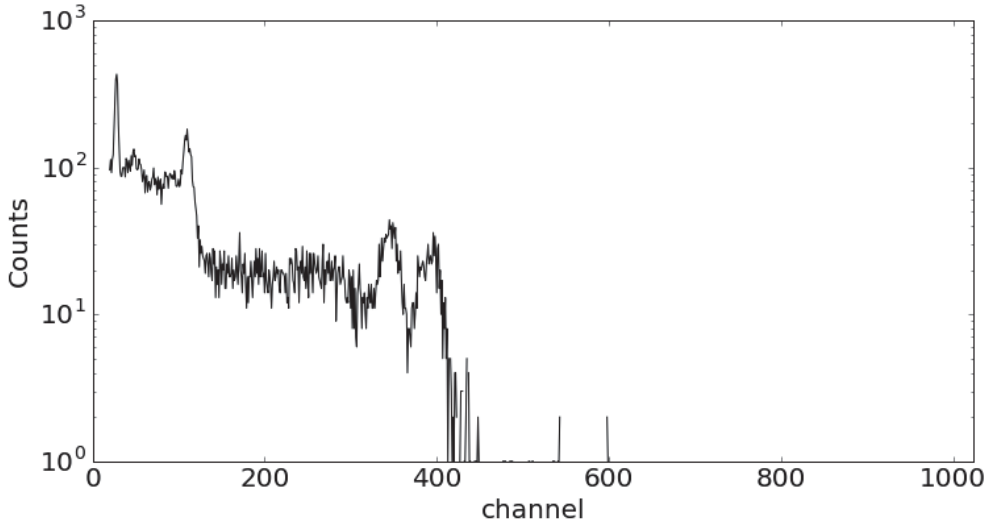

Figure 3.8: A 60 second gamma-ray spectrum of a 0.750 μCi ^{133}Ba source measured from a distance of 17 cm from the detector face and a 0.288 μCi ^{60}Co source measured from a distance of 7.5 cm from the detector face. There are 16787 gross counts in this spectrum.

Table 3.7: ANN output from Figure 3.8 representing relative gross-count contribution

Isotope	Percent Contribution
Background	0.325
^{60}Co	0.145
^{133}Ba	0.110
^{152}Eu	0.080
^{51}Cr	0.057

3.1.6 BeRP Ball Results

The ANN had difficulty identifying the BeRP ball. The BeRP ball is a 4.48 kg sphere of α-phase weapons grade plutonium in a 0.3 mm 304 stainless steel cladding. A 60 second

spectrum of the BeRP ball is shown in Figure 3.9. The ANN saw this spectrum as predominantly background as shown in Table 3.8. This may be due to the fact that each training example was guaranteed to have background, so in order to minimize error, the ANN attempts to give a large value for background regardless of the spectrum. This may also be due to the change in background radiation between this spectrum and the ANNs training set background. Because the BeRP ball was measured at the Device Assembly Facility in the Nevada National Nuclear Security Site, the background radiation was very different from the background found in Illinois.

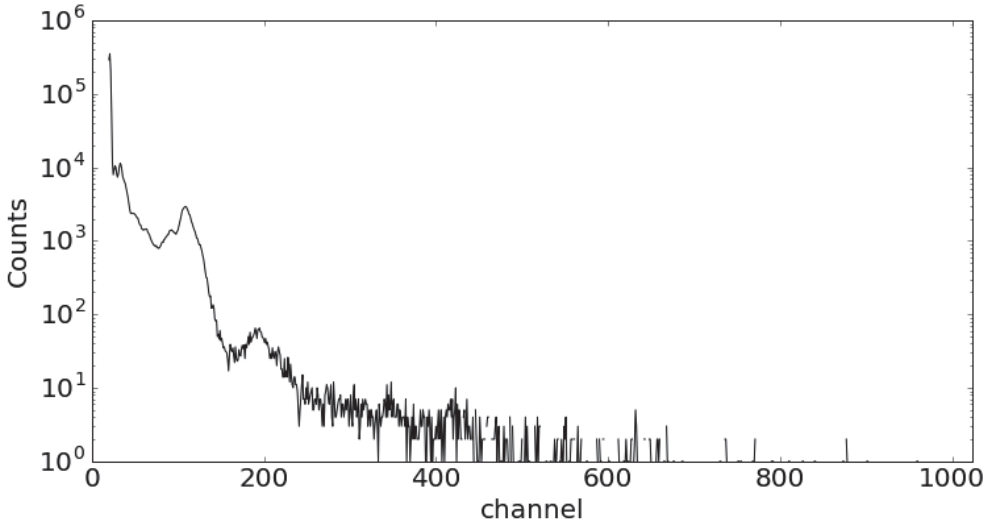

Figure 3.9: 60s spectrum of the BeRP ball.

Table 3.8: ANN output from the BeRP ball.

Isotope	Percent Contribution
Background	0.441
^{133}Xe	0.052
^{204}Tl	0.048
^{201}Tl	0.046
^{240}Pu	0.026

3.1.7 Algorithm Performance with Shielded Spectra

One method to hide a radioactive source is to place shielding material between the detector and source. Shielding material will attenuate lower energy gamma-rays stronger than higher energy gamma-rays, distorting the spectrum. Because the ANN was trained on unattenuated simulated spectra, only isotopes with gamma-ray energies close to each other will be recognized when attenuated by a shield.

Figure 3.10 shows three ^{60}Co spectra, one without shielding, one shielded by 2.27 mm of lead, and one shielded by by 5.33 mm of lead. As the amount of shielding material increases, the photopeaks are attenuated to a larger degree.

Table 3.9 shows the ANN's output for all three spectra in Figure 3.10. As expected, as more shielding material is added, the count contribution from ^{60}Co decreases. The decrease in the count contribution from ^{60}Co is mirrored by an increase in the count contribution from background. Note that the 5^{th} smallest isotope is the only one that changes with the addition of shielding. This shows that the ANN can identify a shielded isotope if the isotope emits energies in a similar range.

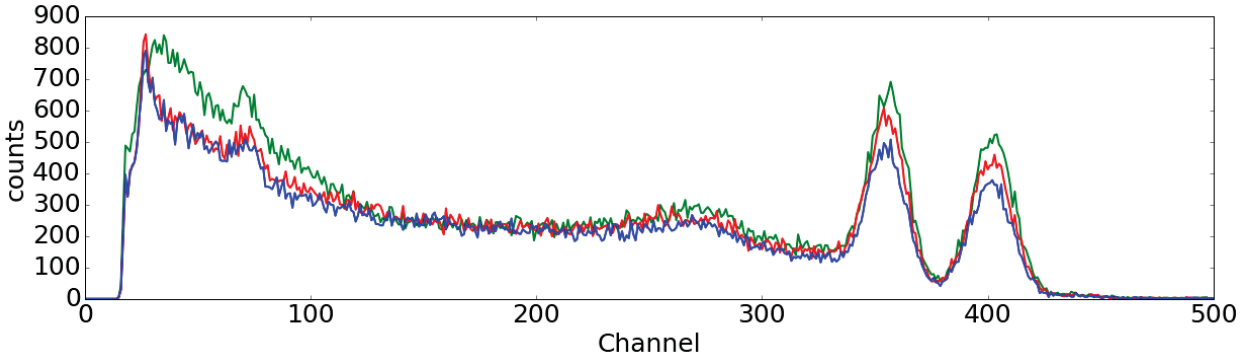

Figure 3.10: Three spectra of a ^{60}Co measured 10 cm from the detector face for 480 seconds. The green spectrum is a bare ^{60}Co source, the red spectrum is the ^{60}Co source attenuated by a 2.27 mm sheet of lead, and the blue spectrum is the ^{60}Co source attenuated by a 5.33 mm sheet of lead.

Table 3.9: ANN output from the three ^{60}Co spectra in Figure 3.10.

Bare Source	
Isotope	Percent Contribution
^{60}Co	0.5325
Back	0.3257
^{40}K	0.0290
^{238}U	0.0160
^{103}Pd	0.0130

2.27 mm Lead	
Isotope	Percent Contribution
^{60}Co	0.5033
Back	0.3437
^{40}K	0.0248
^{238}U	0.0172
^{192}Ir	0.0144

5.33 mm Lead	
Isotope	Percent Contribution
^{60}Co	0.4363
Back	0.3981
^{40}K	0.0233
^{238}U	0.0179
^{89}Sr	0.0169

Figure 3.11 shows three ^{152}Eu spectra, one without shielding, one shielded by 2.27 mm of lead, and one shielded by by 5.33 mm of lead. Note that the peaks in the green spectrum at lower energies disappear when an attenuator is introduced, while the higher energy peaks stay relatively the same.

Table 3.10 shows the output of the ANN for each spectra. Without shielding, the ANN correctly identifies the ^{152}Eu as being the main contributor to the counts in the spectrum. With 2.27 mm and 5.33 mm of lead, the ANN misidentifies the spectrum as being mainly background. Because there are still visible peaks in the attenuated spectra, an algorithm based on photopeak feature extraction may correctly identify the shielded ^{152}Eu spectra. Because the ANN is taught to recognize bare sources, it cannot identify attenuated sources when their photopeak energies are very far apart.

An increase in lead shielding for the ^{152}Eu source created a spectrum that does not match any isotope in the ANN training library. The ANN classified the unknown pattern as mostly background. This is evidence that the ANN prefers to classify unknown patterns as background rather than incorrect isotopes. An ANN may be able to be trained to recognize shielding in cases where shielding is expected. For post-detonation debris analysis, shielding is expected to not be an issue.

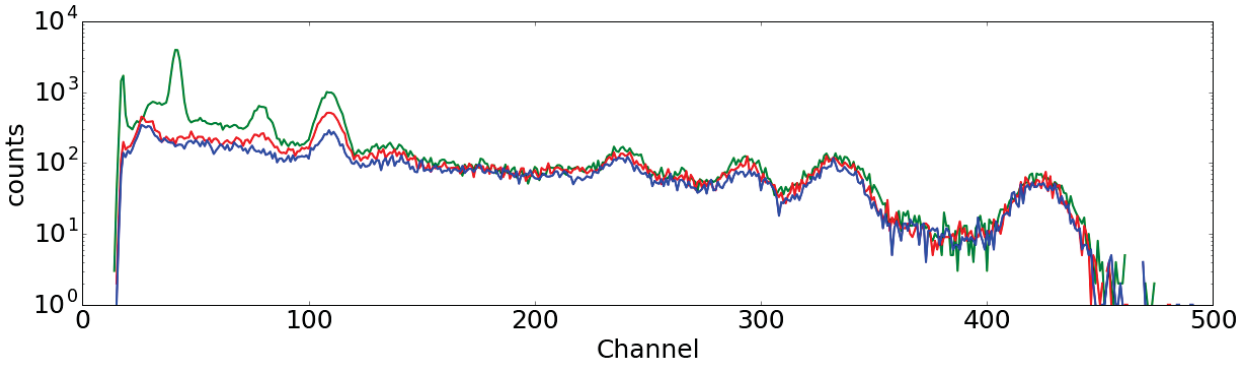

Figure 3.11: Three spectra of a ^{152}Eu measured 15 cm from the detector face for 60 seconds. The green spectrum is a bare ^{152}Eu source, the red spectrum is the ^{152}Eu source attenuated by a 2.27 mm sheet of lead, and the blue spectrum is the ^{152}Eu source attenuated by a 5.33 mm sheet of lead.

Table 3.10: ANN output from the three ^{152}Eu spectra in Figure 3.11.

Bare Source	
Isotope	Percent Contribution
^{152}Eu	0.3740
Back	0.2483
^{131}I	0.0594
^{51}Cr	0.0404
^{99}Mo	0.0398

2.27 mm Lead	
Isotope	Percent Contribution
Back	0.4038
^{60}Co	0.0806
^{131}I	0.0798
^{152}Eu	0.0738
^{40}K	0.0737

5.33 mm Lead	
Isotope	Percent Contribution
Back	0.4705
^{60}Co	0.0926
^{40}K	0.0722
^{152}Eu	0.0661
^{89}Sr	0.0583

3.2 Measurement of sensitivity to gain shift

The ANN also demonstrated some degree of gain invariance. Sixty second spectra of ^{60}Co were taken within a range of biases between 730 V to 765 V in steps of 5 V applied to the PMT. The photopeak channels shifted by an average of 19.4 channels, or 1.89% between each 5 V change. Figure 3.12 shows the difference in detector response at the largest and smallest voltage tested. Table 3.11 shows the 10 largest ANN outputs from the ^{60}Co source taken in a range of applied voltages. In the range between 740 V to 755 V the ANN finds

very similar contributions between ^{60}Co and background. At 730 V, 735 V and 760 V this ratio starts to change significantly, but ^{60}Co and background are still found by the ANN to be the main count contributors. At 765 V, ^{60}Co and background are found as the highest count contributors below ^{40}K. This shows that an ANN can be taught to be gain invariant within a range of applied PMT voltages.

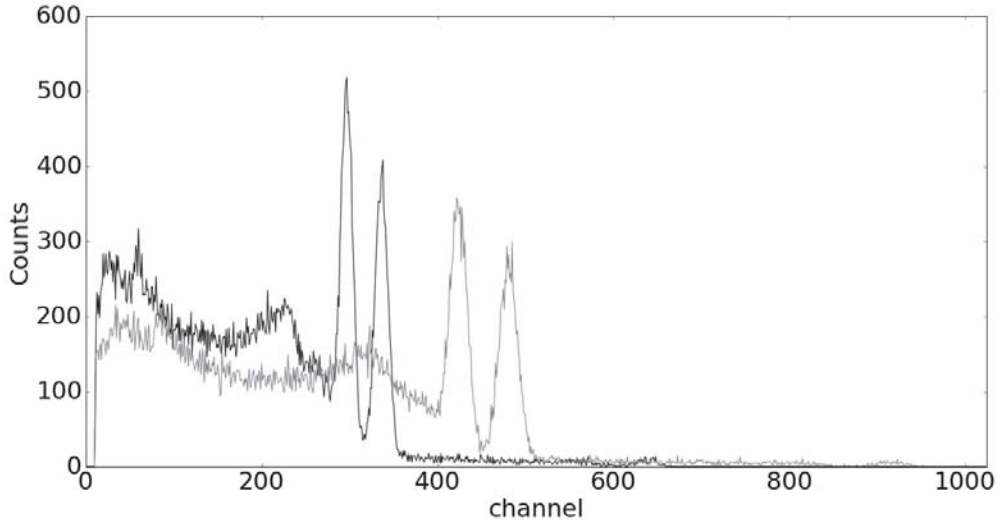

Figure 3.12: Two 60 second ^{60}Co spectra. The PMT was biased at 730 V for the black spectrum and 765 V for the grey spectrum.

Table 3.11: Summary of the top 4 ANN outputs from a ^{60}Co source using different PMT voltage settings.

730 V	
Isotope	Percent Contribution
Background	0.3349
^{60}Co	0.2974
^{89}Sr	0.1498
^{137}Cs	0.0247

735 V	
Isotope	Percent Contribution
^{60}Co	0.4667
Background	0.2746
^{89}Sr	0.0841
^{241}Am	0.0191

740 V	
Isotope	Percent Contribution
^{60}Co	0.8393
Background	0.0939
^{137}Cs	0.0129
^{40}K	0.0086

745 V	
Isotope	Percent Contribution
^{60}Co	0.8313
Background	0.1042
^{40}K	0.0096
^{137}Cs	0.0088

750 V	
Isotope	Percent Contribution
^{60}Co	0.8247
Background	0.1070
^{40}K	0.0129
^{241}Am	0.0078

755 V	
Isotope	Percent Contribution
^{60}Co	0.8388
Background	0.1002
^{40}K	0.0101
^{241}Am	0.0069

760 V	
Isotope	Percent Contribution
^{60}Co	0.7105
Background	0.1472
^{40}K	0.0574
^{241}Am	0.0134

765 V	
Isotope	Percent Contribution
^{40}K	0.5814
^{60}Co	0.1773
Background	0.1467
^{103}Pd	0.0156

CHAPTER 4

CONCLUSIONS AND FUTURE WORK

A method to perform isotope identification and quantification based on NaI gamma-ray spectra using a ANN has been described. This method was shown to be promising for resolving simple isotope mixtures. The ANN presented was also able to operate in a large range of detector calibration setting. Despite the presented advantages, there are many ways in which this work can be improved.

A more thorough study of additional ANN architectures may increase the effectiveness of this method. A more in depth study of deeper architecture, different hidden layer activation functions, and different architectures may increase performance. Deeper ANN architectures increase the capacity of the ANN to model functions with the downside of being difficult to train. A more rigorous hyperparameter study may also improve identification by finding a better combination of hyperparameters.

The ability for an ANN to identify larger isotope libraries is also important. For applications such as post-detonation nuclear forensics, libraries much larger than 32 isotopes are needed. Analyzing the capacity for different ANN architectures to identify larger isotope libraries would aid in determining if ANNs are a candidate for post-detonation nuclear forensics.

In this work it has been shown that an ANN may be trained to reliably perform isotope quantification in a mixture of isotopes with some level of detector gain invariance. It would be interesting to see how far the gain invariance can be stressed given some error. This has the potential to create gain invariant isotope identification algorithms. Creating an algorithm for gain invariant algorithms removes a large hurtle from automated isotope detection.

While the ANN successfully learned to identify real gamma-ray spectra using simulated training data based on a MCNP simulation, the MCNP simulation can be improved. The

MCNP model poorly predicts realistic Compton continua due to a lack of scattering materials present in the simulation. This may confuse the ANN, as it attempts to fit counts from both the photopeak and the Compton continuum. A better way to do this is to vary the continuum and train the ANN to predict only photopeak counts. Doing this will effectively teach the ANN to recognize and ignore various Compton continua. This may be achieved by creating several MCNP models with different scatter geometries. This may cause the ANN to overtrain on a small number of scattering geometries. A better method is to create a Compton continuum perturbation function that generates random continua. This method would not be vulnerable to overtraining and explores more scattering scenarios more efficiently than creating several different MCNP models.

This method is flexible enough to be extended to a variety of detection scenarios through the training dataset simulation. This includes different radiation detectors. Detector material such as CZT and HPGe would both benefit from quick isotope identification and quantification algorithms. Observing how well the ANN structure translates to different materials would be helpful.

An analysis of the error in network predictions would assist in determining if the ANN method is feasible for the problem of isotope identification and quantification using low-resolution detectors. It may be helpful to predict the error as a function of signal to background ratio and count rate. Future work will attempt to analyze the ANNs confidence with its predictions.

REFERENCES

[1] K. J. Moody, P. M. Grant, and I. D. Hutcheon, *Nuclear Forensic Analysis*. CRC Press, 2014.

[2] R. Abdel-Aal and M. Al-Haddad, "Determination of radioisotopes in gamma-ray spectroscopy using abductive machine learning," *Nuclear Instruments and Methods in Physics Research A*, vol. 391, pp. 275–288, 1996.

[3] M. Medhat, "Artificial intelligence methods applied for quantitative analysis of natural radioactive sources," *Annals of Nuclear Energy*, vol. 45, pp. 73 – 79, 2012.

[4] E. Yoshida, K. Shizuma, S. Endo, and T. Oka, "Application of neural networks for the analysis of gamma-ray spectra measured with a Ge spectrometer," *Nuclear Instruments and Methods in Physics Research A*, vol. 484, pp. 557–563, 2002.

[5] W. S. Mcculloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin Of Mathematical Biophysics*, vol. 5, pp. 115–133, 1943.

[6] D. O. Hebb, *The Organization of Behavior*. John Wiley Sons, Inc., 1949.

[7] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[8] F. Rosenblatt, *Principles of Neurodynamics*. Spartan Books, 1962.

[9] B. Widrow, "An Adaptive "ADALINE" Neuron Using Chemical "Memistors"," Solid-State Electronics Laboratory, Stanford Electronics Laboratories, Stanford University, Stanford, California, Tech. Rep., 1960.

[10] M. Minsky and S. A. Papert, *Perceptrons*. The MIT Press, 1969.

[11] W. C. Ridgway, "An adaptive logic system with generalizing properties," Ph.D. dissertation, Stanford University, 1962.

[12] B. Widrow and R. Winter, "Neural nets for adaptive filtering and adaptive pattern recognition," *Computer*, vol. 21, no. 3, pp. 25–39, Mar. 1988.

[13] P. J. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," Ph.D. dissertation, Harvard University, 1974.

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

[15] F. Murtagh, "Multilayer perceptrons for classification and regression," *Neurocomputing*, vol. 2, no. 56, pp. 183 – 197, 1991.

[16] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, Dec. 1989.

[17] S. Jeyanthia, N. Maheswaria, and R. Venkatesh, "Neural network based automatic fingerprint recognition system for overlapped latent images," *Journal of Intelligent Fuzzy Systems*, vol. 28, 2015.

[18] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Annual Conference on Neural Information Processing Systems*, pp. 1097–1105, 2012.

[19] I. J. Selvakumari Jeya and S. N. Deepa, "Lung cancer classification employing proposed real coded genetic algorithm based radial basis function neural network classifier." *Computational Mathematical Methods in Medicine*, pp. 1 – 15, 2016.

[20] L. Hassan-Esfahani, A. Torres-Rua, A. Jensen, and M. McKee, "Assessment of surface soil moisture using high-resolution multi-spectral imagery and artificial neural networks." *Remote Sensing*, vol. 7, no. 3, pp. 2627 – 2646, 2015.

[21] A. Rababaah and S. D., "Integration of two different signal processing techniques with artificial neural network for stock market forecasting," *Academy of Information Management Sciences Journal*, vol. 18, pp. 63–80, 2015.

[22] T. Burr and M. Hamada, "Radio-isotope algorithms for NaI gamma spectra," *Algorithms*, vol. 2, pp. 339–360, March 2009.

[23] H. Xiong, "Automated wavelet analysis of low resolution gamma-ray spectra and peak area uncertainty," Master's thesis, University of Illinois at Urbana-Champaign, 2015.

[24] J. Ely, R. Kouzes, J. Schweppe, E. Siciliano, D. Strachan, and D. Weier, "The use of energy windowing to discriminate SNM from NORM in radiation portal monitors," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 560, no. 2, pp. 373 – 387, 2006.

[25] J. I. T., *Principal Component Analysis*. Springer New York, 2002.

[26] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: analysis and implementation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 881–892, Jul 2002.

[27] V. V. Kumari, B. R. Raju, and A. Naik, "Hybrid clustering algorithm based on mahalanobis distance and mst," *International Journal of Applied Information Systems*, vol. 3, no. 5, pp. 60–63, Jul 2012.

[28] D. Boardman, M. Reinhard, and A. Flynn, "Principal component analysis of gamma-ray spectra for radiation portal monitors," *IEEE Transactions on Nuclear Science*, vol. 59, no. 1, pp. 154–160, Feb 2012.

[29] R. Runkle, "Analysis of spectroscopic radiation portal monitor data using principal components analysis," *IEEE Transactions on Nuclear Science*, vol. 53, no. 3, pp. 1418–1423, 2006.

[30] J. Mattingly and D. Mitchell, "A framework for the solution of inverse radiation transport problems," *IEEE Transactions on Nuclear Science*, vol. 57, no. 6, pp. 3734–3743, 2010.

[31] R. Abdel-Aal, "Comparison of algorithmic and machine learning approaches for the automatic fitting of gaussian peaks," *Neural Computing and Applications*, vol. 11, no. 1, pp. 17–29, 2002.

[32] V. Vigneron, J. Morel, M. Lepy, and J. Martinez, "Statistical modelling of neural networks in y-spectrometry," *Nuclear Instruments and Methods in Physics Research A*, vol. 396, p. 642647, 1996.

[33] V. Pilato, F. Tola, J. Martinex, and M. Huver, "Application of neural networks to quantitative spectrometry analysis," *Nuclear Instruments and Methods in Physics Research A*, vol. 422, pp. 423–427, 1999.

[34] L. Chen and Y.-X. Wei, "Nuclide identification algorithm based on KL transform and neural networks," *Nuclear Instruments and Methods in Physics Research A*, vol. 598, pp. 450–453, 2009.

[35] P. Olmos, J. C. Diaz, J. M. Perez, P. Gomez, V. Rodellar, P. Aguayo, A. Bru, G. Garcia-Belmonte, and J. L. Pablos, "A new approach to automatic radiation spectrum analysis," *IEEE Transactions on Nuclear Science*, vol. 38, pp. 971–975, August 1991.

[36] *American National Standard Performance Criteria for Hand-Held Instruments for the Detection and Idenfication of Radionuclides, ANSI N42.34-2006*, IEEE Std., Rev. ANSI N42.34-2006, 2007.

[37] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, Sep 1999.

[38] R. Fletcher, *Newton-Like Methods*. John Wiley Sons, Ltd, 2000, pp. 44–79.

[39] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, pp. 251–257, 1991.

[40] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015.

[41] X. Yu and G. Chen, "Efficient backpropagation learning using optimal learning rate and momentum," *Neural Networks*, vol. 10, p. 517527, 1997.

[42] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $o(1/k^2)$," in *Soviet Mathematics Doklady*, vol. 27, no. 2, 1983, pp. 372–376.

[43] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

[44] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," *CoRR*, vol. abs/1212.5701, 2012.

[45] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference for Learning Representations*, 2015.

[46] J. S. Bridle, "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition," *Neurocomputing*.

[47] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[48] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.

[49] A. Zheng, *Evaluating Machine Learning Models*. O'Reilly Media, Inc., 2015.

[50] T. Goorley, M. James, T. Booth, F. Brown, J. Bull, L. Cox, J. Durkee, J. Elson, M. Fensin, R. Forster, J. Hendricks, H. Hughes, R. Johns, B. Kiedrowski, R. Martz, S. Mashnik, G. McKinney, D. Pelowitz, R. Prael, J. Sweezy, L. Waters, T. Wilcox, and T. Zukaitis, "Features of MCNP6," *Annals of Nuclear Energy*, vol. 87, pp. 772–783, 2016.

[51] J. B. Stinnett, "Automated isotope identification algorithms for low-resolution gamma spectrometers," Ph.D. dissertation, University of Illinois at Urbana-Champaign, 2016.

[52] E. Browne and R. B. Firestone, *Table of Radioactive Isotopes*, V. S. Shirley, Ed. Wiley-Interscience Publications, 1986.

[53] L. Devroye, *Non-Uniform Random Variate Generation*. Springer-Verlag New York Inc., 1986.

[54] G. Knoll, *Radiation Detection and Measurement*, 4th ed. John Wiley and Sons, 2010.

[55] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga,

S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org.

[56] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Muller, *Neural Networks, Tricks of the Trade*. Springer, 1998, ch. Efficient BackProp.