Summer 7-2023

Asset Cueing Nuclear Radiation Anomaly Detection Using an Embedded Neural Network Resource

April Inamura
University of Nebraska-Lincoln, ainamura@huskers.unl.edu

ASSET CUEING NUCLEAR RADIATION ANOMALY DETECTION USING AN EMBEDDED NEURAL NETWORK RESOURCE

by

April Inamura

A THESIS

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfilment of Requirements

For the Degree of Master of Science

Major: Electrical Engineering

Under the Supervision of Professors Michael Hoffman and Sina Balkır

Lincoln, Nebraska

July, 2023

ASSET CUEING NUCLEAR RADIATION ANOMALY DETECTION USING AN EMBEDDED NEURAL NETWORK RESOURCE

April Inamura, M.S.

University of Nebraska, 2023

Advisors: Michael Hoffman and Sina Balkır

Nuclear radiation detection is inherently a challenging task, coupled with a high background variation or increase in anomalies, the accuracy for detection can plummet. A key factor in the success of nuclear detection hinges on the sensor's ability to generalize its model and directly leads to the model's robustness. The goal of this project is to develop algorithms suitable for use on the University of Nebraska-Lincoln's Pingora chip, a low-power, system-on-chip device with an active neural processing unit (NPU) made for nuclear radiation detection. The thesis aims to improve Pingora's overall generalization ability in nuclear radiation source detection. A multi-phase multi-layer perceptron neural network (MLPNN) design was used to train the network offline until a low error rate was achieved. The development dataset includes over 100,000 samples with varying source presence. The difficulty of working with this dataset was the high variation in the data characteristics for both background and source samples. Regardless, the model achieved on average a 12% error across all test sets, including the worst-case dataset, which was defined as the dataset that includes the least identifiable characteristics.

ACKNOWLEDGMENTS

I wish to express my deepest appreciation to my advisors, Dr. Michael Hoffman and Dr.Sina Balkir for guiding and teaching me throughout my graduate program.

I would like to further extend a special thanks to Ph.D. Candidate Samuel Murray for his patience in answering my endless questions and his kind manner to aid me throughout my program development for my thesis.

Lastly, I wish to offer my sincerest thanks to my loving family, Miyuki Inamura, Ko Inamura, Will Inamura, and Laurel Hansen, for their undying support and understanding.

Contents

List of Abbreviations

GS	Gamma Spectroscopy
ML	Machine Learning
CV	Cross-Validation
MLPNN	Multilayer Perceptron Neural Network
MLP	Multilayer Perceptron
ANN	Artificial Neural Network
SVM	Support Vector Machine
NN	Neural Networks
UNL	University of Nebraska-Lincoln
NPU	Neural Processing Unit
SRAM	Static RAM
JHU	John Hopkins University
APL	Applied Physics Laboratory
SK-Fold	Stratified K-Fold
ST	Source True
SF	Source False

Chapter 1

Introduction

Nuclear materials have become an integral part of our society, used in various fields such as medicine, defense, energy, agriculture, and scientific research. As these materials become increasingly integrated into our daily lives, the importance of nuclear radiation detection has become a globally pressing issue for human and environmental safety. For example, a recent study showed an increase in the illicit traffic of nuclear materials. Many of these materials were also often trafficked during the transfer of legal radioactive material shipments to mask the properties of the illegal material [1]. Nuclear radiation detection has a wide range of applications, such as nuclear waste monitoring [2], medical scanning and imaging [3], and determining cosmic ray intensities [4]. Such need and prevalence for nuclear detection have driven industry and academia alike to engineer more sophisticated and robust detection and identification methods. I am a master's student attending the University of Nebraska-Lincoln and this thesis explores machine learning (ML) in depth, specifically neural networks, as a potential avenue for nuclear detection with improved generalization capabilities. The following chapters of this thesis will cover nuclear radiation detection, machine learning, data pre-processing, the developed project program, and the results achieved.

1.1 Radioactivity

This section briefly details the basics of radioactivity and the various types of nuclear decay. Gamma decay will be highlighted as it is the main type of decay that is analyzed in this thesis. The basics of nuclear decay begin when a nuclide with a lower binding energy, decay into its neighbor nuclide that contains a higher binding energy. These nuclides can be separated into "parent nuclides" and "daughter nuclides", with the parent being the nuclide with lower energy and the daughter being of higher energy. During this process, kinetic energy equivalent to the sum of the difference between the parent and daughter nuclide is emitted and this energy is known as ionizing radiation. The energy levels from ionizing radiation are commonly on the scale of MeV [5]. There are largely three types of nuclear decay: Alpha, beta, and gamma. These types of decay relate to the different types of particles emitted during ionization.

Alpha and Beta Decay

Alpha decay occurs as a result of the emission of an alpha particle, which is a ^{24}He nucleus. Alpha particles mostly lose their initial energy very rapidly and thus have a low penetration ability, which may be as thin as a layer of paper. Beta decay occurs when a neutron changes into a proton by emitting a beta particle or an electron. Beta particles travel near the speed of light and thus have a much higher penetration ability, up to a millimeter of aluminum [6].

Gamma Decay

The lifetime of an excited state nuclide is generally very short, on the order of 10^{-10} to 10^{-13} s. However, there are some nuclides with a metastable excited state, which results in a much longer lifetime; these are known as "nuclear isomers". For example, the $^{27}_{58}Co$ has a metastable excited state with an energy of 24.9 keV and a half-life of 9 hours [7]. When a "nuclear isomer" either directly or indirectly decays to a lower energy level or to its ground state, one or more photons are emitted with an energy value in the keV to MeV range [8]. Unlike alpha or beta particles, gamma particles have a much higher penetrating power and can be observed up to a few centimeters of

lead or fully penetrate the human tissue. Additionally, gamma rays are one of the most common forms of emission and are active in many radioactive reactions. Due to the nature of this decay and its high energy characteristics, gamma rays are used as the main source of radiation data in this thesis.

1.2 Gamma-Ray Spectroscopy

Gamma-rays occur as a result of gamma decay and contain the higher energy form of electromagnetic radiation. Many nuclear reactions include gamma ray emissions thus posing them as a desirable energy form to detect. One method of gamma particle detection is Gamma Spectroscopy (GS), which involves the capture of individual gamma rays' pulse heights. As previously mentioned, gamma ray emissions have a broad band of energy levels ranging from keV to MeV. Often, multiple particles may be emitted at the same energy level from the source. Gamma spectroscopy then captures this burst emission by representing energy into a histogram generated by quantizing energy into pulse heights, resulting in a gamma spectrum. These gamma spectrums play an integral role in isotope identification and detection, and are covered in section 1.3.

1.3 Nuclear Radiation Identification and Detection Methods

Nuclear detection and identification tend to work hand in hand, where one cannot exist without the other. The main difference between detection and identification would be in its application. Nuclear detection will likely lead to the sensor activating a higher asset or power-consuming device in the sensor or to purely begin logging data to be identified. The complexity of detection is determining how the device will behave in response to the collected gamma spectrum. What constitutes an isotope presence? How will background vs valid radiation data be handled? What

level of accuracy and precision can we expect depending on the method? These are only some of the core issues that isotope detection can face. Thankfully, several different detection methods exist to date such as peak-finding, Artificial Neural Networks (ANN), Support Vector Machines (SVM), Decision Trees, and the Bayesian Method [9]. This thesis strictly explores machine learning methods in ANNs and will be covered in more detail in Chapter 2. The ANNs intuitively adopt peak-finding strategies, but also address many of the method's issues. However, it is important to cover some basics in peak-finding before progressing into other machine learning methods. Peak finding, sometimes known as peak searching, is the most traditional and common method of isotope identification. The objective of peak finding is to locate gamma-ray peaks in the spectrum by measuring the number of gamma rays detected within an energy level. There are also multiple ways to implement peak detection. Some basic implementations include regions of interest, channel differences, and peak derivative searches. Regions of interest take leverage on the fact that peak detection is driven by target energy ranges and does not care for the energy distribution of the spectrum. Thus, rather than observing the entirety of the spectrum, regions of interest can be specified for further analysis. Channel differences use an intuitive peak searching method where the number of counts per channel is referenced with respect to the channel before and after it to analyze the rise and fall behavior of the spectrum. From a statistical perspective, if the spectrum follows enough subsequent rises before a fall, the point of interest may be labeled as a peak. Lastly, the derivative peak search method, also known as the derivative method, makes use of the natural Gaussian distribution peaks in the spectrum by taking the derivative of the distribution. The derivative of a Gaussian distribution has features that can be used to detect the presence of peaks. Figure 1.1A visualizes the first and second derivatives of a Gaussian random distribution. The first derivative highlights the centroid peak at the center of the sign change, the second derivative highlights the peak by crossing the y-axis. Figure 1.1B presents an example of the first and second derivatives in peak detection using real spectrum data [10].

These Gaussian derivatives cannot be mathematically represented in the spectrum since they

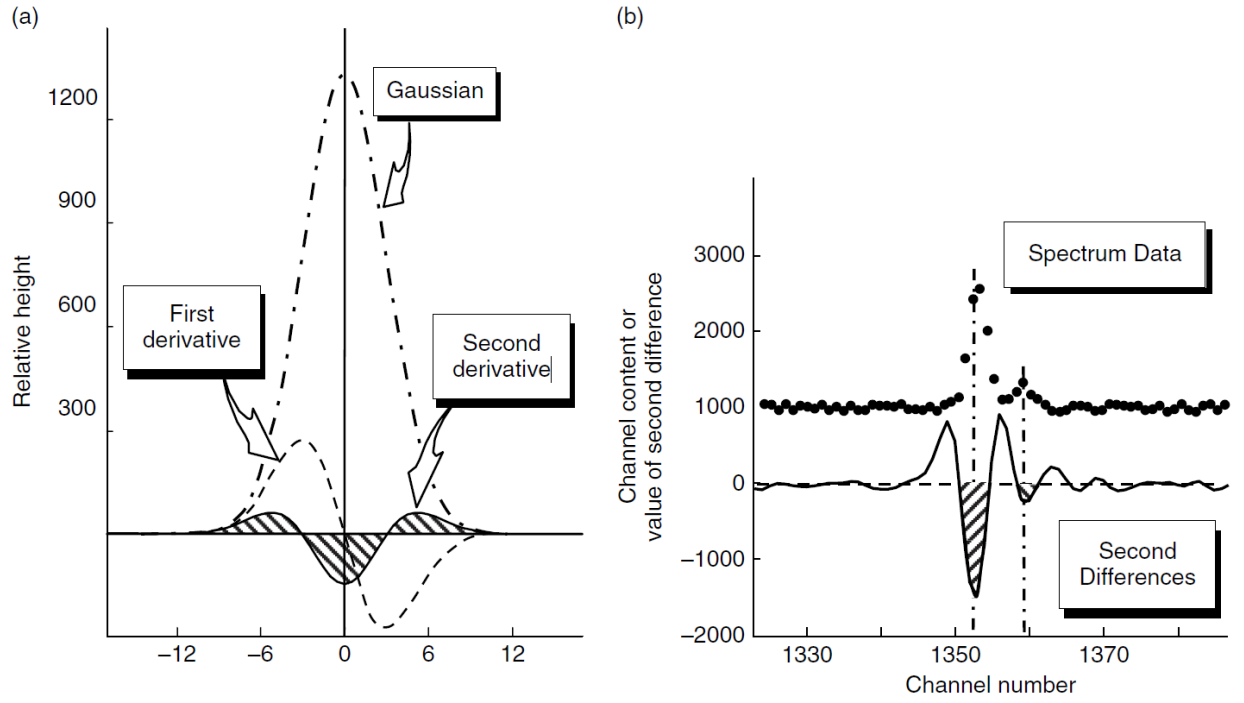


Figure 1.1: Peak Finding

are not pure Gaussian distributions; however, an approximation can be achieved. The derivative method identifies a peak based on calculating the difference between channels as an approximation to the gradient. Peak detection, while the most traditional and well-used method of isotope identification, comes with a myriad of limitations. The most prevalent is its lack of robustness in design to external variables such as identification in the presence of multiple isotopes, especially a collection that may include isotopes with similar energy levels. This is not even considering the inevitable background variation surrounding the system, which will add more errors to the peak detection method. The device may experience several false positives or even miss a gamma emission. Due to the issues listed above, peak searching generally does not produce reliable results in real-time or automated applications. Machine learning (ML) was introduced as a potential alternative to peak-finding methods and has found major success in recent years. Researchers have turned to ML as a popular choice of isotope detection to address the real-time application constraints identified in peak-finding. What peak-finding methods lack is the ability to increase

generalization. ANNs are specialized in this task and are a part of the motivation for selecting this form of ML for this thesis. In this thesis, a multi-layer perceptron neural network (MLPNN) was selected as an approach to nuclear isotope detection in end-to-end testing. The network architectures will be covered in detail in Chapter 2.

1.4 UNL Background and Project Motivation

The University of Nebraska-Lincoln (UNL) has developed a novel, low-power, system-on-chip for nuclear radiation detection. The chip, known as Pingora, contains an on-chip hardware NN accelerator that allows for real-time isotope identification in addition to detection. The NN accelerator, notated as the Neural Processing Unit (NPU), utilizes a multi-layer perceptron (MLP) architecture and performs inference-only operations with pre-trained synaptic weights to predict the proportion of each radioisotope contribution in the collected histogram. The NPU on the Pingora is a fixed-point ANN hardware accelerator that calculates a single layer of an MLP. The NPU processes an MLP layer largely in three parts: the input vector, the synaptic weights, and the output vector. The input vector and output vector must both be of signed Q0.15 integers (signed int16) and the synaptic weights are 24-bit signed Q8.15 numbers (signed int24). Previous implementations of the Pingora use 64 neurons for its input layer and 5 neurons as its output layer [11]; however, the number of hidden neurons or hidden layers can vary depending on the source isotopes observed as well as the number of isotopes observed in one histogram. There is no specified limit to the number of hidden neurons or hidden layers that the NPU specifies. However, a critical limitation of the NPU is that all network weights, the input vector, and the output vector must fit within a single 16 KiB dual-port SRAM [12]. This restriction in memory limits the complexity of the network, which is an issue when dealing with more sophisticated applications of this chip. The output of the NPU produces 5 separate percentages that represent the different proportions of contributions to the histogram and are known as relative abundances.

When all relative abundances are summed together, the sum should equal 100%. Four of the five relative abundances are individual isotopes, currently tuned for Ba-133, Co-60, Cs-137, and Na-22. The data of the individual isotopes are included in the training of the network. The fifth relative abundance is called background and is equivalent to any noise or isotopes that are not included in training and thus are unidentifiable. This method of training and testing is an efficient and simple method of identification. However, for more complex applications where background variation or background anomalies are frequent, this system will experience difficulty in analyzing and deducing the detection of isotopes. The greater concern is that due to the indiscernible background anomalies, the device will experience greater energy loss from constant false detection. The intention of this project is for the device to remain in sleep mode with a passive low-cost detection method, which will then activate a higher asset resource for detection confirmation or identification. The goal of this thesis is to achieve an error rate of source presence that is less than 10% for either source or background sample. To build a program with a robust design and high-accuracy detection, UNL has partnered with John Hopkins University (JHU) Applied Physics Laboratory (APL) to make use of its expansive nuclear radiation database for development. The description and use case of APL's database are covered in detail in Chapter 3. The program for this project was developed in MATLAB.

Chapter 2

Machine Learning

Artificial neural networks assume the design of neurons in the brain, where neurons are connected through synaptic brain nerves. There are multiple forms of neural networks such as MLPNN, CNN, and RNN. This thesis uses the same NN architecture that is used on the Pingora hardware, the multilayer perceptron neural network (MLPNN). The network implementation is detailed in Chapter 4 and the results in Chapter 5. The content of this chapter targets the details of each network architecture.

2.1 MLPNN Approach

The multi-layer perceptron neural network (MLPNN) is one of the most classic and powerful forms of neural networks and serves as a foundation for many ANNs. An MLPNN is a feedforward network, meaning that data only flows in one direction, input to output, and no feedback connections from the output are fed back into the network. The network architecture of an MLPNN is generally a series of functions chained together to be applied to the input data also called neurons. The series of functions are separated by each layer of the network and a similar mathematical operation is applied to each layer only varying in the applied function otherwise

known as the activation function. Figure 2.1 provides an example of the MLPNN architecture from a high level. In an MLPNN, there are generally 3 layers as shown in Figure 2.1, the input layer, the hidden layer, and an output layer. There will always be an input and output layer to the network because either layer signifies the input data into the network and trained network data in the final layer. On the other hand, the result of each hidden layer is a result of the applied activation function of that layer and does not result in a value of the desired output, thus the data viewed in a hidden layer is inconclusive giving its meaning behind the name of a "hidden layer". Additionally, the number of hidden layers is not restricted to only one. There can be no hidden layers or a number of hidden layers. Note that increasing the number of hidden layers to greater than one qualifies the MLPNN as a deep network because it is analogous to the depth of its layers by the number of hidden layers there are in the network. Therefore, the example in Figure 2.1 is classified as a 2-layer (some articles do not consider the first layer) shallow network.

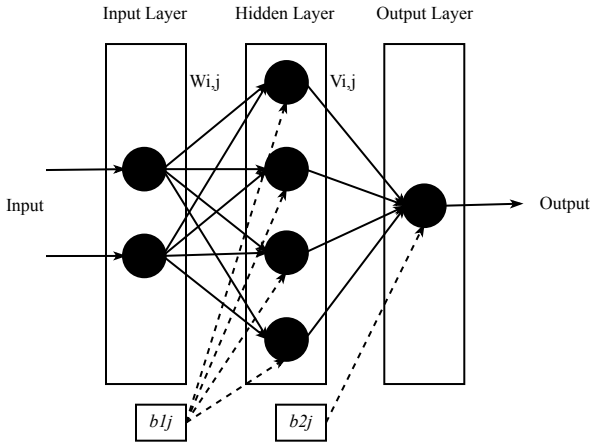


Figure 2.1: FeedForward MLPNN

The MLPNN accepts a numerical input of any vector size from n_{I1} to n_{Ii}, where i is the number of input neurons. For our application, the number of input neurons is the same as the size of the gamma spectrum. Minus the first layer, the neuron interaction for every layer is fundamentally the same. Equation 2.1 represents the transfer function of the network, where the layer neurons are multiplied with the synaptic layer weights, and optionally a bias value is applied

$$a = xW + b \tag{2.1}$$

The weights of the network are the key defining factor of the network's ability to learn and predict the results of the network. The bias value is key in applying the activation function as it acts as an offset value for the layer neurons to be shifted into an operable range of the activation function. The second half of the layer operation is handled by the activation function represented in Equation 2.2.

$$d = f(a) \tag{2.2}$$

The activation function serves as a simple yet powerful mathematical tool to determine the neuron's importance in the network by choosing to activate the neuron or not. There are several different activation functions that can be applied to MLPNNs such as linear, logsig, tansig, softmax, etc. Figure 2.2 provides a high-level view of the network layer operation.

2.2 Training and Backpropagation

This thesis makes use of the MATLAB built-in neural network library, where multiple complex training programs were explored based on the size and complexity of the dataset. In particular, three training algorithms were used in the development of the program. These include the gradient descent with learning and momentum rate backpropagation (traingdx), the scaled conjugate gradient method (trainscg), and the Levenberg-Marquardt backpropagation method (trainlm). The traingdx method was first explored based on the heritage of the MLPNN design used in Murray's previous work [11]. The trainscg method is an upgraded version of traingdx, with increased descent optimization and search for global minimums. This particular training method was also more efficient with larger datasets.

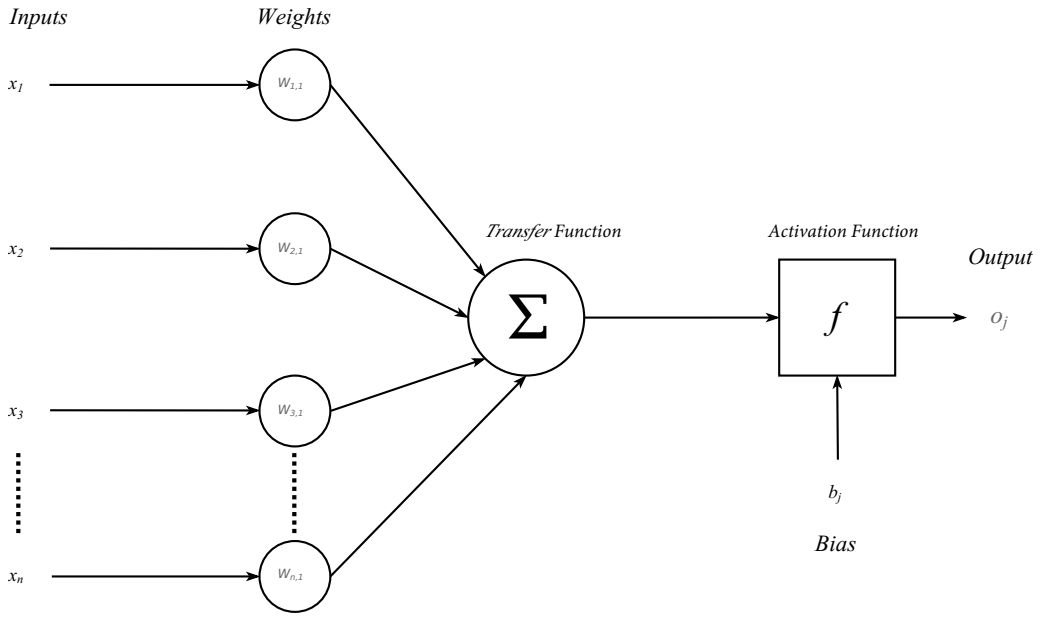


Figure 2.2: Activation Function

The trainlm method has a longer training per epoch for the training set but was recommended as a good training option for colossal datasets. The APL development data used in this thesis contain 100,000+ samples, which does classify the dataset as colossal. The details and effects of these training algorithms are provided in detail in Chapter 5. The details for these complex mathematical training algorithms can be found by searching any of the keywords on Mathworks [13].

Chapter 3

Data Handling

This section is dedicated to describing the data used for the development of the machine learning program and how the raw data was handled prior to training or testing on the network. Data handling plays a key role in the behavior of the network and in many cases can influence the pass or fail rate of the network objective.

3.1 Development Data

Historically, UNL has collected its own data on small scales in controlled lab space for the development of Pingora. Lab-taken data is valuable due to its controlled nature, however, to develop a program to target robustness and generalization ability, the UNL's dataset can be limiting. Hence, UNL has partnered with John Hopkin University (JHU) Applied Physics Laboratory's (APL) Dr. Peter Heimberg to make use of their expansive nuclear radiation database to develop this project. To compare, UNL's radiation database includes hundreds of gamma spectrums, whereas APL's starter kit data set includes over 100,000 spectrums in the source files alone and falls under the categories of a colossal data set. These databases were supplied to UNL through the DTRA Algorithm Development Resource page.

3.1.1 Data Variables

APL provided UNL the "Starter Kit Data Set", which includes 2 sets of 9 colossal datafiles. One set contains "source" data, and another is labeled as "background" data. Each data file in either set contains 43 variables that are divided by column and the samples that are taken in 1-second increments are divided by row. The variables of interest are defined in Table 3.1 below. The full description of the variables in the dataset can be found in a file that came with the starter kit dataset under Supporting Documentation in Appendix A.

A key parameter to note from Table 3.1, is IPRES, which stands for is-source-present. This variable identifies for every sample, whether a source was present during the collection of that sample. The IPRES variable is a key parameter that acts as the predicted value of this network. The core difference between the data "source" and the "background" folder is that the "background" folder does not contain any samples with IPRES = TRUE. Whereas the "source" folder contains samples with both IPRES = TRUE or FALSE. The ratio of samples that are TRUE vs FALSE is heavily skewed towards FALSE. If training on the network was implemented with the raw data, the network would be highly tuned to accepting spectrums with a false source presence. Given the small quantity of IPRES = TRUE in the "source" folder, the data in the "background" folder was not used in this research. Moving forward, all data is implied to be from the "source" folder of the APL Data Starter Kit.

Figure 3.1 presents the source data file 20130617 from APL's open file viewer (OFV) software. The OFV allows the user to easily view the samples collected in the data file with the x-axis representing time, the left y-axis representing gross counts, and the right y-axis representing neutron counts. Finally, the second graph above the green and grey graph, in red represents the histogram of the sample. The OFV allows the user to select a single sample or sum multiple histograms together. The APL database is unique to the UNL database not only due to its size but also in its application. The APL data was not collected in a stationary environment, but rather in

Table 3.1: Data Variables of Interest

LINE #	Name	ID	Description
1	record	**REC**	record number
3	utc-time	**UTC**	The timestamp in human-readable time (UTC) format: YYYMMDDHHMMSS
4	timestamp	**TS**	The timestamp in milliseconds from Jan 1, 1970 (unix time)
7	gc0	**GC0**	gross count for gamma element 0 (integer).
12	spectrum-channels0	**SC0**	The spectrum for gamma element 0 (comma delimited floating point or integer values)
13		**GC1, SC1, etc.**	<Repeat gamma elements here for elements 1,2,3>
14	nc0	**NC0**	The neutron gross counts
17			<Repeat neutron elements here for elements NA>
18	is-in-zone	**IZONE**	TRUE if the sample has taken inside the test zone, FALSE outside the test zone
19	is-closest-approach	**IAPPROACH**	TRUE if this sample was estimated (by location) to be the point of closest approach to the source, FALSE otherwise or if the quantity is not calculated
20	is-source-present	**IPRES**	TRUE during times when a source configuration is present in the test area and FALSE otherwise
21	source-id	**SID**	The source configuration name (or "no source"). Field may be an empty tab when no source configuration is in use
22	source-offset	**SOFF**	[ft] The offset distance between the detector's distance of closest approach and the source, or 0 if "no source". Field may be an empty tab when no source configuration is in use.
25	distance-to-doca	**DOCA**	[ft] The distance to the point of closest approach. Field is set to -1 when unspecified.
26	is-active	**IACTIE**	TRUE if data were within experimental "live" conditions, FALSE otherwise. This is used to eliminate times when the sensors may have been inadvertently exposed to a test source or when detectors were being initialized.

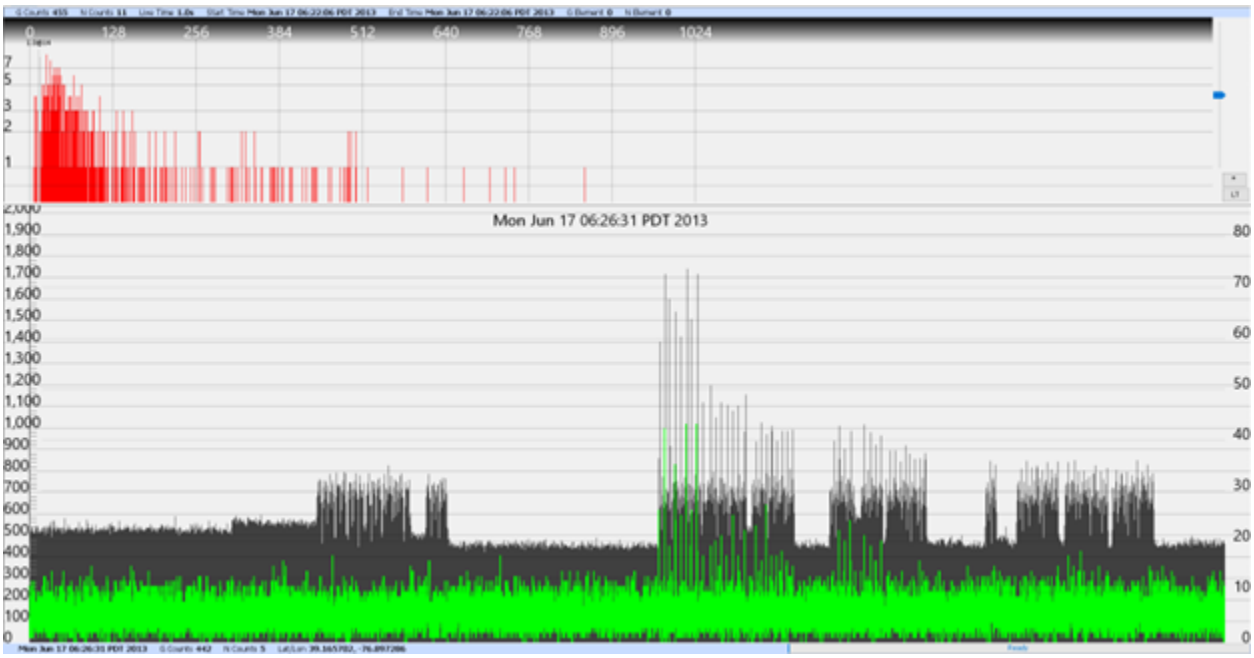


Figure 3.1: JHUAPL Source 20130617 Open File Viewer

a dynamic environment. By viewing the GPS functionality of the OFV as shown in Figure 3.2, we can determine that the radiation sensor was mounted to the back of a car and driven around on a track. Additionally, the windows of source presence and no source coincide with the midpoint of the track.

There is another set of data files that accompany the “source” and “background” files mentioned earlier, known as valid files. These files contain three variables differentiated by column. The variables in the valid files are UTC-time, timestamp, and is-valid. The purpose of the valid files is to determine whether the particular sample observed in the “source” or “background” file is valid or not. This nomenclature is separate from source presence and solely measures the validity of the data in each sample. The reason for what causes a data sample to be invalid is unknown; from speculation, we may assume that the sample was taken at the start-up of the device or during the installation or removal of sensor shielding.

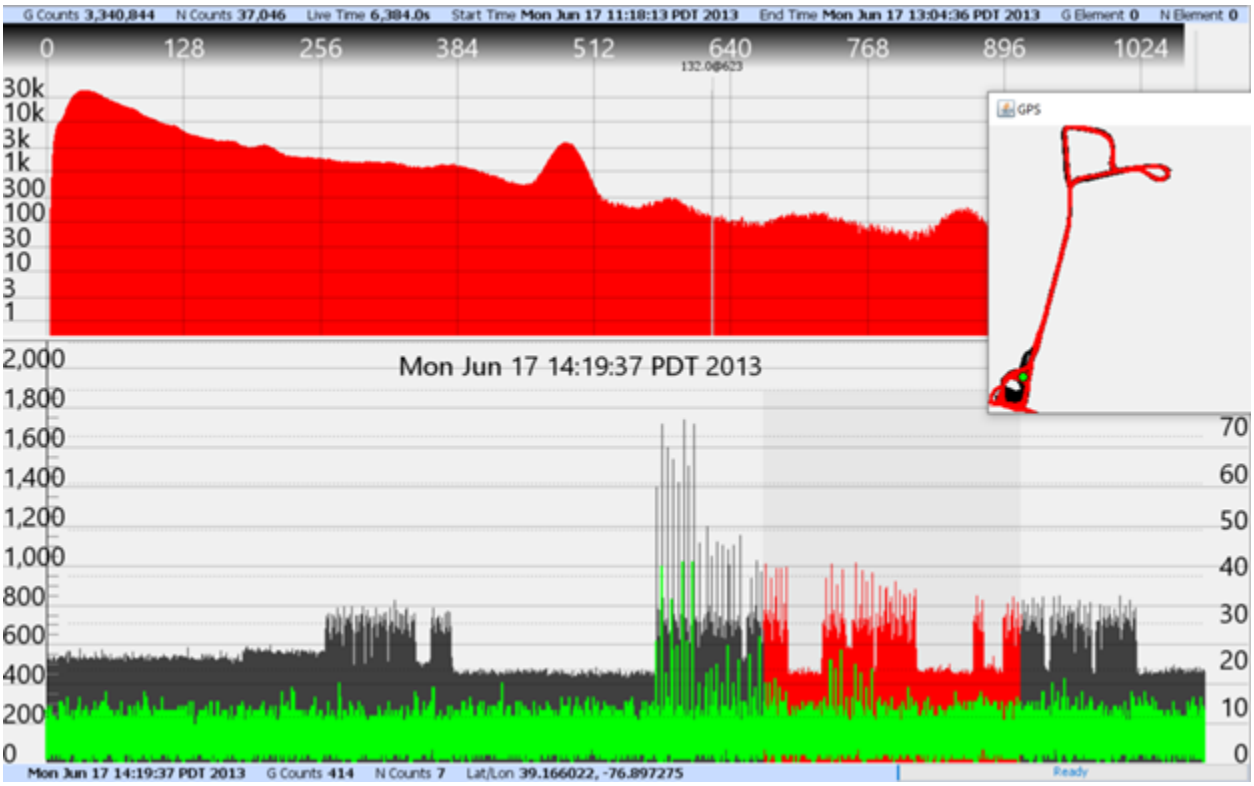


Figure 3.2: Open File Viewer GPS

3.1.2 Data Characteristics

The dataset provided by JHU is massive, on the scale of 100,000 samples after compressing the dataset. Given the nature of any large dataset, there are a lot of characteristics throughout the data files that mirror one another and other properties that provide discrepancies between certain samples and data files. For example, file 20130624 contains the most samples out of all of the data files and the highest number of source true data. The average of the source false or background data generally falls between 200-500 counts, whereas the average source true data is 400-700 counts. There are outliers in either case, where the source true data has a sample with 2000 counts or lower around 200 counts. These outliers are the main contributors to the difficulty of classifying this dataset. Achieving a low error rate in the classification of not only the average samples in the dataset but also the outlier samples will define whether a network has good generalization.

Different combinations of the data files were used for testing vs training based on this discrepancy in characteristic and the results of these tests are covered in Chapter 5. Table 3.3 provides the statistics of the source presence per data file and compares it against all the data files in the source folder.

Table 3.3: Dataset Statistics

Filename	ST Data	SF Data	Total File Data	Equal ST/SF #	All data - datafile
’20130617’	2661	18632	21293	5322	114584
’20130618’	9181	40811	49992	18362	101544
’20130619’	5618	24725	30343	11236	108670
’20130620’	9431	40835	50266	18862	101044
’20130621’	5163	23490	28653	10326	109580
20130622’	5106	23082	28188	10212	109694
’20130624’	12842	35235	48077	25684	94222
’20130625’	3629	18678	22307	7258	112648
’20130626’	6322	20792	27114	12644	107262

3.2 Data Compression and Sampling

3.2.1 Data Compression

The raw data as described in Section 3.1.1, define that for each sample there are 4 gamma-ray detectors and a single neutron-ray detector. Each gamma ray histogram count per bin average around 200 on the higher-end values. If the program relied on the gross counts of each sample, then the program would be weak against outlier background data that have on average higher counts. For example, say for a single sample there is an average source true count rate of 250 counts and an average background count rate of 50 counts. Then another sample in the same set may have an average source true rate of 500 counts and a background rate of 200 counts. In this scenario, the margin between the source true and background rate between samples is small, there are many more instances where this margin is even smaller! To highlight the contrasting differences between the source to background data, the overall average of each bin was increased by summing all 4 gamma histograms together by bin for each sample. There is only one neutron ray histogram, so the data was not altered. To further compress the data and accent the bin-to-

bin count levels, the summed 1024 bin histogram was rebinned to 64 bins using 16 bin intervals. The rebinning method is simple and consists of summing 16 bins together from bin 1 to bin 16, creating the 1st bin of the compressed dataset. The second bin would sum from 17 to 32, and so on and so forth. The initial dataset is comprised of the rebinned dataset with another variable to numerically determine the source presence as listed in Table 3.5.

Table 3.5: Compressed Dataset

LINE #	Field	Description
1	REC	Record Number – Specific to each file
2	UTC	(utc-time) - The timestamp in human-readable time(UTC) format: YYYYMMDDHHMMSS
3	TS	Timestamp – The timestamp in milliseconds form Jan 1, 1970 (unix time)
4	GC0	Gross Counts – The gross counts for gamma element 0 (integer)
5	SC0	Spectrum-channels0 – The spectrum for gamma element 0 (comma delimited floating point or integer values)
6	NC0	Neutron gross counts
7	IZONE	Is-In-Zone – TRUE if the sample was taken inside the test zone, FALSE outside the test zone
8	IAPP	Is-closest-approach – TRUE if this sample was estimated (by location) to the point of closest approach to the source, FLASE otherwise or if the quantity is not calculated
9	IPRES	Is-Source-Present – TRUE during times when a source configuration is present in the test area and FALSE otherwise
10	DDOCA	Distance-to-DOCA – [feet] The distance to the point of closest approach. Field is set to -1 when unspecified.
11	IACT	Is-Active – TRUE if the data were within experimental “live” conditions, FALSE otherwise. This is used to eliminate times when the sensors may have been inadvertently expose to a test source or when detectors were being initialized.
12	SC0_BIN	The spectrum-channel0 has been binned from the original 1024 element histogram to a 64 element histogram. The format is converted to a MATLAB ready “double” matrix.
13	SRC_TF	A MATLAB integer value to represent IPRES. For TRUE the value is 1 and for FALSE the value is 0.

3.2.2 Data Sampling

To further compress the data between samples, interval sampling or convolutional sampling was used after data compression. The basic idea around interval sampling is like the convolution method used in mathematics. Figure 3.3 provides an example of the interval sampling method with a window of 4 seconds. The original dataset includes 9 samples, however, post the interval sampling method, the remaining sampled data include 6 samples. In each window, 4 samples sum together the numerical variables in the sample such as gross counts, spectrums, neutron counts, etc. The intention of interval sampling is to capture the average bin counts into a single histogram for training. If the first 5 seconds of data are source false with lower average bin counts and the last 4 seconds are source true with high average bin counts, then the general trend of the increase in counts should be captured by the convolution. Particularly, the last convolved signal should have a significantly higher average in bin counts than its previous samples in the interval sampled data. Note that for every interval combined sample, the samples were only combined if all source presence types matched. If the first 5 samples are source false, then 2 sampled signals would output from this algorithm and 1 source true sample would exist. All other samples would be ignored because they are mixed source presences and that would skew the source presence average counts.

3.3 Cross Validation

The basic idea behind machine learning is that, given a dataset with a certain characteristic, a predictive model is trained on the input data and then able to predict an outcome based on its behavior. For every model, there are three datasets – the training set, the test set, and the validation set – each of which plays a different but vital role in model learning. The training set is a dataset that is used for training the model. The model weights and biases are affected and adjusted based on the error. The test set is a dataset that is used to evaluate the model and the

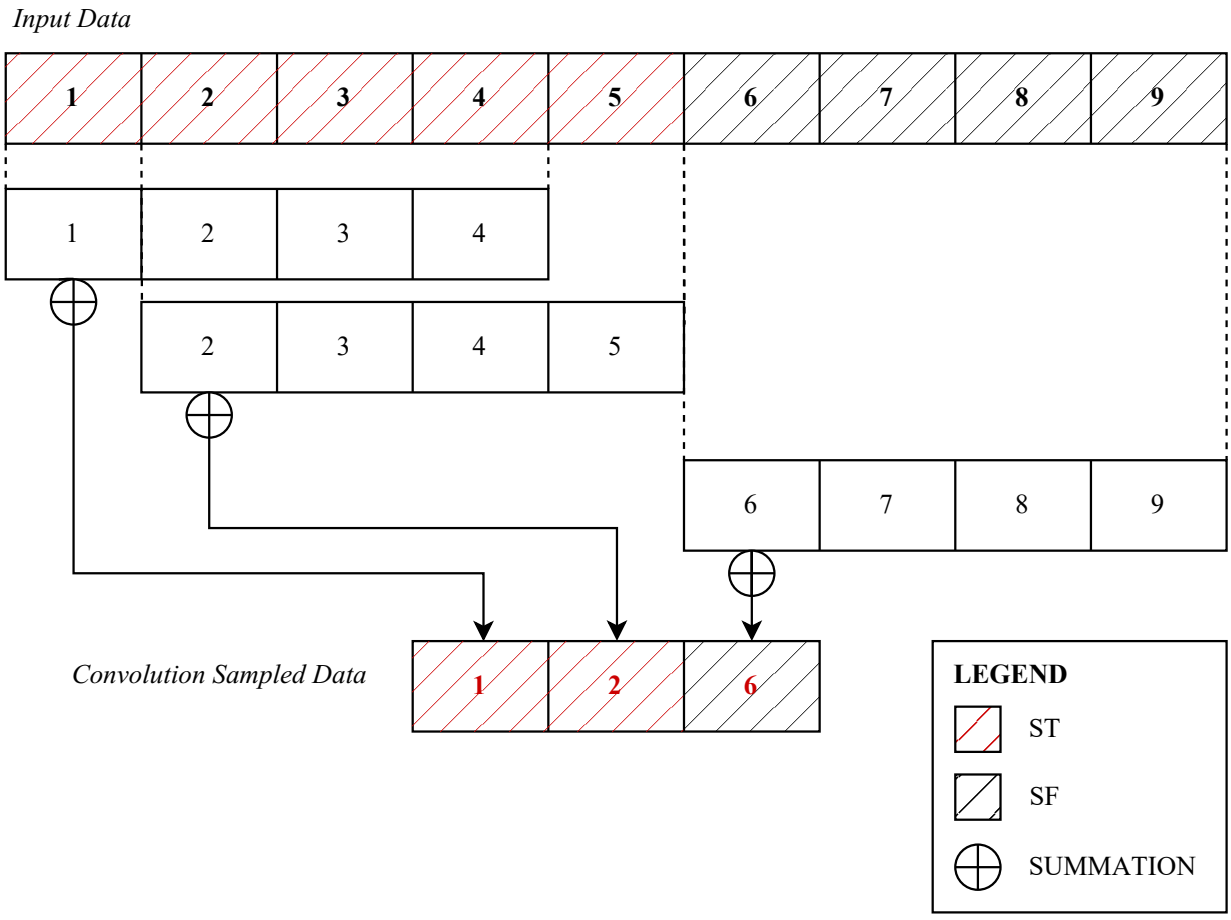


Figure 3.3: Convolutional Interval Sampling

test set is purposefully excluded from the training set. This means that the model parameters, weights, and biases are unaffected by the test set. The test set purely evaluates the predictive performance of the model. The validation set is used to adjust the hyperparameters of the model, but not the weights and biases. The validation set acts more as a fine-tuning instrument to the model than causing a drastic change to the performance like the training set. Traditionally, the training set is the largest dataset of the three, whereas the test and validation set is smaller and equal or similar in size. Most machine learning problems start with a large single dataset that is then separated into a train, test, and/or validation set. Properties that hinder a model's ability to generalize or converge, such as over- or under-fitting, can be avoided with the use of these three

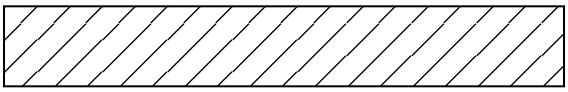


Figure 3.4: Re-substitution Method

datasets. Overfitting is the idea that the model perfectly adapts to the training set but produces entirely different results with high volumes of an error on the test set. This behavior indicates that the model is overfitted to the training set. On the other hand, underfitting occurs when the model is incapable of distinguishing the relations between the class labels and other network variables. Cross-validation (CV) is a data resampling method that is used to estimate the generalization ability of a predictive model and prevent overfitting. The previously described method of a model containing a train-test-valid set is the simplest form of cross-validation; however, there are many methods of cross-validation, each with its own benefits and drawbacks. In this section, the different forms of cross-validation and their applications will be defined in detail. To preface, though the original intention of cross-validation is to evaluate the predictive performance of the model, this project makes use of different CV methods to organize the training and testing datasets at different points throughout the program. The full utilization of the cross-validation methods is explained in Chapter 4.

3.3.1 Re-Substitution Method

The re-substitution method uses the entire dataset for both the train and test sets. Since no new data is introduced into the network, the train and test errors are equal. If the network is provided with an external dataset, there is a high likelihood the network will be overfitted and will respond poorly to the new data. Figure 3.4 provides a block diagram of the train and test sets.

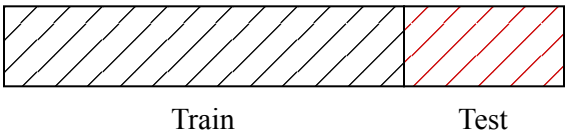


Figure 3.5: Hold Out Method

3.3.2 Hold Out Method

The hold-out method splits the model dataset into two mutually exclusive datasets, where one is used to train and the other to test [14]. The training set is dedicated to training the model and updating the network weights and biases. The test set is withheld from training and does not influence the network parameters in any way. It is reserved purely to evaluate the predictive performance of the model as a new dataset that is introduced post-training. The split ratio between the train/test set is often not 50%. The training set is about 70 90% of the dataset and the test set is 10 30%; the total of the combined training and test set must sum to 100%. The split ratio largely dictates whether the network will overfit or not. For example, in a network with a 50/50% split and the dataset does not have a fair distribution, then there is a higher likelihood for the network to be skewed and the results will greatly vary from multiple test cases. Figure 3.5 shows a visual representation of the hold-out method applied to a dataset.

3.3.3 Stratified K-Fold Cross Validation

The K-fold cross-validation method splits the dataset into k-equal folds. All the folds hold equal or near equal amounts of data, normally for a dataset size that is not divisible by k, the kth fold will hold the remaining data that did not equally fit into a fold. The model is trained on k-1 folds and tested on the remainder fold. This process is repeated k times for each fold in the k-fold set as shown in Figure 3.6. The model training and testing for each fold are carried out independently. A performance measurement such as the mean squared error (MSE) is produced k times. These performance values are averaged to determine the overall performance of the model. Equation

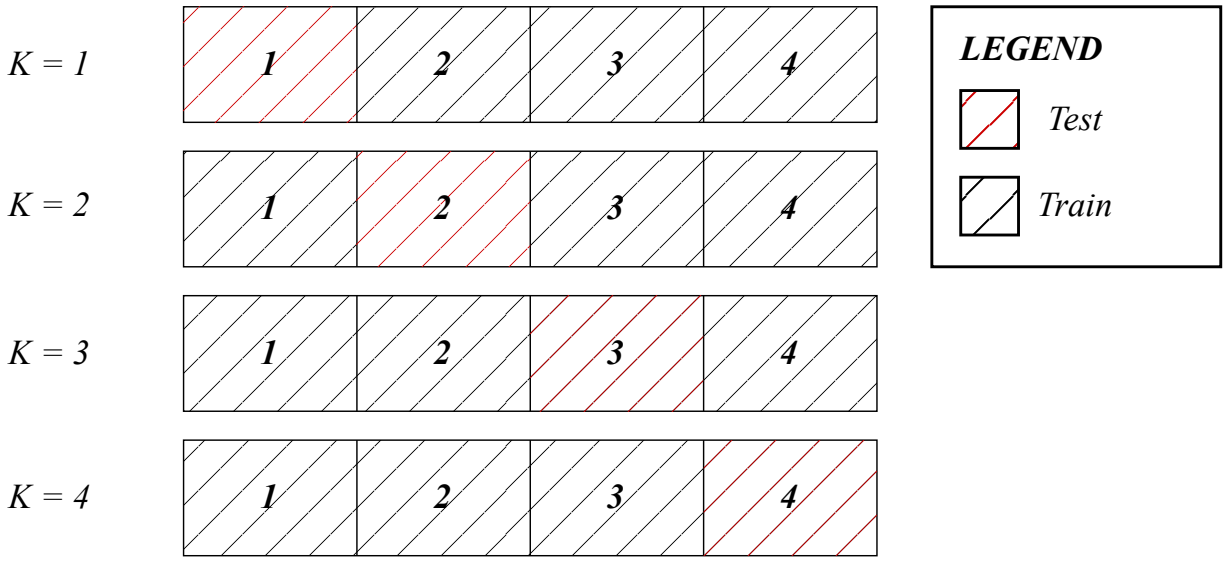


Figure 3.6: Stratified K-Fold Cross Validation Method

3.1 [15] details the performance measurement of the K-fold cross-validation method.

$$\hat{\epsilon}_{cv} = \frac{1}{n}\mathcal{L}(y_i, \hat{f}_{-k}(x_i)) \tag{3.1}$$

Where x_i is the input data, y_i is the expected value of x_i, $\hat{f}_{-k}(x_i)$ is the predicted value from the model of x_i, $\mathcal{L}()$ is the loss function of the model which produces the model performance measurement.

A slight variation to the K-fold method is known as the stratified K-fold cross-validation (SK-Fold) method. Stratification is the process of resampling the data such that each fold has an equal representation of the class label, which creates an unbiased representation of the dataset in each fold. For example, imagine a 4-class label dataset with 4 folds, the distribution of each fold would contain 25% of each class label. Stratification removes single-class biases in the dataset and improves the model performance and robustness. Figure 3.7 provides an example of stratification applied to the hold-out method.

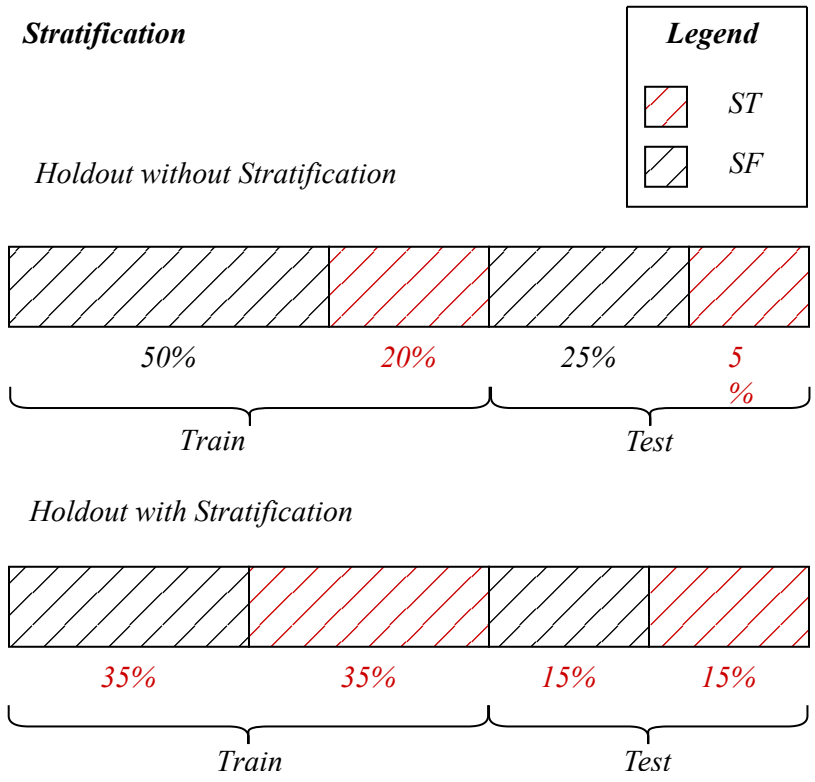


Figure 3.7: Stratification

Chapter 4

Program Design

In this section, the in-depth design and implementation of the core project program for our approach is defined. Many of the machine learning techniques covered in Chapter 2 and data handling techniques detailed in Chapter 3 are referenced in this section.

4.1 Multi-Phase MLPNN

This approach is defined as a multi-phase MLPNN program containing 4 phases; phase 0-3. Each phase divides the data that the network will experience under a given metric and influences the network accuracy in the last phase. Phase 0, data initialization, is always run at the beginning of the main program. Phase 1, the discrimination network, and phase 2, the confidence network, respectively are optional to run. Phase 3, the target network, is the final network run in the program and is also the only network that is used to evaluate the overall program performance and produces the radiation detection results. There are several options in what order or to omit certain phases of the program that is detailed in Section 4.1.1-4.1.4. The data prepared for this approach is the compressed and interval-sampled data described in Chapter 3.2.

4.1.1 Phase 0 - Data Initialization

The input to a neural network rarely uses raw data from the dataset. The network user may perform several data augmentation, normalization, and compression methods to the dataset depending on the program architecture. Phase 0 handles the preparation and organization of the dataset, before running the data in the network. Several datasets are defined prior to the program, thus the first step is to select the interval and number of input neurons to the dataset. The sampled interval options are 1, 3, and 5 seconds, and input neurons are either 64 or 128 neurons. The number of input neurons is a translation of the number of bins in the gamma histogram. The interval selection and input neuron number were selected based on the limits of data compression. Various interval and input neuron sizes were explored. For the number of input neurons, the maximum number of bins provided by the dataset was 1024. This value was reduced by a factor of 2, factoring the number of bins into values like 512, 256, 128, and 64. The interval numbers explored were 1, 3, 5, and 10. The best network performance was achieved at 64 input neurons. Networks with larger input neurons performed poorly due to the lack of contrast between energy bins in the histogram. The general trend of the sampled interval data was that for higher interval values, the performance of the network increased. The network performance plateaued around 5 seconds, thus the value was chosen as the sampling interval for this approach.

The source data is labeled one of two ways, source true (ST) or source false (SF). The data loaded into the main program contains 9 data structures which contain data from each data file provided in the source data folder from APL. The source folder indicates that a source at one point or another was present in the data file that was collected, and the source presence is indicated by line item 20 Appendix A. An equal number of ST and SF are combined into one dataset, where the data files are row indexed by sample. Thus, to combine them, each dataset is concatenated at the end of the previous dataset. For all data files, there is fewer number of data ST than data SF. Thus when extracting the data SF from the data file, periodic sampling is applied to the entire data SF

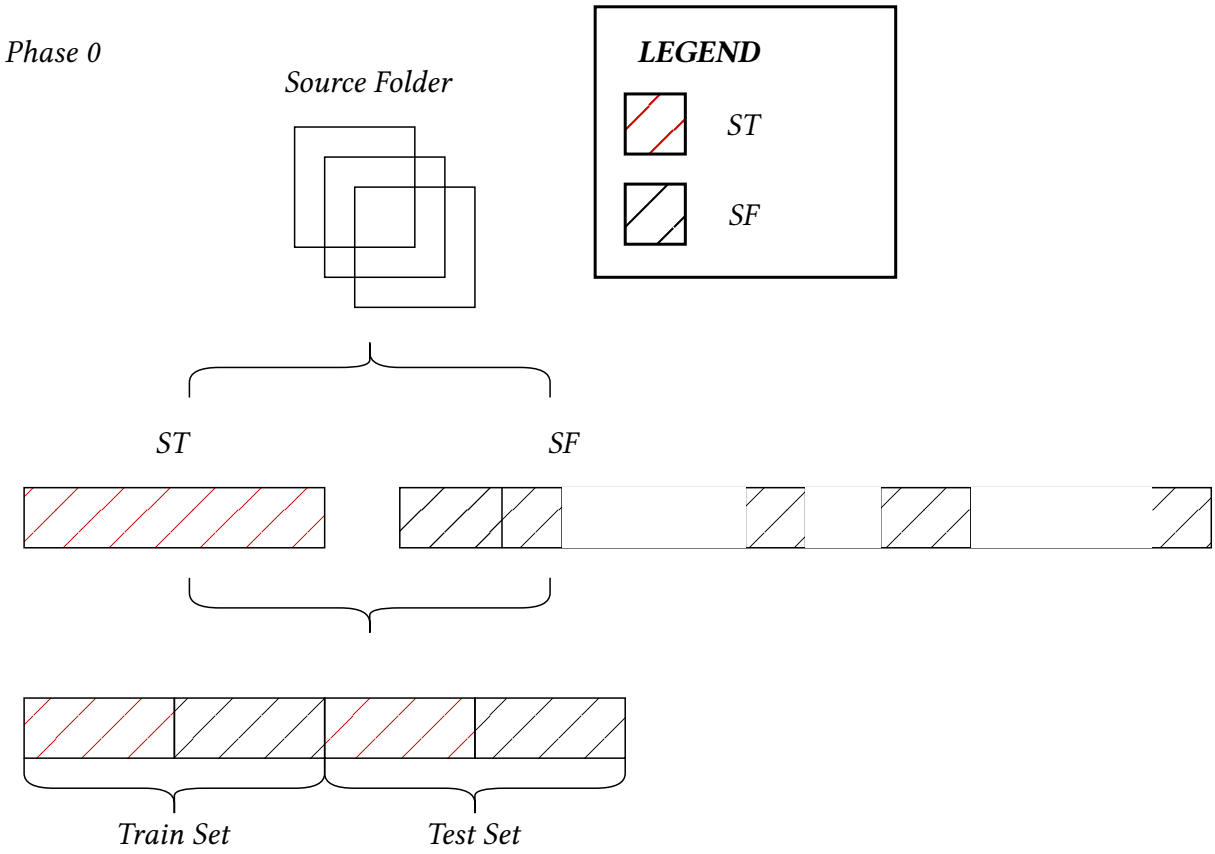


Figure 4.1: Phase 0 Diagram

set, until the number of extracted data SF equals the number of data ST. By equalizing the number of data ST and SF in the dataset, the network remains unbiased towards a particular classifier. The remaining data SF is not thrown away but stored in a variable called dataSFR (Data Source False Remainder). The details of Phase 0 are captured visually in Figure 4.1. There is an additional option to normalize the dataset histogram for both data ST and SF. The normalization option includes scaling and shifting the histogram data. The basic implementation of this program does not apply normalization to the input data, this is because the MATLAB functions for the neural network library include a normalization function in its training algorithm.

4.1.2 Phase 1 - Discrimination Network

Phase 1-3 makes use of the MLPNN categorization capability and divides the data into multiple categories based on the evaluation criteria in the network. Phase 1 is the first phase where data will be categorized into subgroups by an MLPNN. The results of this network feed directly into the next network in Phase 2, and so on. Phase 1 is titled the discrimination network because the network makes use of the Stratified K-fold Cross Validation (SK-Fold) method as a training method for dividing the input dataset into different cross-sections. The specifics of SK-Fold were covered in Section 3.3.3. Phase 1 begins by dividing data ST and SF into K-different folds, aside from the last fold, which stores the remaining data samples that could not fit into a fold. The K-folds store randomly generated index values and references to data ST/SF. Figure 4.2 represents the fold index reference in detail. Additionally, the folds are purely used to reference the ST/SF datasets. Pulling from data ST/SF does not make the input dataset network ready. The network inputs 64 neurons, thus the test variable that enters the MATLAB function train() must be a 64xN matrix, where N is equal to the number of samples in the dataset. The discrimination network is trained separately K-times, this allows the network to train and evaluate each fold individually. It is important to note that for every fold, the network is reinitialized. The network weights and biases are set back to their default states specified in Table 4.1. A train and test set are constructed at the start of each run, where the training set is all folds but one combined together and the test set is created by the remaining fold. Both the test and train set should have equal parts ST and SF data. Additionally, the network accepts variables data, x, t, and y. Data contains the condensed raw data that was extracted from the open files in Phase 0. The variable x is a 64xN matrix, where N is equal to the number of samples in the dataset. Finally, variables t and y are the expected and predicted outputs, respectively. The expected output criteria for the discrimination network are the sample source presence, i.e., whether the sample is labeled ST or SF.

The discrimination neural network uses the MLP Feedforward and Backpropagation architec-

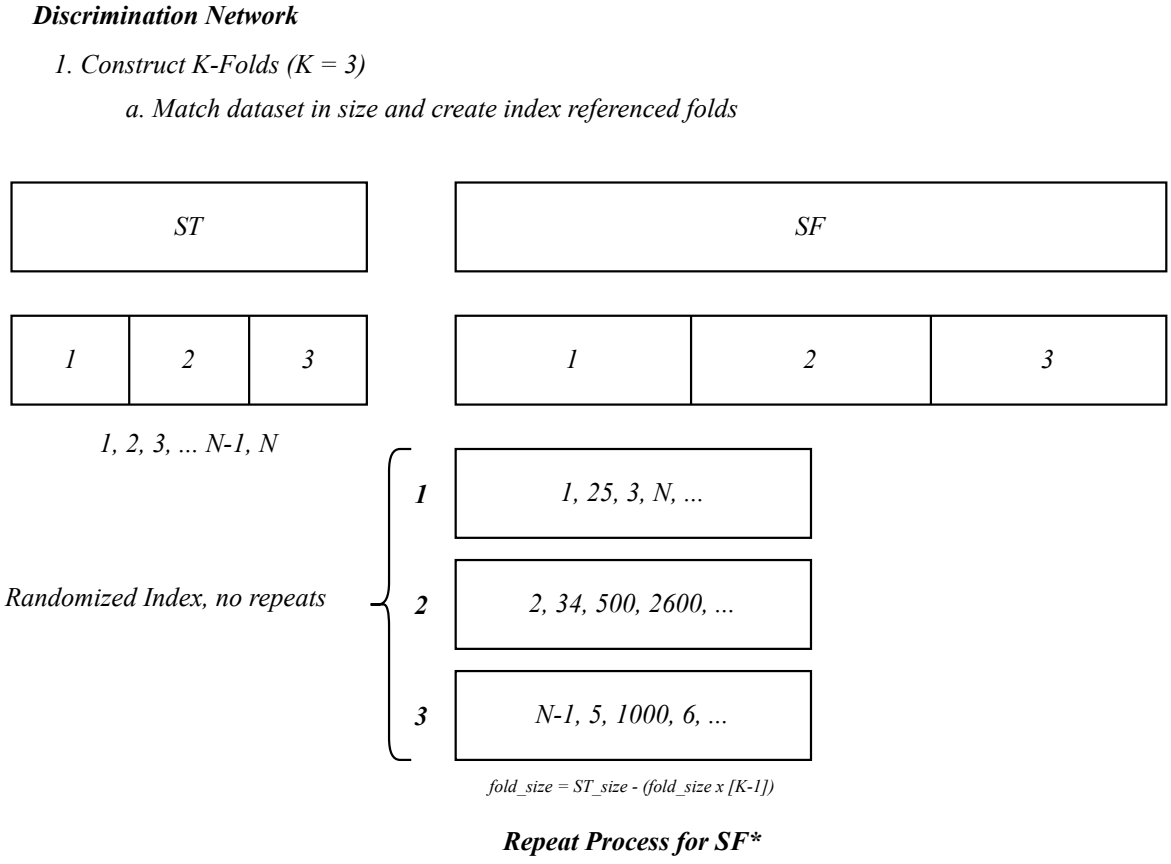


Figure 4.2: Phase 1 Cross Validation Index

ture as its main framework. The MLP network is detailed in Section 2.1 (Chapter 2). The default network parameters are detailed in Table 4.1. Many of the default network parameters defined in Table 4.1 are altered and tested, these test results are detailed in Chapter 5.

The discrimination network seeks to label the predicted output to its network. The main criterion, as previously mentioned, is to determine whether the sample is ST or SF based on its histogram. The supervised learning style allows the program to know whether the viewed sample is ST or SF. The predicted and expected results evaluate a score between 0 and 1, where 0 represents SF and 1 represents ST. The program is far from ideal and produces predicted outputs that

Table 4.1: Default Discrimination Network Parameters

Network Parameters	**Values**
Input Neurons	64
Hidden Neurons	15, 15, 15
Output Neurons	1
Layers	5
Hidden Layers	3
Training Function	Trainscg
Transfer Function	Logsig, logsig, logsig, tansig
Weight Initialization	Gaussian Distribution
Bias Value	0
Threshold LD	0.3
Threshold HD	0.7

are within the range. For example, the expected result for an ST is 1, but the predicted result may be 0.1 or 0.2, the predicted result, in this case, is incorrect. The opposite applies to an SF-predicted value. On an absolute test criterion for any predicted result for ST that is not 1 or an SF that is not 0, the sample is labeled as incorrect. Such criteria are too stringent for the network and unnecessary for this program. Thus, a thresholding method is applied to determine whether a predicted output for a sample is correct or incorrect. Two threshold values are applied to the program. One to evaluate the ST results and another to evaluate the SF results. The thresholds must fit within the expected output criteria, meaning the threshold will be within the range. For example, if the ST threshold is set to 0.7, any samples that score greater than or equal to 0.7 for their predicted result are labeled as a high discrimination (HD) sample. Any ST sample that scores less than 0.7 is labeled as a low discrimination (LD) sample. These labels do not alter the dataset, but simply attach a label to each sample in the dataset because of the discrimination network. An example plot of a test fold that has been run through the discrimination network is provided in Figure 4.3. The output of the discrimination network will result in the dataset containing 4 dominant labels, including STLD, STHD, SFLD, and SFHD. These are variants of ST/SF and LD/HD. These

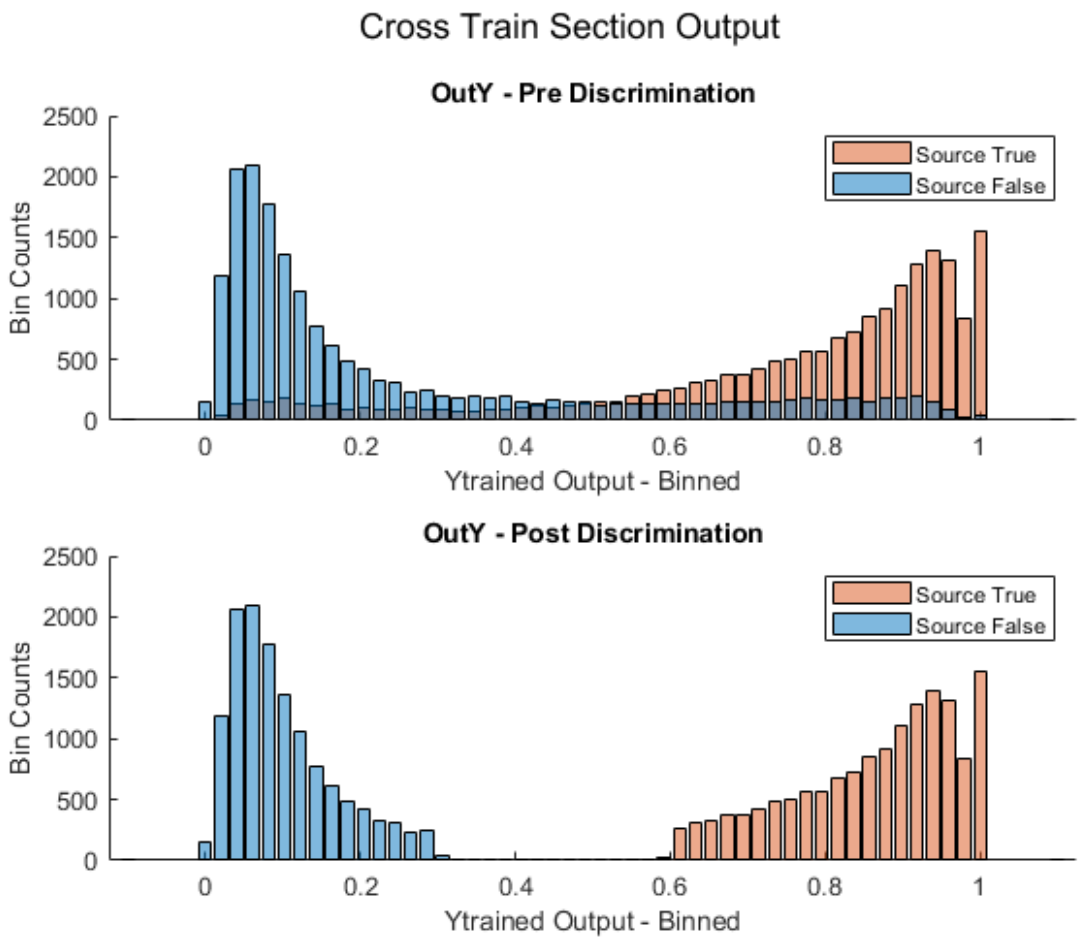


Figure 4.3: Discrimination Network Thresholding

labels are not equal in size and vary greatly depending on the discrimination threshold values. The results of the discrimination network are carried into Phase 2, the confidence network which makes use of the 4 dominant labels. The content of Phase 2 is in Section 4.1.3. To prevent the program from creating an endless number of new datasets, only the index of data ST/SF is kept and stored in the form of an index tree. The index tree is visualized for the discrimination labels in Figure 4.2.

4.1.3 Phase 2 - Confidence Network

Phase 2, known as the confidence network, makes use of the results of the discrimination network and is similar in design to the network in Phase 1. The confidence network differs from the discrimination network in a few key parameters. The output criteria predict a value between 0 and 1 based on the network's ability to discriminate outputs accurately. From the discrimination network, the samples that categorize based on their high discrimination value will result in a 1, whereas a low discrimination value will result in a 0. The confidence network, much like the discrimination network, makes use of the SK-Fold Cross-Validation Method to organize the input data into K-equal or semi-equal folds. From the introduction of the discrimination network, there are now 4 labels to each fold, meaning the dataset for each fold must contain equal parts from each label. Figure 4.4 details the data stratification method used in the confidence network.

When creating the fold, the label with the fewest values is taken as the batch size, and the remaining labels are paired down to the size of the smallest label. This creates a remainder bucket for each label which is stored as a variable with an R indicator after it, such as HDSTR, LDSTR, HDSFR, and LDSFR. The default network parameters of the confidence network are provided in Table 4.3. For each fold, the confidence network is re-initialized and all but one fold is merged together as the train set. The test set is the left-out fold in the dataset. The confidence network, like the discrimination network, uses a 2-way thresholding technique to determine the result of the network and is additionally labeled with a low or high confidence label. The purpose of this network is to determine whether the sample can produce a predicted result in a network that evaluates the sample source presence with low or high confidence. The core difference between the discrimination and confidence networks is that the discrimination network trains based on the true sample source presence, whereas the confidence network trains based on the sample's high classification characteristics.

Confidence Network

1. Construct K-Folds (K = 3)

a. Between the 4 labels (HDST/LDST/HDSF/LDSF) randomly pull data such that all labeled data is of equal size.

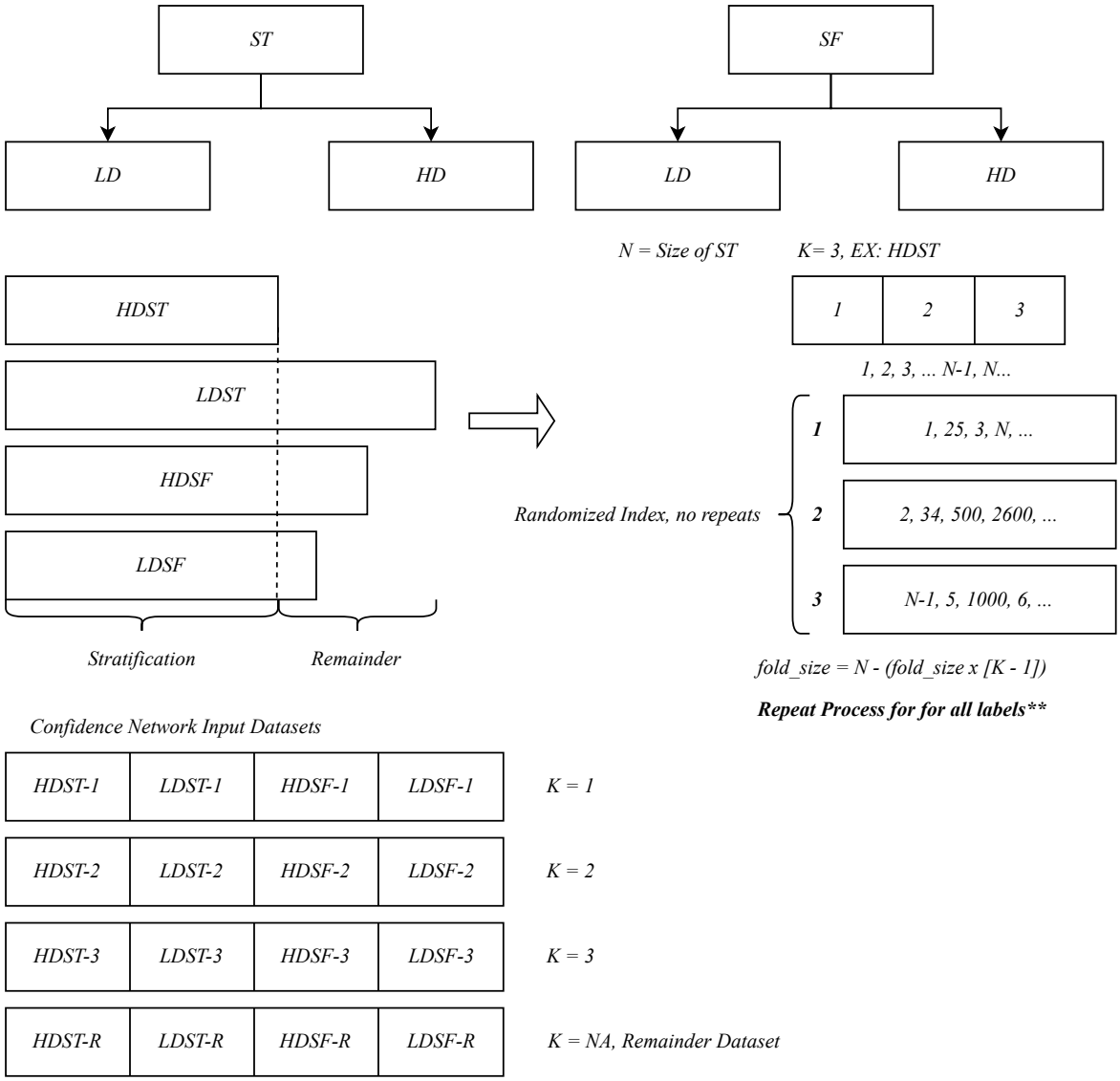


Figure 4.4: Confidence Network Input Data K-Fold Indexing

Table 4.3: Default Confidence Network Parameters

Network Parameters	**Values**
Input Neurons	64
Hidden Neurons	15, 15, 15
Output Neurons	1
Layers	5
Hidden Layers	3
Training Function	Trainscg
Transfer Function	Logsig, logsig, logsig, tansig
Weight Initialization	Gaussian Distribution
Bias Value	0
Threshold LC	0.7
Threshold HC	0.9

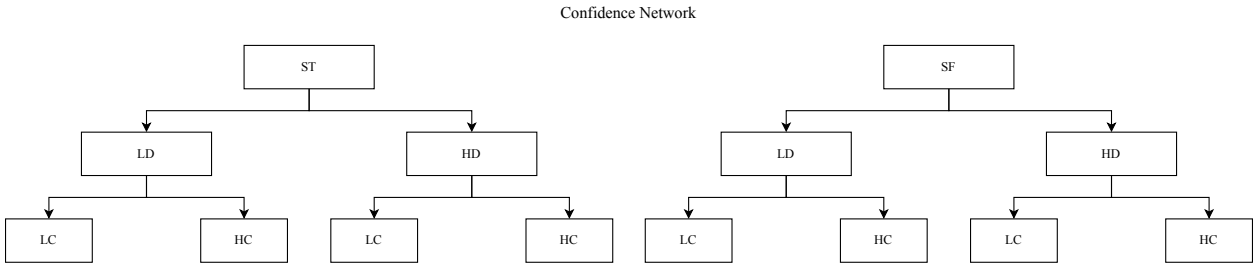


Figure 4.5: Phase 2 Index Tree

The labels produced by the confidence network are an extension of the discrimination network, where the confidence network cannot be run without the discrimination labels pre-existing. The index tree for the confidence network in relation to data ST/SF can be found in Figure 4.5. Ultimately, the confidence network generates 8 new labels to the input dataset which include high or low confidence and are as follows: HCHDST, HCLDST, HCHDSF, HCLDSF, LCHDST, LCLDST, LCHDSF, and LCLDSF.

The confidence network iterates through K-folds and labels each sample with a high or low confidence label. The remainder data set is passed through the network after the last test fold to

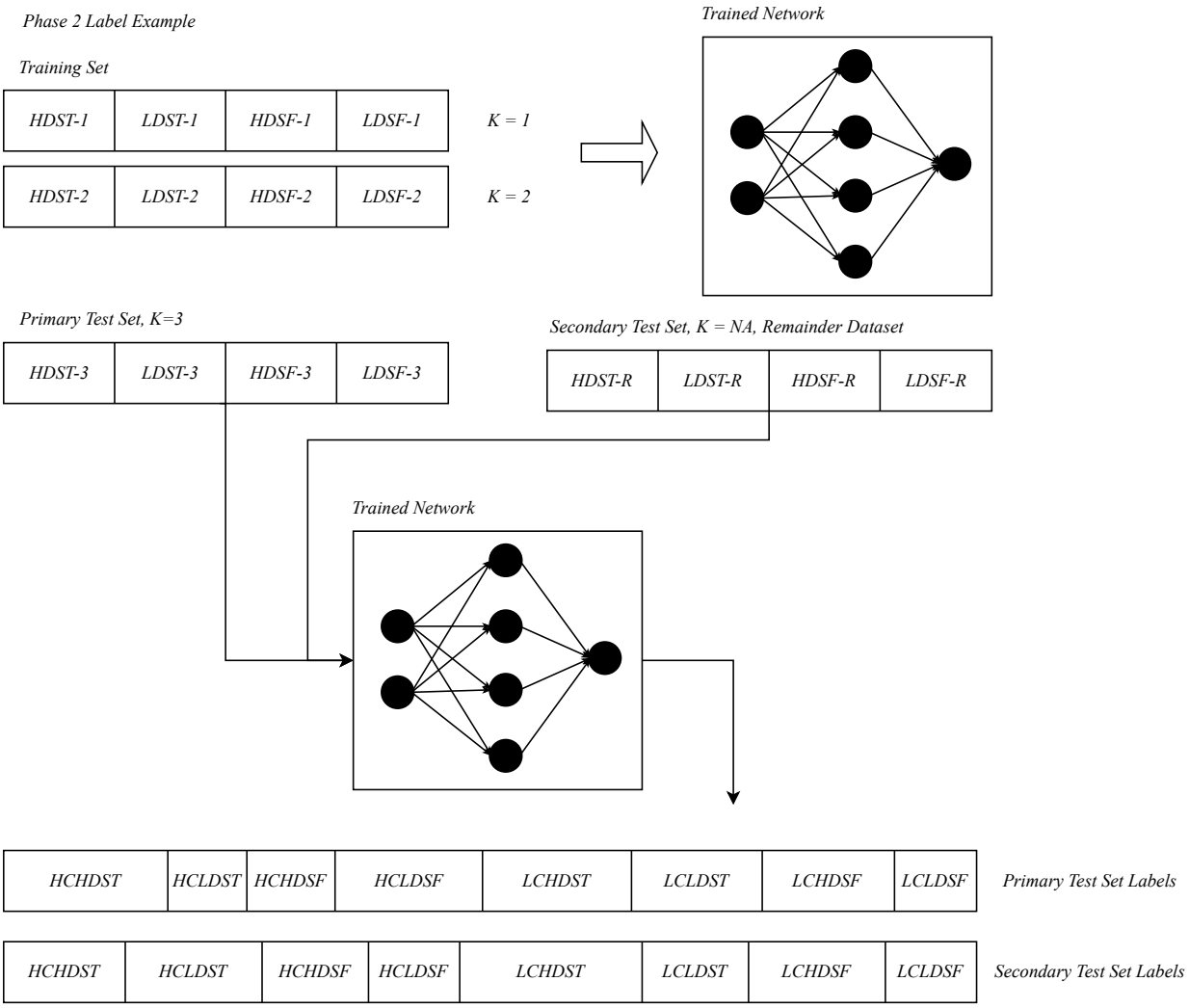


Figure 4.6: Phase 2 Labeling

produce a confidence label for the remainder set. The data flow for phase 2 can be observed in Figure 4.6.

4.1.4 Phase 3 - Target Network

The final and evaluated network in this program is the Phase 3 Target network. In this phase, the network performance is evaluated based on the source presence using the pre-labeled dataset

from the discrimination network, and the confidence network, or it can bypass the previous phases and train on raw data. The network design for Phase 3 is identical to those of Phase 1 and Phase 2. The default network parameters of Phase 3 are provided in Table 4.5.

Table 4.5: Default Target Network Parameters

Network Parameters	**Values**
Input Neurons	64
Hidden Neurons	15, 15, 15
Output Neurons	1
Layers	5
Hidden Layers	3
Training Function	Trainscg
Transfer Function	Logsig, logsig, logsig, tansig
Weight Initialization	Gaussian Distribution
Bias Value	1
Hold Out Train/Test Ratio	80/20 %
Evaluation Threshold Score	0

The purpose of the target network is to train a model that can determine if a sample source presence is true or false. The design of the target network is identical to that of the discrimination network; however, the train and test datasets do not use the SK-Fold Cross-Validation Method. Instead, the hold-out method as detailed in Section 3.3.2, is used as the input data allocation method to the network. The hold-out percentages for the train and test set are detailed in the results in Chapter 5, but as a rule of thumb, the train set is always larger than the test set. For phase 3, there are three datasets; the train set, the test set, and the second test set composed of any remaining data that is excluded from the primary train or test set. The target network has 3 options for network input data. The first option acts as a control group for this project since there is only one network being applied to the sample histograms without any data filtering applied. Additionally, the first option serves as a comparison for improvement or deterioration in the model performance. The second option runs all phases in the program, but phase 2. There

are 4 labels because of the discrimination network, LDST, HDST, LDSF, and HDSF. The target network in option 2 trains solely on the high-discrimination samples from phase 1 and the low-discrimination samples are placed in the second test set. The third and final option implements all three phases. Eight new labels are generated from the confidence network in phase 2 and the target network trains on the high-confidence samples. The low-confidence samples are placed into the second test set. Figure 4.7 visualizes the train/test and remainder datasets used for the target network. The test set and remainder set are equally important as a method for evaluating the performance of the model. The test set includes data like the network training set for any option, however, the characteristics of the remainder set (if applicable) vary greatly from the train and test set. In many cases, the remainder set produces a predicted output that behaves in the opposite manner of the expected output. I.e. if the sample is labeled source true, then the predicted result will be source false. The ideal behavior is for the model to be able to predict the output accurately for both the test and the remainder set.

4.1.4.1 Network Performance Metrics

Phase 3 is the last network in the program and is the only network in the program that is evaluated on its performance. Due to the complex nature of the dataset and program design, the result of the network cannot be assessed solely on one metric. This section covers the different evaluation metrics used to determine whether the network achieved a high level of classification. Table 4.7 describes the performance variables provided in this program.

The target network is an MLP architecture with backpropagation for weight and bias correction. The loss function for the target network is the mean-squared error (MSE), which is one of the most common loss functions used with backpropagation models [16]. The loss function is incorporated into the MATLAB train() function and provides a performance metric in a numerical value ideally close to 0. The lower the MSE value, the better the performance of the network. While the MSE provides some insight into the overall performance and trend of the network, it

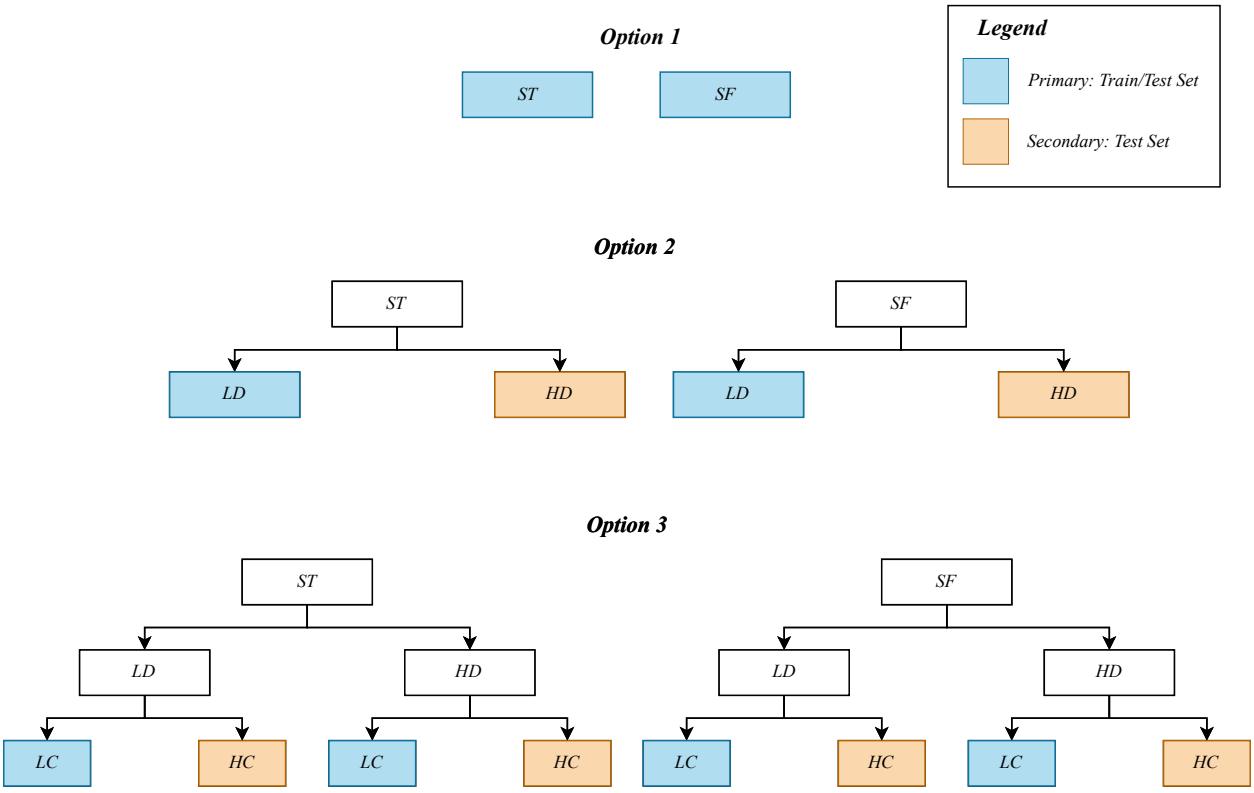


Figure 4.7: Phase 3 Options

Table 4.7: Performance Variables

Variable Acronym	Variable Name
MSE	Mean Squared Error
e_STov	ST Overlap Error
e_SFov	SF Overlap Error
TPR	True Positive Rate
FPR	False Positive Rate
AC	Accuracy

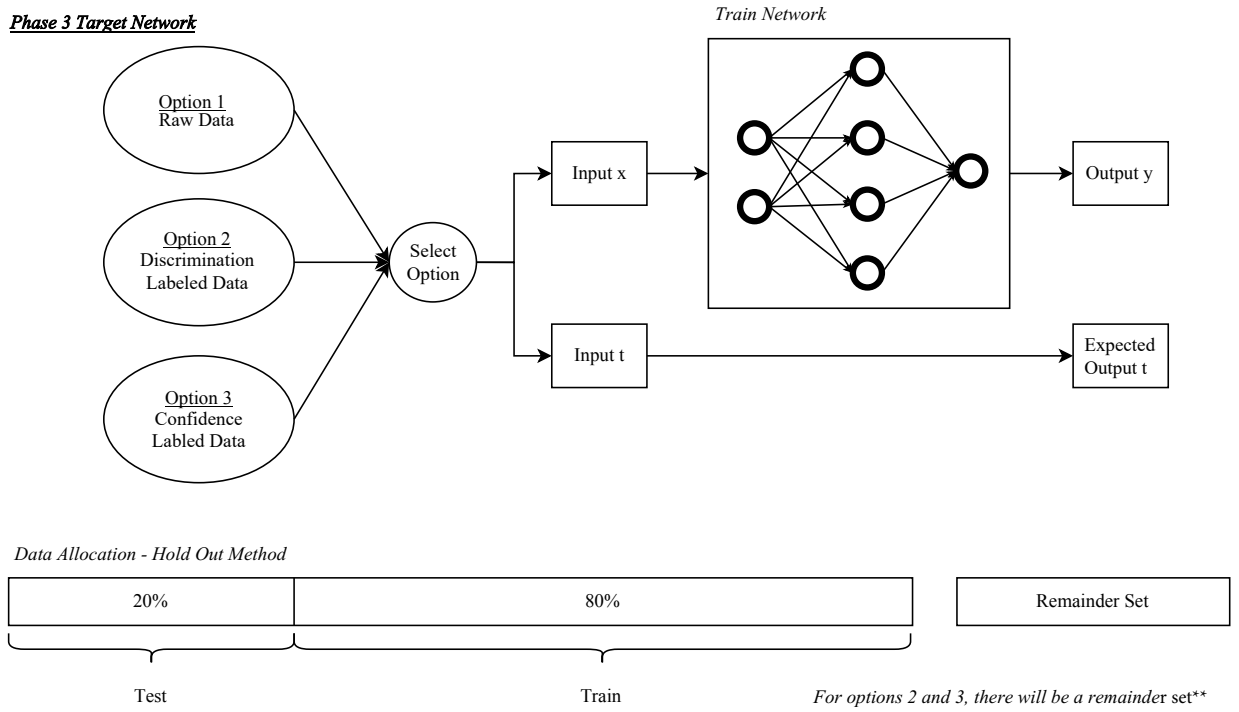


Figure 4.8: Target Network Dataflow

does not completely represent the amount of error experienced in the network results. Another performance variable is the overlap error, which is the amount of error in a predicted label past a specified threshold on a numerical scale. Overlap error is preferred to MSE because it provides a clearer view of the real-time performance of the target classifier.

In an ideal case, the neural network can accurately classify the sample histograms to determine the source presence. The predicted output would be either 0 or 1, which numerically represents ST and SF. However, the network is non-ideal. There are many non-idealities such as background variation, energy conversion error, and internal noise from the sensor. The noise influences the network performance creating predicted results that are no longer the same as the expected results. Classification is still obtainable, where the predicted results will be close to the value of the expected output. The impurities introduced by the system and data collection variation may result in some overlap from predicted results for the two classes as seen in Figure 4.9 [17].

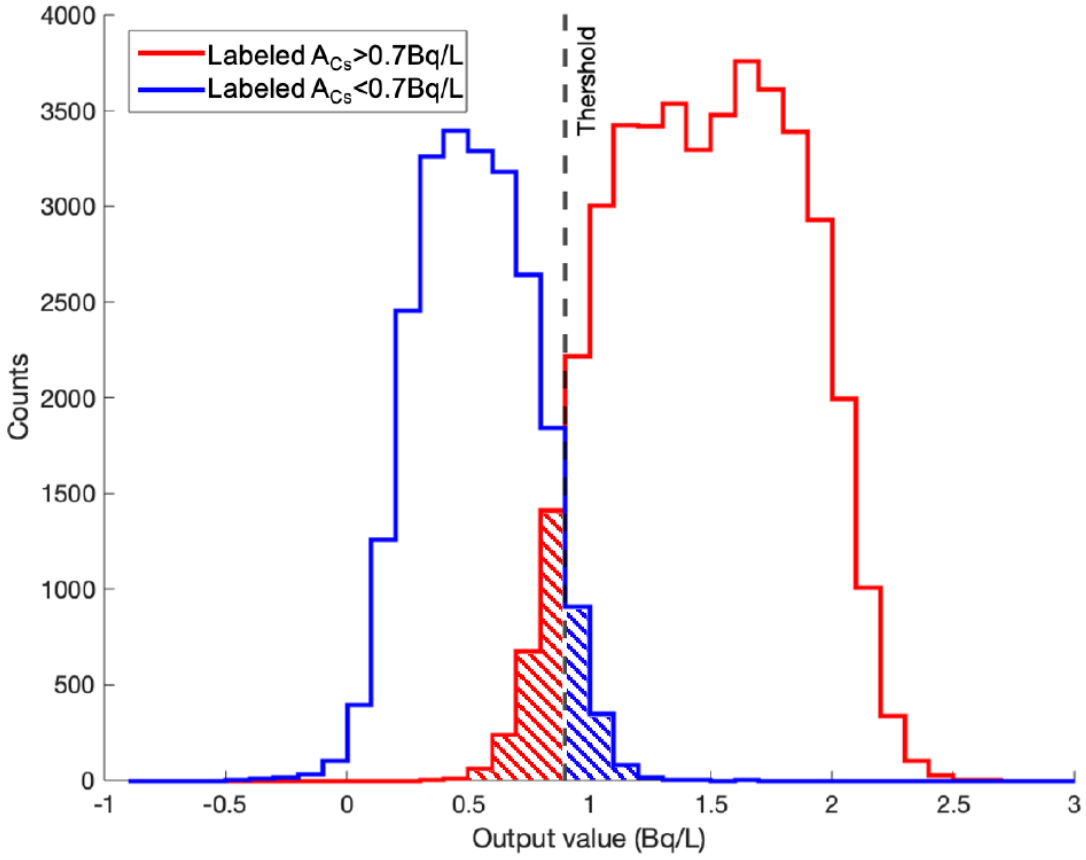


Figure 4.9: Overlap Error

The last performance variable is intended to quantify the accuracy of a network using an automated diagnostic system that compares the classification of the network to an ideal code. The target network performance is evaluated mainly on the comparison of the expected value vs the predicted value, this comparison results in classifying the sample. There are 4 resulting classifications, which are organized in Figure 4.10 [18].

The classifications are the labels on the left of the table and the ideal codes are the labels on top of the table in Figure 4.10. For the ideal codes, the labels are “TRUE” and “FALSE”, when compared to the classifications, these indicate whether the output of the model for a particular class was successful in classifying itself. The samples are known to be ST or SF, before being passed into

		"Gold standard" diagnosis: Positive	"Gold standard" diagnosis: Negative
System diagnosis	Positive	TP (true positive)	FP (false positive)
	Negative	FN (false negative)	TN (true negative)

Figure 4.10: Region of Convergence

the network, the ideal code is then determined from the predicted output of the network. The first decision is a true positive (TP) class, which means that the expected output of the sample is ST, and the predicted output aligns with ST. For the histogram data, the TP decision is sometimes referred to as a HIT, because the network has successfully and accurately identified that a source is present within the proximity of the device. Normally, a HIT in radiation detections leads to further action on the device that requires higher power. Additional measurements or programs may be activated upon a HIT. The second decision is a false positive (FP) class, where the expected output is ST, but the predicted output is SF. In radiation detection, the FP is referred to as a MISS, because the detection device missed a source when a source was known to be present. The same behavior from the first and second classes is repeated for the third and fourth but mirrored in the expected and predicted outputs. The third class is a false negative (FN) and the expected output is ST with a predicted output of SF. And the fourth and last class is a true-false, where the expected output is SF, and the predicted output is SF. The classes are mathematically represented as ratios according to the equations in Equation 3. The performance of the network is good if the TP and TN ratios are high, and the FP and FN ratios are low.

$$TPR = \frac{TP}{TP + FP}, \quad FPR = \frac{FP}{TP + FP}, \quad TNR = \frac{TN}{TN + FN}, \quad FNR = \frac{FN}{TN + FN} \tag{4.1}$$

From Figure 4.9, there are two class labels shown in blue and red. The overlapping shaded regions from the 2 plots make up the FP and FN classes, whereas the unshaded regions are the TP and TN classes. Ignoring the legends, imagine the blue plot is the predicted output for the

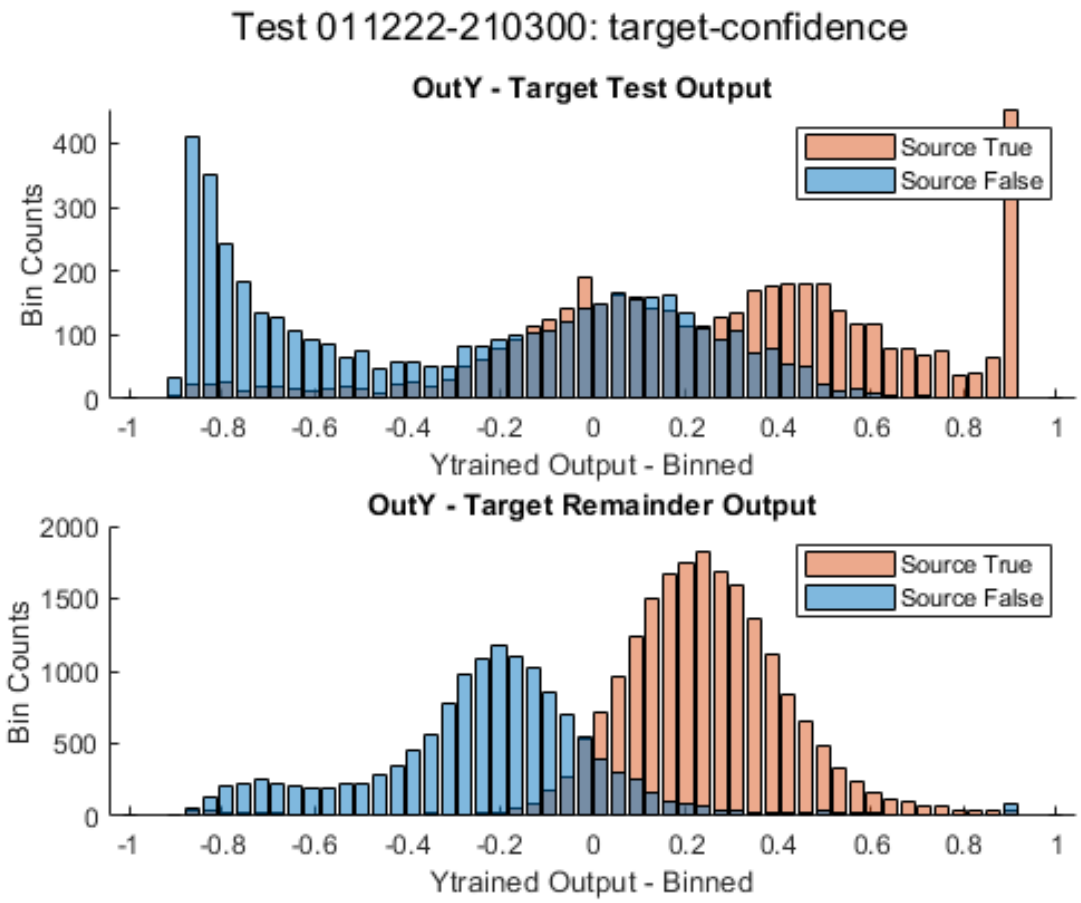


Figure 4.11: Option 3 Target Network Results Overlap

SF samples and the red plot is the predicted output for the ST samples. The shaded red region is the FP region, and the shaded blue region is the FN region. These two regions act as the overlap error and the second performance metric for this network. For the target network, two sets of class ratios were generated. One set for the target test output and the second set for the remainder set output. The test set and the remainder set produce different results from the network. It is necessary to generate two independent sets of class ratios since the two test sets have unique behaviors in their predicted results. Figure 4.11 provides an example of the target network performance plots.

Chapter 5

Testing & Results

This section provides the results of the program design explained in Chapter 4. Section 5.1 details the results achieved using the MLPNN Multi-Phase approach. Different network parameters were tested to achieve the lowest error and highest accuracy values. Unless specified otherwise, the results presented in section 5.1 are generated by the phase 3 target network in the program.

5.1 MLPNN Multi-Phase Results

As mentioned in Chapter 4, there will be two test cases for every test. The primary test case includes test data that is similar to the training data used in the phase 3 target network. The primary dataset is a subset of either phase 1 or 2 networks, if applicable, and matches the same condition in which the training data was generated. The secondary test set only applies when option 2 or 3 of the target network were used. The remainder data is any data that has been left out of the training and test set. The goal of the program is to determine sufficient detection of less than 10% overlap error on either test case. It is expected that the primary test set will perform better than the secondary test set due to its similar characteristics to the training set. The overlap error, accuracy, and true and false positive rates of the network were evaluated in

this section. The test results in this section will be presented and identified by either line number or filename. There is too much data to present in one table for each test result, so all sections will include 4 tables. The first includes the filename and the test parameter. The second is the filename and test results for the primary test set. The third includes the filename and test results for the secondary/remainder test set. Lastly, the fourth table includes any information about the neural network and program parameters that were set before the program was run.

5.1.1 Activation Function

For this test result, the activation function was varied to observe the response of the network. As previously mentioned in Chapter 2, the different activation functions and combinations of functions heavily influence the behavior of the weights in each layer. The model learning is directly influenced because in backpropagation the derivative of the activation function is applied directly to the weights update.

A combination of logsig, purelin, and tangsig was tested on a 5-layer network. The specific configuration of each activation function is detailed in Table 5.1 and the results can be found in Table 5.3. Note, there is no secondary test set in this section because the network results collected for this particular test utilized an older model that did not include the remainder set at the time. The overlap error in this section also follows a slightly different scheme without the absolute function applied to the error value.

While mostly the performance of the network can be evaluated based on the overlap error and accuracy of the output. It is beneficial to evaluate the network based on a visual representation of the output results. Figure **??** includes the plots for all the results outlined in Table 5.3. As we can see, Figure 5.2a appears to have the best performance in overlap error in its second test set compared to all other network results.

Table 5.1: Varying Activation Function

Filename	Activation Function
’011222-161707’	logsig,logsig,logsig,tansig
’011222-163326’	logsig,logsig,logsig,logsig
’011222-164705’	tansig,tansig,tansig,tansig
’011222-174945’	tansig,tansig,tansig,purelin
’011222-203113’	logsig,tansig,tansig,purelin
’011222-205016’	tansig,tansig,purelin,tansig

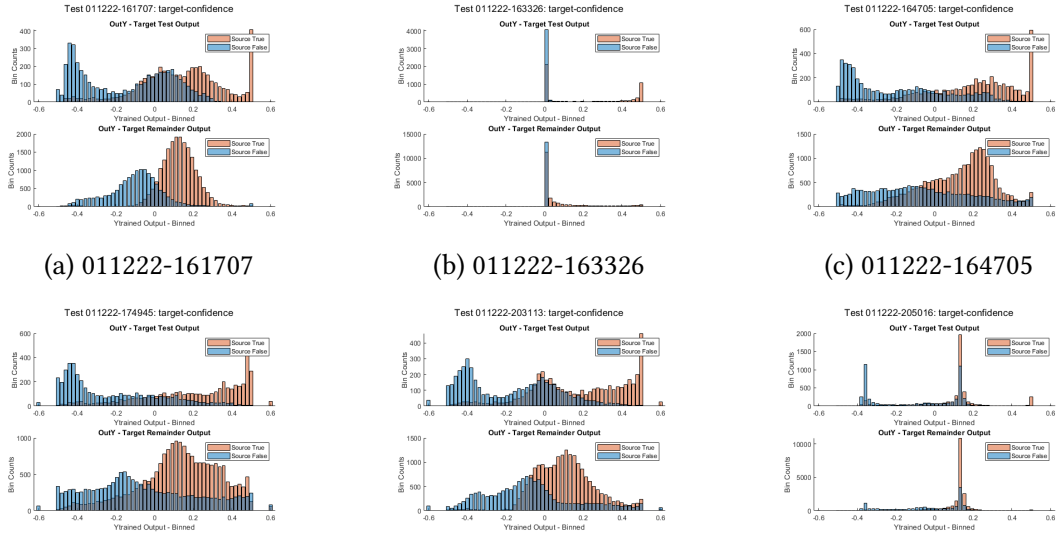
Table 5.3: Activation Function Results

Filename	MSE	ST Overlap Error	SF Overlap Error	STR Overlap Error	SFR Overlap Error
’011222-161707’	0.1741	-22.6874	-31.4117	-7.3561	-18.8449
’011222-163326’	0.2019	48.4217	-7.431	-56.1625	-3.0512
’011222-164705’	0.156	21.2407	-24.463	-15.1757	-46.0007
’011222-174945’	0.1406	-21.0872	-21.1968	-16.8303	-41.6636
’011222-203113’	0.1595	17.317	-33.779	-8.2532	-45.514
’011222-205016’	0.2012	-19.0706	-43.3143	-7.7249	-50.0036

Table 5.5: Activation Function Program Parameters

Target Condition	Hidden Neurons	Bias	Initialize Weights	Discrim/Conf K Folds
target-confidence	[64,15,15,15,1]	0	gaussian	3

HD Threshold	LD Threshold	HC Threshold	LC Threshold
0.7	0.3	0.7	0.9



(a) 011222-161707 (b) 011222-163326 (c) 011222-164705

(d) 011222-174945 (e) 011222-203113 (f) 011222-205016

Figure 5.1: Variable Activation Function Output Plots

5.1.2 Training Function

This section explores the different training functions outlined in section 2.2. The trainscg, trainlm, and traingdx methods were applied according to Table 5.7. The primary and secondary test sets are in Table 5.9 and 5.11, respectively.

Table 5.7: Variable Training Function

Filename	Train Function
'151222-133733'	**'trainscg'**
'141222-114425'	'trainlm'
'141222-120458'	'traingdx'

Table 5.9: Training Function - Primary Results

Filename	ST Overlap Error	SF Overlap Error	True Positive Test	False Positive Test	Accuracy Test
'151222-133733'	**14.4286**	**12.2418**	**0.8134**	**0.1224**	**0.404**
'141222-114425'	13.7946	9.2858	0.8527	0.0929	0.3883
'141222-120458'	12.8482	16.662	0.7653	0.1666	0.4373

Table 5.11: Training Function - Secondary Results

Filename	STR Overlap Error	SFR Overlap Error	True Positive Remainder	False Positive Remainder	Accuracy Remainder
'151222-133733'	**9.0341**	**9.1423**	**0.9381**	**0.0914**	**0.5854**
'141222-114425'	29.4142	21.6133	0.8327	0.2161	0.5118
'141222-120458'	12.4638	31.4290	0.8093	0.3143	0.6530

Table 5.13: Training Function Program Parameters

Hidden Neurons	Transfer Function	Bias	Initialize Weights	Discrim/Conf K Folds
[64,15,1]	logsig,tansig	0	gaussian	3

HD Threshold	LD Threshold	HC Threshold	LC Threshold
0.7	0.3	0.7	0.9

5.1.3 Network Layers

Another explored parameter is the effects of layer network depth. In this section, the results are influenced only by the number of hidden layers. The number of hidden neurons is controlled in this test by setting hidden neurons to be of equal size for every layer. Table 5.17 and 5.19 present the results of the network for hidden layers 1, 2, 4, 6, and 12. As a general trend, the network performance for SF Overlap performs worse as the number of hidden layers increases. Other test parameters such as ST Overlap, Accuracy, TP, or FP error seem unaffected by the increase in network layers.

Table 5.15: Variable Network Layer

Filename	Hidden Neurons
’141222-120458’	[64,15,1]
’141222-124807’	[64,15,15,1]
’141222-125426’	[64,15,15,15,15,1]
’141222-130646’	[64,15,15,15,15,15,15,1]
’141222-131952’	1x12 double

Table 5.17: Network Layers - Primary Results

Filename	ST Overlap Error	SF Overlap Error	True Positive Test	False Positive Test	Accuracy Test
’141222-120458’	12.8482	**16.662**	0.7653	0.1666	0.4373
’141222-124807’	12.8304	**18.3321**	0.7478	0.1833	0.4477
’141222-125426’	13.4911	**16.8402**	0.7621	0.1684	0.436
’141222-130646’	8.9196	**25.6694**	0.6887	0.2567	0.5079
’141222-131952’	11.9018	**21.3272**	0.7203	0.2133	0.4697

Table 5.19: Network Layers - Secondary Results

Filename	STR Overlap Error	SFR Overlap Error	True Positive Remainder	False Positive Remainder	Accuracy Remainder
’141222-120458’	12.4638	**31.429**	0.8093	0.3143	0.653
’141222-124807’	11.915	**36.4608**	0.7864	0.3646	0.6763
’141222-125426’	11.7677	**29.9118**	0.818	0.2991	0.6512
’141222-130646’	5.4875	**50.0077**	0.7422	0.5001	0.7688
’141222-131952’	9.4609	**44.5348**	0.7559	0.4453	0.7231

Table 5.21: Network Layers Program Parameters

Train Function	Transfer Function	Bias	Initialize Weights	Discrim/Conf K Folds
'traingdx'	logsig, logsig, ..., tansig	0	gaussian	3
HD Threshold	**LD Threshold**		**HC Threshold**	**LC Threshold**
0.7	0.3		0.7	0.9

5.1.4 Hidden Neurons and Layers

Another network parameter that was explored was the effect of not only varying the network depth but varying the number of hidden neurons per layer. There are different tradeoffs to creating a network with an infinite number of neurons and layers. First, is the computation time. If the network is so massive, then there is a physical limitation on the CPU's real-time computation of the network. The second is the complexity of the network to induce overfitting. Network complexity does not always ensure the solution to a classification problem. The larger the network is the more complex the computations become and more demand is required of the computation ability of both the hardware and the software of the device. Hidden neurons and layers benefit the most in improving the network by tuning the network parameters to the applied problem. This section seeks to find the tuning of hidden neurons to layers in order to optimize the network performance. Table 5.23 defines the different hidden neuron and layer configurations for the tests. These were chosen arbitrarily, but smaller in size due to the physical limitation of my CPU and run time. Table 5.25 and 5.27 cover the results of the variation in network parameters. Interestingly the network performs better on the secondary test set than the primary for test case "171222-151455".

Table 5.23: Variable Hidden Neurons/Layers

Filename	NN_Hidden Neurons
’151222-145012’	[64,30,30,1]
’151222-150809’	[64,30,15,1]
’171222-140003’	[64,30,10,1]
’171222-140957’	[64,10,30,1]
’171222-142022’	[64,30,30,30,1]
’171222-143715’	[64,30,30,10,1]
’171222-150746’	[64,30,10,10,1]
’171222-151455’	**[64,10,10,10,1]**
’171222-151944’	[64,30,15,5,1]
’171222-152915’	[64,15,15,5,1]
’171222-154503’	[64,15,10,5,1]

Table 5.25: Hidden Neuron and Layer - Primary Results

Filename	ST OvError	SF OvError	True Positive Test	False Positive Test	Accuracy Test
’151222-145012’	15.1071	12.1583	0.8132	0.1216	0.4009
’151222-150809’	14.1429	13.3886	0.7999	0.1339	0.4122
’171222-140003’	20.5438	13.7422	0.7343	0.1374	0.3499
’171222-140957’	20.0841	13.4786	0.7392	0.1348	0.3496
’171222-142022’	20.8328	13.4283	0.7381	0.1343	0.3469
’171222-143715’	21.7785	13.0517	0.7412	0.1305	0.3413
’171222-150746’	22.1201	12.6185	0.7468	0.1262	0.3372
’171222-151455’	**21.3057**	**13.4158**	**0.7371**	**0.1342**	**0.3453**
’171222-151944’	23.4861	12.8508	0.74	0.1285	0.3344
’171222-152915’	21.4764	13.5539	0.7347	0.1355	0.3456
’171222-154503’	20.7146	13.3216	0.7399	0.1332	0.3465

Table 5.27: Hidden Neuron and Layer - Secondary Results

Filename	STR OvError	SFR OvError	True Positive Remainder	False Positive Remainder	Accuracy Remainder
’151222-145012’	10.533	11.8826	0.9198	0.1188	0.5872
’151222-150809’	12.098	18.2149	0.8803	0.1821	0.6029
’171222-140003’	10.4663	13.1496	0.9145	0.1315	0.5981
’171222-140957’	7.9862	12.3749	0.9211	0.1237	0.6103
’171222-142022’	10.2305	14.3562	0.9076	0.1436	0.6043
’171222-143715’	11.8592	11.9978	0.9202	0.12	0.5851
’171222-150746’	10.4925	10.421	0.931	0.1042	0.5874
’171222-151455’	**7.9469**	**7.7266**	**0.9493**	**0.0773**	**0.5924**
’171222-151944’	11.0296	9.6051	0.9357	0.0961	0.5809
’171222-152915’	9.104	12.0389	0.9222	0.1204	0.6021
’171222-154503’	8.2351	12.0527	0.9228	0.1205	0.6075

Table 5.29: Hidden Neuron and Layer Program Parameters

Train Function	Transfer Function	Bias	Initialize Weights	Discrim/Conf K Folds
’trainscg’	logsig, … ,tansig	0	gaussian	3
HD Threshold	**LD Threshold**		**HC Threshold**	**LC Threshold**
0.7	0.3		0.7	0.9

5.1.5 Phase Training

This section goes into detail about the phase training results. In Phase 3, the target network has 3 options to choose from to select its input data. The first is taking in the raw data, where no data handling or data labeling has been applied. The second option includes selecting portions of the data to train based on its discrimination network label. The third option is selection another subset of data to train based on its confident network label. Table 5.31 provides the filename and associated target network input data condition. Based on the results in Table 5.33 and 5.35, it is fairly easy to determine that the phase 3 network using the results from phase 2 or the confidence label provides the best network performance on both the primary and secondary test set. Either overlap error averages 12%, the TP and Accuracy are above 0.87 or roughly 87% in accuracy. Additionally, the false positive rate is extremely low at around 0.12 or 12%. The visual differences between the three plots produced by each different input option also characterize the successful network performance when comparing Figures **??** - **??**.

Table 5.31: Phase Training Variable Options

Filename	**Target Network Conditions**
’250623-212921’	Tar - NA
’250623-213903’	Tar - D
’250623-220112’	**Tar - C**

Table 5.33: Phase Training Option - Primary Result

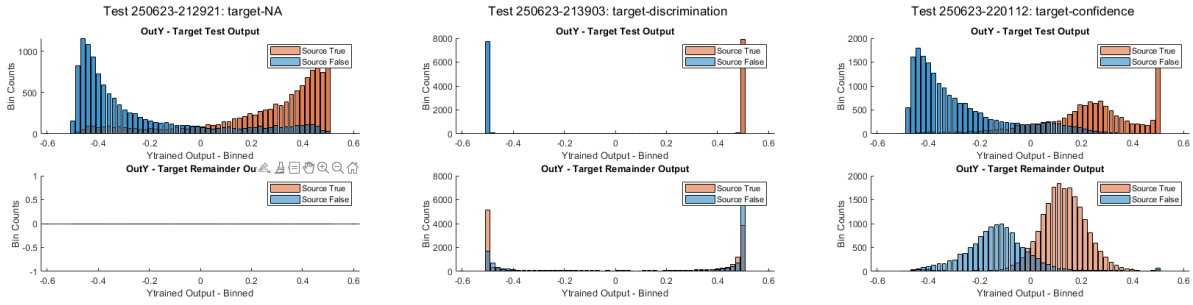
Filename	**ST OvError**	**SF OvError**	**True Positive Test**	**False Positive Test**	**Accuracy Test**
’250623-212921’	14.8683	17.6748	0.8513	0.1767	0.8373
’250623-213903’	0.1233	0.2264	0.9988	0.0023	0.9983
’250623-220112’	**12.8633**	**11.9898**	**0.8714**	**0.1199**	**0.8767**

Table 5.35: Phase Training Option - Secondary Results

Filename	STR OvError	SFR OvError	True Positive Remainder	False Positive Remainder	Accuracy Remainder
’250623-212921’	-	-	-	-	-
’250623-213903’	47.4922	68.1316	0.5251	0.6813	0.4213
’250623-220112’	**7.9811**	**12.603**	**0.9202**	**0.126**	**0.9023**

Table 5.37: Phase Training Option Program Parameters

Train Function	Hidden Neurons	Transfer Function	Bias	Initialize Weight
trainscg’	[64,15,15,15,1]	logsig,logsig,logsig,tansig	0	gaussian
Discrim/Conf K Folds	**HD Threshold**	**LD Threshold**	**HC Threshold**	**LC Threshold**
3	0.7	0.3	0.7	0.9



(a) NA (b) Discrimination Labeled (c) Confidence Labeled

Figure 5.2: Target Network Input Data Option Plots

Chapter 6

Conclusion

This thesis explored the use of neural networks to improve the generalization capabilities of the onboard neural processing unit on UNL's nuclear radiation detection chip "Pingora". The goal of this project was to achieve an overlap error below 10% on both ST and SF data using the multi-phase MLPNN approach. Two test sets were used to evaluate the performance of the model, a primary and a secondary test set. The characteristics of the primary test set are similar to the training set and therefore, it is expected for the test set to perform well on the model. The secondary test set includes the remainder data that did not fit into the labeled data training or test set due to stratification. Additionally, if the labeled data from Phase 2 was used as an input option to the target network in Phase 3, then the secondary test set would include the worst-case data. The remainder data is considered the worst-case because of its difficulty to classify, this characteristic would have been identified by the labeling process used in Phases 1 and 2. Achieving a low error rate on the secondary test set shows that the model's generalization ability has improved and the network is therefore more robust. Success was found in the target network using option 3 inputs from the confidence network in phase 2. The result of this model was 12.86/11.99% ST/SF overlap error and the secondary test result was 7.98/12.60% ST/SF overlap error. Overall, the model was able to achieve an average of 12% error across all datasets.

There are a few considerations that should be accounted for as to the difficulty of achieving the project goal when using the APL dataset. The key point is to recognize the difference in application between APL and UNL's sensors. From section 3.1.2, we know that the APL sensor was intended to detect as a fly-by detector. This application implies that the device will experience a high variation in its background, and sources are introduced to the sensor due to proximity. In contrast, the Pingora is intended to be used as a stationary sensor where the device waits for events to occur in a neutral and relatively low background variation environment. It is important to acknowledge that the APL dataset is difficult to classify. Simply put, there is a high variation in what histogram falls under source true and source false; this was, in fact, the motivation for the multi-phase filtering process in the first place. The dataset includes samples that are labeled source true but may not include any identifiable characteristics in the sample histogram. The same can be said about the source false. There is a high variation in the source false data samples, where the gross gamma counts vary between roughly 200-500. The source true samples range around 400-2000 gross gamma counts, though the average is generally around 700. The assumption for the variation in source true data samples may be due to the proximity to the sensor. For example, the sensor may have come within range of the sample to determine that a source should be present, but it may have also been too far for the sensor to collect any identifiable data. This discrepancy in a sample's source presence label and the content of the sample histogram is another consideration for the project's difficulty.

6.1 Future Work

The multi-phase MLPNN program was ultimately able to achieve a 12% error on average across all datasets. This achievement is substantial. There are still many parameters from the APL dataset that were not extensively explored, and the multi-phase MLPNN approach showed promise in further improving its model performance. Future work may include revisiting the alternate vari-

ables provided by the APL dataset. Building off the foundation of this program design, it would be interesting to see how UNL's dataset, which has substantially less background variation would perform. I would propose that if UNL's dataset were to perform well under the data filtering method I have proposed in the multi-phase MLPNN design, then future work should reconsider using the APL dataset with another that is more suited for Pingora's application.

Appendix A

DTRA ADR Open File Format v20191203

LINE	Name	Description
1	record	record number
2	detector	The detector name
3	utc-time	The timestamp in human-readable time (UTC) format: YYYMMDDHHMMSS
4	timestamp	The timestamp in milliseconds from Jan 1, 1970 (unix time)
5	azimuth	[decimal degrees] The azimuth between the direction of motion and the source as determined by the detector (0 if detectors do not provide this information)
6	azimuth-uncertainty	[decimal degrees] The uncertainty in the azimuth as determined by the detector (180 if detectors do not provide this information)

7	gc0	gross count for gamma element 0 (integer).
8	glt0	The live time for gamma element 0 in milliseconds (integer)
9	grt0	The real time for gamma element 0 in milliseconds (integer)
10	spectrum-lt0	The spectrum live time in milliseconds for the spectrum (usually the same as the glt0, unless spectra are accumulated separately)
11	spectrum-rt0	The spectrum real time in milliseconds for the spectrum (usually the same as the glt0, unless spectra are accumulated separately)
12	spectrum-channels0	The spectrum for gamma element 0 (comma delimited floating point or integer values)
13	<Repeat gamma elements here for elements 1,2,3>	
14	nc0	The neutron gross counts
15	nlt0	The neutron live time (set to real-time if the neutron counter does not report live time)
16	nrt0	The neutron real-time

17	<Repeat neutron elements here for elements NA>	
18	is-in-zone	TRUE if the sample has taken inside the test zone, FALSE outside the test zone
19	is-closest-approach	TRUE if this sample was estimated (by location) to be the point of closest approach to the source, FALSE otherwise or if the quantity is not calculated
20	is-source-present	TRUE during times when a source configuration is present in the test area and FALSE otherwise
21	source-id	The source configuration name (or "no source"). Field may be an empty tab when no source configuration is in use
22	source-offset	[ft] The offset distance between the detector's distance of closest approach and the source, or 0 if "no source". Field may be an empty tab when no source configuration is in use.
23	latitude	[decimal degrees] The current latitude
24	longitude	[decimal degrees] The current longitude
25	distance-to-doca	[ft] The distance to the point of closest approach. Field is set to -1 when unspecified.

26	is-active	TRUE if data were within experimental "live" conditions, FALSE otherwise. This is used to eliminate times when the sensors may have been inadvertently exposed to a test source or when detectors were being initialized.
27	heading	[North, South, East, West] indicates the direction of sensor movement; generally North or South for data obtained in the APL test zone. Field is an empty tab when outside of the test zone.

Bibliography

[1] M. Reinhard, D. Prokopovich, H. Van der Gaast, and D. Hill, "Detection of illicit nuclear materials masked with other gamma-ray emitters," in *2006 IEEE Nuclear Science Symposium Conference Record*, vol. 1, 2006, pp. 270–272.

[2] S. Nawaz, M. Hussain, S. Watson, N. Trigoni, and P. N. Green, "An underwater robotic network for monitoring nuclear waste storage pools," in *Sensor Systems and Software*, S. Hailes, S. Sicari, and G. Roussos, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 236–255.

[3] T. Dayalu and M. Krishna, "Simple nuclear medical techniques," in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society.* IEEE, 1988, pp. 1865–1866.

[4] G. G. Eichholz, *Principles of nuclear radiation detection.* CRC Press, 2018.

[5] A. Belyaev and D. A. Ross, *The Basics of Nuclear and Particle Physics.* Springer, 2021.

[6] J. M. Ghawaly Jr, "A datacentric algorithm for gamma-ray radiation anomaly detection in unknown background environments," p. 6, 2020.

[7] A. Belyaev and D. A. Ross, *The Basics of Nuclear and Particle Physics.* Springer, 2021.

[8] L. Cerrito *et al.*, "Radiation and detectors," *Cambridge University*, p. 33, 2017.

[9] S. Galib, P. Bhowmik, A. Avachat, and H. Lee, "A comparative study of machine learning methods for automated identification of radioisotopes using nai gamma-ray spectra," *Nuclear Engineering and Technology*, vol. 53, no. 12, pp. 4072–4079, 2021.

[10] G. Gilmore, *Practical gamma-ray spectroscopy*. John Wiley & Sons, 2008.

[11] S. Murray, S. Balkir, and M. Hoffman, "Cmos radioactive isotope identification with multichannel analyzer and embedded neural network," M.S. Thesis, University of Nebraska-Lincoln, 2018.

[12] S. Murray, "Mcu user guide for pingora 2," Aug 2021.

[13] T. M. Inc., "Matlab version: 9.12.0 (r2022a)," Natick, Massachusetts, United States, 2022. [Online]. Available: https://www.mathworks.com

[14] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, vol. 14, no. 2. Montreal, Canada, 1995, p. 1138.

[15] D. Berrar, "Cross-validation." p. 3, 2019.

[16] R. Eberhart and R. Dobbins, "Neural network performance metrics for biomedical applications," in *[1990] Proceedings. Third Annual IEEE Symposium on Computer-Based Medical Systems*, 1990, pp. 283–284.

[17] W. Dai, Z. Zeng, D. Dou, H. Ma, J. Cheng, J. Li, and H. Zhang, "Marine radioisotope gamma-ray spectrum analysis method based on geant4 simulation and MLP neural network," *Journal of Instrumentation*, vol. 16, no. 06, p. P06030, jun 2021. [Online]. Available: https://doi.org/10.1088%2F1748-0221%2F16%2F06%2Fp06030

[18] R. Eberhart and R. Dobbins, "Neural network performance metrics for biomedical applications," in *[1990] Proceedings. Third Annual IEEE Symposium on Computer-Based Medical Systems*, 1990, pp. 284–287.