

Neural-net-based imager offset estimation in fieldable associated particle imaging

A Dissertation Presented for the
Doctor of Philosophy
Degree

The University of Tennessee, Knoxville

Justin Ryan Powers-Luhn

May 2021

Copyright © by Justin Ryan Powers-Luhn, 2021
All Rights Reserved.

Dedicated to people in up to their chins everywhere

Acknowledgments

I would first like to thank every one of the ORNL scientists in the Radiation Detection and Imaging group, especially Matthew Blackston, Zane Bell, Andrew Nicholson, and Gomez Wright. Each of them provided their knowledge and time unstintingly, even when I inevitably needed things explained just one time more.

My family has shown me more patience than I deserve as I begin the third career of my adult life. I always appreciate their love and support. Don't worry, Mom, Dad, and Britt—I don't really understand this stuff either. Don't tell anyone.

The expertise of my committee, especially Prof. Jason Hayward, was valuable at every stage. It seems that being a professor at a research institution is not unlike a duck in a pond—apparent serenity above the waterline belies paddling-like-hell below it. I will likely never be called upon to understand the multitude of demands on my committee's time, but I was always grateful for the time they could give me.

My ORNL mentor Paul Hausladen has forgotten more about neutron imaging than I will ever know. This makes it all the scarier when he expands his knowledge further on a daily basis. He also has the dubious distinction of being probably the most grounded physicist I have ever met. His confidence in his expertise, excellent intuition, and intelligent skepticism are some of my aimpoints as I move into the next stage of my professional life.

Ben, Mitch, Shane, Darby, Emily, and Rachel: collectively you have 63 degrees representing 278 years of graduate educational experience and you *still* couldn't talk me out of going back to school. Your advice and friendship have been a steadying hand; it was much easier knowing you all had walked this path before (or in some cases with) me.

Last but not least: Liz. You're the love of my life and I can't wait to stop doing this on hard mode.

Abstract

Associated particle imaging with deuterium-tritium generators has been demonstrated to be extremely useful in laboratory settings. The Oak Ridge National Laboratory Nuclear Materials Identification System is one such system. A portable imaging system with similar capabilities could be of value to non-destructive analysis of potentially hazardous items. A neural network modeled after the ResNet architecture was trained to predict the position of the detector array relative to the neutron generator and object being imaged. The network was trained using an MCNP simulation of the NMIS system with the neutron array offset in two position and one rotation dimensions. The final network was accurate to within 0.49 cm, 0.66 cm, and 0.66° for axial, lateral, and horizontal plane rotations, respectively. Images reconstructed with this predicted values are qualitatively comparable to the original object and significantly improved when compared to images without these corrections applied.

Table of Contents

1	Introduction	1
1.1	Other nuclear imaging modes	6
1.2	Angular dependence	8
1.3	Image reconstruction	9
1.4	Neural Networks	11
1.5	Motivation	16
2	Review of Prior Work and Statement of Original Contributions	17
2.1	Nuclear Materials Identification System (NMIS)	17
2.2	Computed tomography	19
2.3	Original contributions	22
3	Description of Associated Particle Imaging System	23
3.1	Detector system hardware components	23
3.2	Time-alignment of measured data	27
3.3	Impact of timing accuracy in alpha and neutron detectors	31
3.4	Computed tomography and required accuracy	34
4	Description of Simulation	42
4.1	Design of Experiment	42
4.2	MCNP Modeling	45
4.3	Angular emission distribution	47
4.4	Quasi-curved array	55

4.5	Differences between simulation and reality	55
5	Neural Network Analysis	57
5.1	Data pre-processing	57
5.2	Learning rate optimization	57
6	Results	62
6.1	Error in prediction	62
6.2	Analysis of errors	62
6.3	Measurement time sensitivity	65
6.4	Tomographic reconstruction	68
7	Conclusions and future work	78
	Bibliography	81
	Appendices	89
A	Example of rendered MCNP input deck	90
B	Conversion from MCNP output to image	97
B.1	Library code	97
B.2	Command line tool	101
C	Neural network specification and training	106
D	Python script to align detectors to the same time scale	112
E	An overview of neural network architectures examined in this work	116
	Vita	118

List of Figures

1.1	Possible outcomes for a neutron generated in the associated particle imaging system. Additional neutron sources (e.g. environmental background or fission) are not represented.	3
1.2	Possible imaging system interactions for neutrons and photons. The relationship that carries the transmission image information is emphasized in blue; other processes may represent noise that must be accounted for. As the NMIS system examined in this work produces few detectable photons in the absence of a scattering option, these are not included in this figure. Other API systems produce photons in their generator component and have more complicated interactions.	4
1.3	Distribution of neutron detector counts correlated with alpha detection events in APNIS, a related Oak Ridge API system. The gray bands represent photons produced in the generator or imaged object; the red band identifies neutrons that pass through the object without interacting (transmitted neutrons), and the green/orange band identifies neutrons produced by scattering or fission in the object [3].	7
1.4	Regions of applicability for neutron and X-ray imaging. Each can be usefully employed to image up to ten mean free paths of material, but what defines that thickness is different for each particle [3].	7
1.5	Distribution of neutron scatter angles for elastic scatter off of depleted uranium, iron, carbon, and hydrogen nuclei. Data from [5].	10
1.6	Each layer of the convolutional neural network combines features from the previous layer to compose more complex features [8].	13

1.7	ResNet architecture (right) compared with previous architectures. The skip connections are represented as curved arrows and allow each set of layers to operate on the residuals of the previous block [10].	15
3.1	Position of alpha detectors relative to the tritiated target in the ING-27 generator.	24
3.2	Illustration of the operation of a constant fraction discriminator (CFD). [29]	26
3.3	Coincident counts between each pairing of alpha and neutron detectors as a function of time lag. Time lag is on the vertical axis while the each column represents the pairing of one alpha and one neutron pixel. Each of the fourteen contiguous shapes represents several neutron pixels in coincidence with one alpha pixel.	28
3.4	Calculated offsets for each detector with standard deviation from five void measurements.	30
3.5	Amplitude of 1000 pulses recorded from alpha pixel <i>dtp08</i> with a minimum threshold of 80 mV. The cluster of pulses at the left corresponds to signal noise.	32
3.6	Pulse width vs. amplitude distribution for various triggering thresholds. Each shows an apparent tail of wider pulses at lower amplitudes.	33
3.7	Standard deviation of 1000 pulses recorded from alpha pixel <i>dtp08</i> vs. threshold amplitude. Above approximately 315 mV the pulse timing becomes much more self-consistent.	35
3.8	Cumulative distribution for amplitude of pulses from detector <i>dtp08</i> . A threshold of 315 mV rejects 20 % to 25 % of detected events.	35
3.9	Streaking artifacts attributed to motion and undersampling [20].	36
3.10	Distortions introduced in image reconstruction from array positioning errors. Ring artifacts are a function of the comparatively small number of projections.	38
3.11	Reconstructions with various lateral array offsets.	40

3.12	Earth mover's distance in arbitrary units for each pair of images from Figure 3.11. The figure is symmetric: both the bottom left and top right square show the distance between reconstructions with 1 cm and 15 cm. The distance increases monotonically with the magnitude of the offset.	41
4.1	Linear attenuation coefficients for the eight simulated materials for 14.1 MeV neutrons. Four materials (PVT, Aluminum, TNT, and RDX) are within 0.035 cm^{-1} of each other.	43
4.2	Example of model geometry. The nominal position of the array is shown in light blue. The array (dark blue) shown here has been translated in the $+X$ (rightward), $+Y$ (upward), and $+\theta$ (counter-clockwise) directions. The magnitude of the array offsets has been exaggerated for illustration.	46
4.3	Number of counts recorded as a function of horizontal angle θ associated with each α pixel. Measured counts (blue stars) are counts in each neutron pixel that have been correlated to a detection in the relevant alpha pixel. Orange lines are curves fit to the measured counts.	50
4.4	Number of counts recorded as a function of vertical angle ϕ associated with each α pixel. The similarity of the curves indicate minimal differences between the alpha pixels with respect to vertical angle. The asymmetry between positive and negative angles was attributed to neutron scatter off of the floor of the room.	51
4.5	Example simulation output, bottom row in log scale. The simulated annulus was 17.22 cm in diameter with an inner core of TNT and a 2.32 cm thick outer ring of depleted uranium. As increasing numbers of neutrons are simulated additional features become apparent.	52
4.6	Conceptual workflow for simulation.	54
5.1	Consequences of poor learning rate choices. Here, J represents the network cost as a function of the trainable parameters θ . The network updates for a small learning rate will converge slowly (left) while high learning rates may oscillate about a minimum error. Image reproduced from [8].	58

5.2	Comparing the use of pre-training on a general image classification data set.	61
6.1	Distribution of errors in the final network. While the distribution in Figure 6.1a appears multi-modal, this is a consequence of histogram binning.	63
6.2	Dependence of error on radius of object imaged.	64
6.3	Dependence of error on material of object imaged.	67
6.4	Count rate as a function of source-detector angle for a simulated void measurement (top) and with a 82 kg cylinder of depleted uranium present. For this simulation the array was shifted 12 cm in the $-X$ direction and 13 cm in the $+Y$ direction.	70
6.5	Sinogram for the object described in Figure 6.4.	71
6.6	Flow chart to reconstruct image from simulated measurement.	72
6.7	Distribution of cross sectional error for the idealized reconstruction as well as the corrected and uncorrected MCNP simulations.	74
6.8	Reconstruction based on ray trace (left) and MCNP simulation (right). The two cross sections (center) show that the MCNP reconstruction has significant oscillations not present in the ray trace-based reconstruction. The overall radius is a close match in both reconstructions.	74
6.9	Cross sectional error in reconstructed images vs. radius of imaged object. . .	75
6.10	Support area for an example measurement of a steel cylinder surrounded by a jacket of TNT. The object extends beyond the support area (shown as a red dashed line), resulting in insufficient information to perform tomographic reconstruction.	76

Chapter 1

Introduction

The ability to look inside of an object without disassembling it is of great value to a variety of fields, including the detection of smuggled nuclear material. Because they interact via mechanisms different from X-rays, neutron imaging provides different information about the material being interrogated. Because of the nature of nuclear material, it would likely be shielded behind dense, high-Z structures that are effective at blocking X-ray and gamma radiation. Neutrons, however, are not as attenuated by shields of this nature. Further, neutrons interact with only the nucleons (protons and neutrons) of a material, while X-rays interact with the electron field around atoms, ignoring the uncharged particles in the nucleus. This means that neutron imaging has the ability to resolve the isotopic composition of an interrogated sample [1]. These features make neutron imaging an effective tool in the identification of smuggled nuclear material. Smuggled nuclear material may be illicitly transported via a number of undeclared means and locations, increasing the cost of deploying permanent neutron radiography devices in every potential smuggling site. A portable neutron radiography device would be an alternative to permanent installations but would require the removal of much of the structural weight associated with current versions of these systems.

The goal of neutron transmission imaging is to measure the probability that a neutron emitted at a certain angle Ω reaches the imaging detector pixels without interacting with the objects in the system¹. From this probability it is possible to infer the likelihood of

¹From this point forward, the region being examined by the imaging system will be referred to as the “**imaging stage**” or just “**stage**”

interaction along the path from the neutron source to the detector pixel at angle Ω and, using equation 1.1, draw conclusions about the material properties along the path $\vec{s}$. The intensity I passing through the object is given by

$$I = I_0 \exp\left(-\int_{\vec{s}} \mu(\vec{x}, E) d\vec{s}'\right), \quad (1.1)$$

where $1/\mu$ is the mean free path of the radiation in the material—the distance in that material at which point I_0/e particles are expected to remain un-interacted. μ is a function of the type of radiation, the energy of that radiation, and the material through which the radiation travels.

Unfortunately, this is complicated by several factors. Each detector pixel (corresponding to the angle) has its own intrinsic material and geometric efficiency which is inherently less than one; it is likely that many of the neutrons that do not interact in the imaging stage will not be registered by the detector pixels. Further, a neutron that interacts in the imaging stage does not necessarily leave the system immediately. It may continue to interact, with each interaction altering the path along which it travels. The possible outcomes are shown in Figure 1.1. The neutrons that undergo these additional interactions now carry information about path $\vec{s}'$ instead of $\vec{s}$; if they subsequently interact in the imaging pixels they are no longer part of the transmission image signal.

Neutrons may scatter in the imaging stage in two ways: elastically and inelastically. In an elastic scatter the energy of the original neutron is distributed between the resultant neutron and the target of the interaction (some nucleus in the system). In an inelastic scatter a photon is created which carries some of the energy from the reaction. The photon may escape, interact in the imaging stage, or interact in an imaging pixel, again registering an event that is not part of the transmission image signal. The updated process is shown in Figure 1.2. The addition of these photons detection events to the total system counts further reduces the signal to noise:

$$\frac{P_{transmission} \times P_{detection}}{\sum_n (P_{detection|scatter_n} + P_{\gamma|scatter_n} \times P_{detection|\gamma})}, \quad (1.2)$$

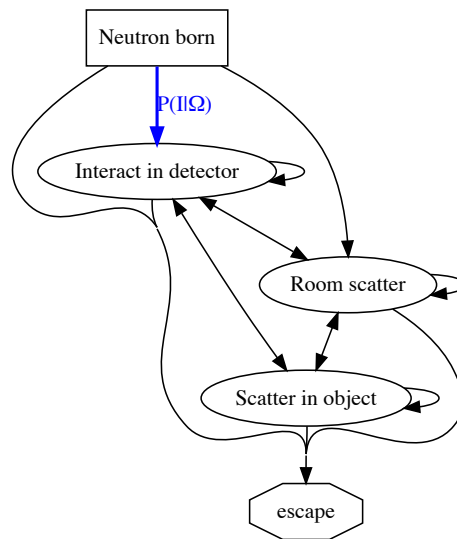


Figure 1.1: Possible outcomes for a neutron generated in the associated particle imaging system. Additional neutron sources (e.g. environmental background or fission) are not represented.

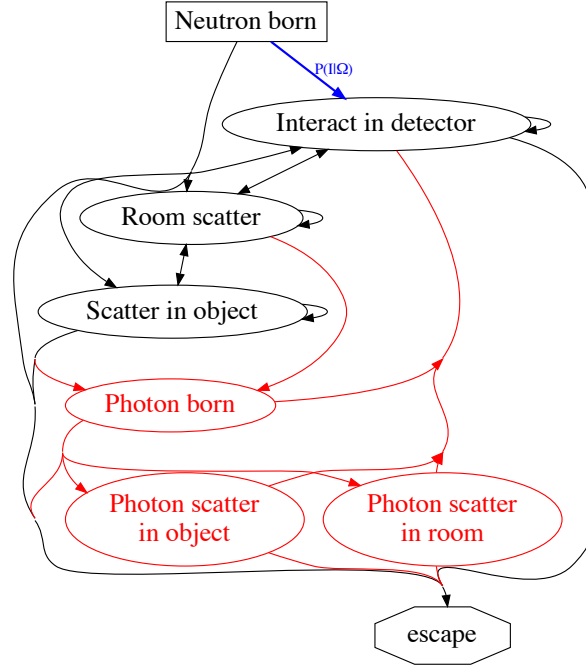


Figure 1.2: Possible imaging system interactions for neutrons and photons. The relationship that carries the transmission image information is emphasized in blue; other processes may represent noise that must be accounted for. As the NMIS system examined in this work produces few detectable photons in the absence of a scattering option, these are not included in this figure. Other API systems produce photons in their generator component and have more complicated interactions.

where $P_{transmission}$ is the probability of a neutron reaching a detector without interacting in the imaging stage, $P_{detection}$ is the probability of a particle registering a pulse in a detector, $P_{detection|scatter_n}$ is the probability of a neutron registering a pulse in a detector after n scatters, $P_{\gamma|scatter_n}$ is the probability that scatter n produces a photon, and $P_{detection|\gamma}$ is the probability of a detector registering a pulse given that a photon has been produced.

Associated particle imaging (API) is one method used to correct for the noise in this signal. In general, API uses particle sources that produce both a particle emitted towards the source and one or more secondary particles whose properties are related to the imaging particle. Oak Ridge National Laboratory's Nuclear Materials Identification System [2] and its variants employ a deuterium-tritium (DT) generator. A beam of deuterium ions is accelerated through a voltage field towards a target impregnated with tritium. Most of these nuclei fuse, producing a neutron and an alpha particle:



The energy of the resultant particles can be calculated from the masses of the reactants and products:

$$Q = \left(m_{{}^1_0\text{n}} + m_{{}^4_2\alpha}\right) - \left(m_{{}^3_1\text{H}} + m_{{}^2_1\text{H}}\right) c^2 \approx 17.59 \text{ MeV}. \quad (1.4)$$

In the rest frame of the reaction, the net momentum of the products is zero. This means that the energy is distributed according to the masses of the products:

$$T_n = \frac{1}{2}m_n v_n^2 = \frac{Q}{1 + m_n/m_\alpha} \approx 14.05 \text{ MeV}, \quad (1.5)$$

and

$$T_\alpha = \frac{Q}{1 + m_\alpha/m_n} \approx 3.54 \text{ MeV}. \quad (1.6)$$

From this energy it is possible to compute a velocity for the neutron and alpha particle using the equation

$$T_n = \left(\frac{1}{1 - v^2/c^2} - 1\right) m_n c^2. \quad (1.7)$$

This gives a speed of 5.127 cm ns^{-1} for the neutrons and 1.306 cm ns^{-1} for the alpha particles.

If the initial energy of the deuteron in the laboratory frame is neglected, the alpha particle and the neutron travel in precisely opposite directions. Oak Ridge-produced API systems to this point use alpha detectors to record the time and direction of an alpha particle from the source and establish a time and direction window corresponding to a neutron travelling in the opposite direction. Detection events outside of this window are considered to be noise; events inside this window are used to construct the transmission signal. An example measurement for a single neutron pixel is shown in Figure 1.3.

Current versions of the NMIS system employ a DT Generator with an integrated pixelated alpha detector. The imaging pixels are placed at a fixed distance from the generator in an arc with a radius corresponding to that distance, ensuring that each pixel corresponds to the same unit of angle at the same distance as all of the others.

1.1 Other nuclear imaging modes

Neutrons and photons each interact stochastically in matter and may both be treated in accordance with equations 1.1, but their differences mean that they can be employed in a complementary manner for imaging. Unlike neutrons, X-rays travel at the same speed no matter what their energy. They also tend to interact primarily with the electron cloud surrounding the nucleus rather than the nucleus itself (though high-energy photons— $>1.7\text{ MeV}$ —may directly interact with the nucleus). As a result, high- Z materials (which have more electrons) are more effective at shielding X-rays than they are at shielding neutrons. The electromagnetic nature of photons also means that they do not differentiate between different isotopes of the same atom, which limits their applicability to some questions of interest in the field of nuclear safeguards and non-proliferation. By contrast, neutrons are effectively shielded by low- Z materials and largely unaffected by high- Z materials. This makes them ideal for imaging objects which are contained within dense metal shielding and for high-contrast imaging of low- Z (e.g. hydrogenous) materials. The regions of applicability are illustrated in Figure 1.4.

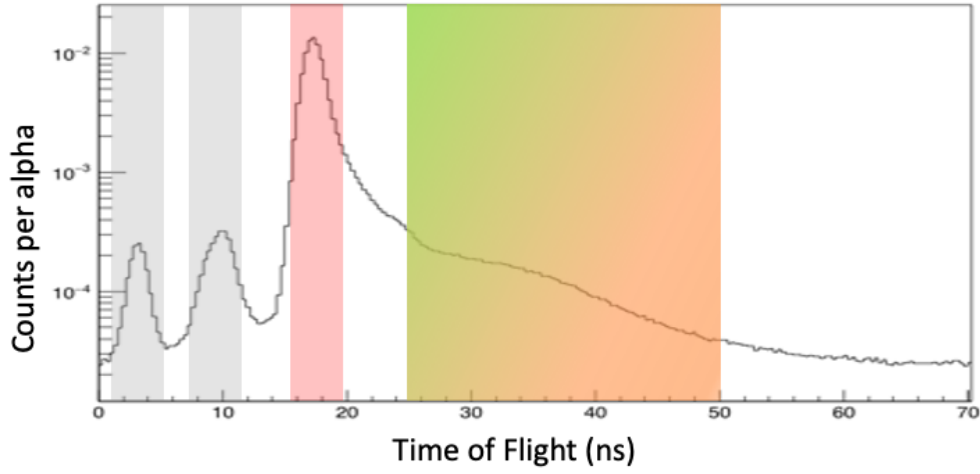


Figure 1.3: Distribution of neutron detector counts correlated with alpha detection events in APNIS, a related Oak Ridge API system. The gray bands represent photons produced in the generator or imaged object; the red band identifies neutrons that pass through the object without interacting (transmitted neutrons), and the green/orange band identifies neutrons produced by scattering or fission in the object [3].

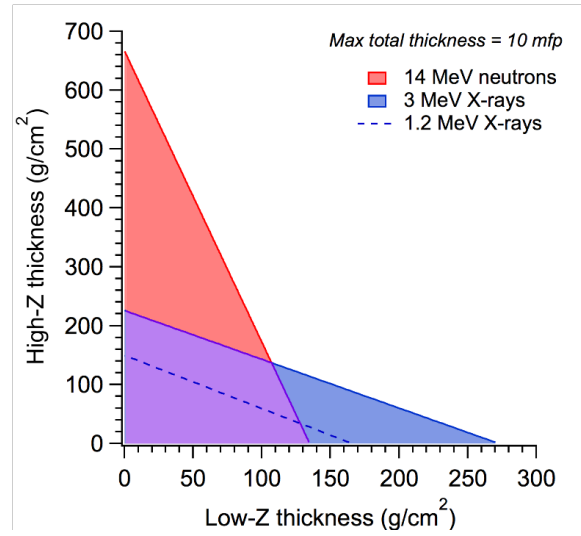


Figure 1.4: Regions of applicability for neutron and X-ray imaging. Each can be usefully employed to image up to ten mean free paths of material, but what defines that thickness is different for each particle [3].

1.2 Angular dependence

Determination of the position of the array where nothing is between the source and the neutron detector array (assuming that the imaging and alpha pixels have been aligned to the same time coordinates and neglecting the timing performance of the detectors) is relatively simple. The fastest a neutron can arrive in the detector is along a direct path from the generator to the pixel without undergoing any scattering interactions. With no scattering material in the imaging stage there are few secondary gammas to detect, so the distance to any pixel can be calculated by taking the time corresponding to maximum coincidence between that pixel and an alpha pixel using

$$d_{img_i} = \left(t_{corr} + \frac{d_\alpha}{v_\alpha} \right) v_n, \quad (1.8)$$

where d_{img_i} is the distance from the source to neutron detector i , t_{corr} is the time bin with the highest number of correlated counts between an alpha and neutron detector, d_α is the distance from the source to the alpha detector, and v_α and v_n are the speed of the alpha particle and neutron respectively.

In the presence of scattering material this problem becomes more complicated. There are two types of scattering reactions: elastic and inelastic. Elastic scattering describes a reaction where a neutron encounters a nucleus and transfers some of its energy. The speed and direction of travel of that neutron are then altered. The amount of energy lost by the neutron can be any value between zero and

$$T' = \left(1 - \left[\frac{A-1}{A+1} \right]^2 \right) T_0,$$

where A is the mass number of the target nucleus [4]. The angle of travel of the neutron after the collision (relative to its initial direction) is described by

$$\theta' = \text{acot} \left(\frac{1 - A \cos(2\theta'_A)}{A \sin(2\theta'_A)} \right),$$

where θ'_A describes the direction of travel of the target nucleus.

Figure 1.5 shows the probability distribution of scatter direction (relative to its initial direction of flight) for a 14 MeV neutron from various materials. A consequence of equation 1.8 is that high-Z materials (which mostly produce small-angle scatters) are more likely to change the velocity of a scattered neutron a small amount. In order to accurately measure the interaction probability in these materials, therefore, it is necessary to have very precise timing and angular resolution—i.e. the ability to differentiate between two events that occur very close to each other in either time or direction.

Inelastic scattering is similar, but the energy is no longer distributed between the neutron and the scattering nucleus. Instead, some of the energy places the nucleus in an excited state. The nucleus then decays to its low-energy ground state emitting a photon in the process. The probability of inelastic scatter is typically lower than that of elastic scatter, but these characteristic photons are visible in neutron radiography measurements.

Each of these scattering effects can produce particles which interact in the neutron detector pixels, contributing to noise in the measurement and making the path from the source to the detector pixel appear less attenuating than it is.

1.3 Image reconstruction

A single projection through an object of interest is of value, but significant additional information can be gained by taking multiple radiographs from different angles, each of which is referred to as a projection. These measurements combine to form a sinogram—a map of the attenuation as a function of detector angle on one axis and projection angle on the other. The method in computed tomography that NMIS employs is “fan beam” tomography, which uses a theoretical point source (approximated in NMIS by a 3.5 mm diameter disc) projecting rays through an object to a pixelated detector system. The source and detector are then rotated about the center of the imaged object a number of times and projections are taken at each stage. These measurements give an attenuation probability as a function of projection and detector angle. These can be converted to a cartesian coordinate frame via inverse transform to produce an image whose intensity is the inverse of the mean free path of the particle (neutrons in the case of NMIS) in that location. A computer algorithmically

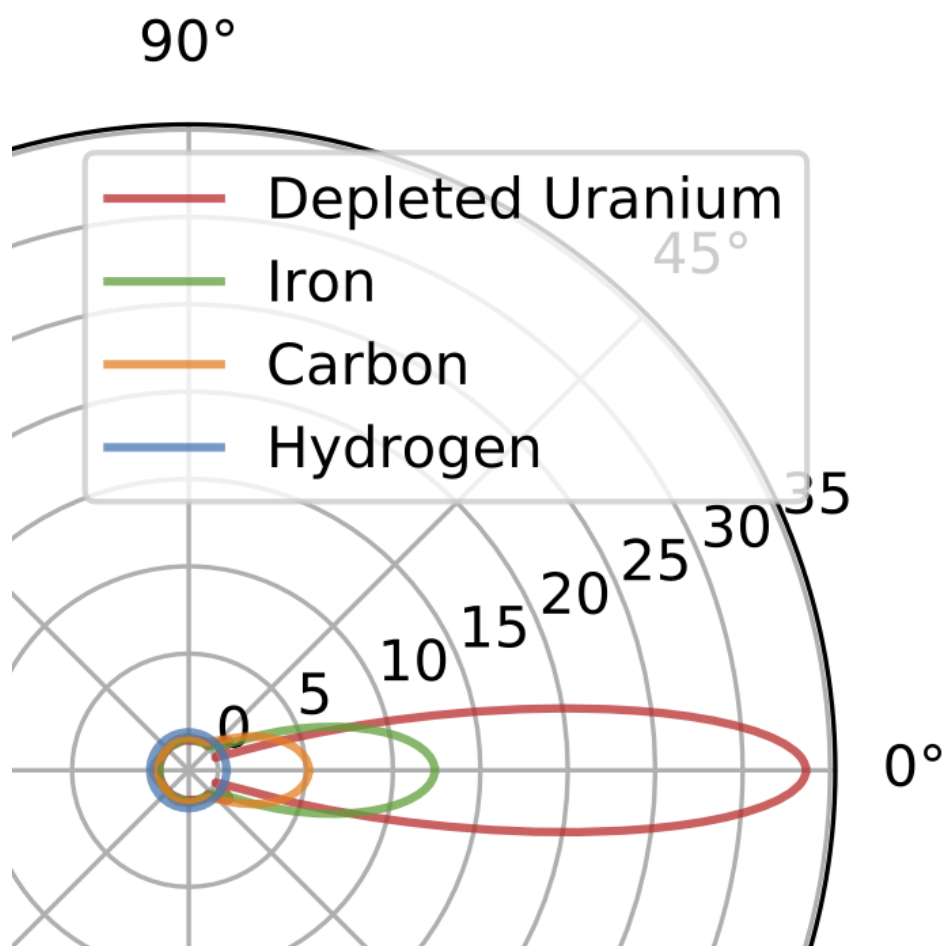


Figure 1.5: Distribution of neutron scatter angles for elastic scatter off of depleted uranium, iron, carbon, and hydrogen nuclei. Data from [5].

attempts to find the attenuation as a function of position that simultaneously solves Equation 1.1 for each detector pixel angle and projection angle.

One such algorithm is the simultaneous algebraic reconstruction technique developed by Kak and Slaney [6]. This is an iterative algorithm that creates and solves a system of linear equations. First, rays are projected from the source to each detector pixel through a discretized grid of unknown attenuation values. Each ray is converted to an equation based on the length of its intersection with each of the grid squares through which it passes. This is repeated for each projection angle to generate $n_{projections} \times n_{pixels}$ linear equations, where $n_{projections}$ is the number of projections taken and n_{pixels} is the number of detector pixels. Once this system of equations is solved, the process can be repeated if necessary to converge to a more accurate solution.

In the case of real world measurements, this problem is typically ill-posed. When an interrogating particle (neutrons in this case; x-rays in most commercial systems) is scattered it remains in the imaging system and may be detected at some later time. If it does register an event in a detector pixel it causes the path between the source and detector to appear less attenuating than it is. Additional sources of reconstruction error include background radiation fields, Poisson error from the detectors, ring effects from imperfect angular resolution of the detector system, and streaking effects from the energy dependence of the attenuation probability [6].

1.4 Neural Networks

Neural networks describe a general class of computer algorithms, roughly inspired by biological neurons. Each neuron has a number of inputs and a single output:

$$N(\vec{x}) = f \left(b + \sum_i w_i x_i \right), \quad (1.9)$$

where each weight w_i and bias b represents a variable parameter. These parameters are “learned” over the course of training a neural network. The symbol f represents an “activation function,” which transforms the weighted sum of the inputs in some (typically

nonlinear) way. A neural network consists of many of these functions whose outputs provide the inputs to the next neuron. In the case of supervised learning, the training algorithm adjusts the weights and biases to minimize some error function.

The algorithms most commonly used to train a neural network are variations on gradient descent. The concept of gradient descent is simple: for each trainable parameter, the partial derivative of the error function with respect to that parameter (i.e. the direction in which changing the parameter would increase the error and the magnitude of that increase) is found. The parameter is then adjusted in the opposite direction in proportion to the magnitude of the gradient by

$$w'_i = w_i - \eta \frac{\partial E}{\partial w_i}, \quad (1.10)$$

where w_i is the parameter to be adjusted, $\frac{\partial E}{\partial w_i}$ is the partial derivative of a cost function with respect to that parameter, and η is a term that sets the scale of the adjustment of w_i during each iteration. The combination of many of these neurons allows the approximation of highly nonlinear functions. Several different types of networks (“architectures”) have been developed that optimize well for specific tasks. One such architecture, convolutional neural networks, was originally developed for processing images, speech, and time-series data [7]. It relies upon the intuition that the information contained in a data presentation may be related to the adjacent data bins. It further supposes that the nature of these relationships is consistent across the entire data vector. This allows the network to learn meaningful convolution “kernels”, effectively looking for patterns across an image. Convolution in the discrete case is described by

$$N(i, j) = (K * I)(i, j) = \sum_m \sum_n I(m, n) K(i - m, j - n), \quad (1.11)$$

where i and j are the indices describing the location of each pixel, K is the convolution kernel, I is the image being convolved, and $*$ is the convolution operator. In practice, networks use multiple convolution layers. This allows small, primitive features (edges, peaks, solid regions) to be combined spatially into more complex features as the input propagates through the network. The combination of multiple features is illustrated in Figure 1.6. This figure shows the regions of an input image that respond the most (neurons output high values) in

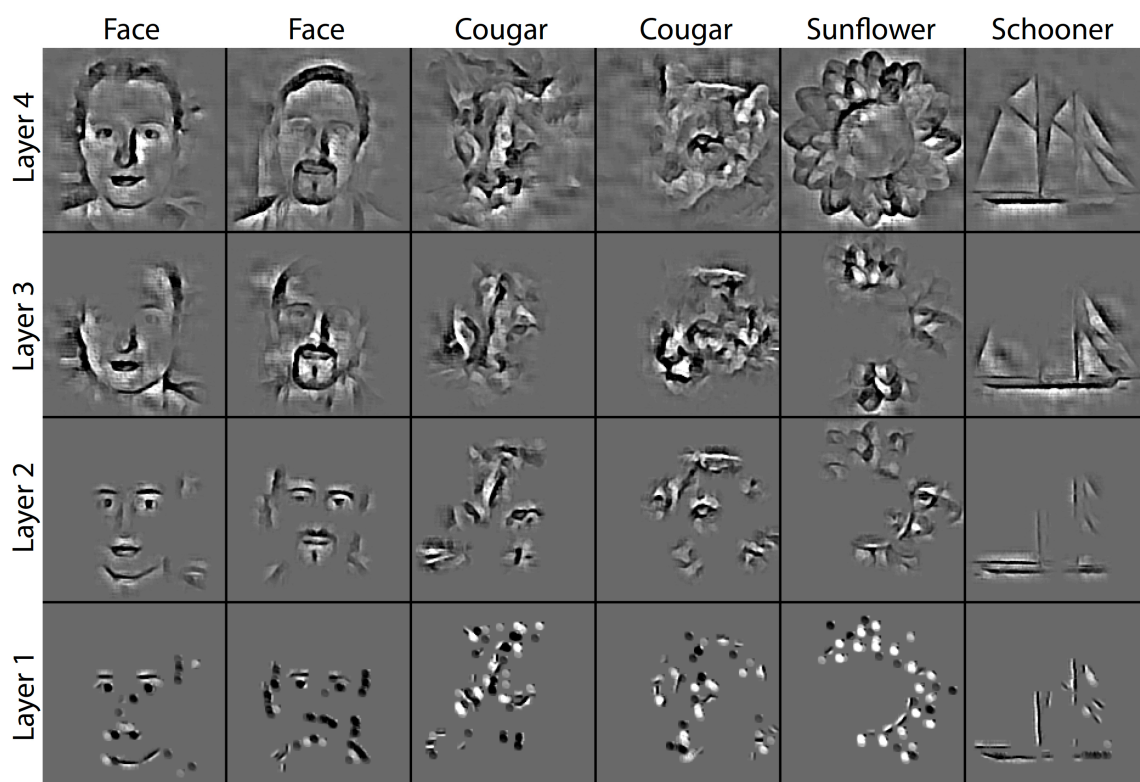


Figure 1.6: Each layer of the convolutional neural network combines features from the previous layer to compose more complex features [8].

subsequent layers for specific inputs. In early layers (near the bottom of the picture) the patches correspond to regions of color and edges. As the inputs propagate through the layers they are combined into more complex shapes, and by the end are recognizable as images from their respective categories.

While neural networks provide universal function approximation, they require increasing depth (number of layers) in order to model this behavior. Training these networks based on gradient descent algorithms (or variants thereof) has the potential to be delayed by flat regions on the error surface arising from repeated multiplication by a Hessian matrix (the matrix of partial derivatives of error with respect to the parameter being adjusted) with eigenvalues much larger or much smaller than 1 [9]. This means that for many problems, layers further from the output of the network tend to exhibit small update values. As layers are added to these networks the value of the gradients by which they are trained can approach zero or “vanish”. If the partial derivative of the error is zero, the parameters in these layers will not be adjusted during training and the network will not converge. He et al have introduced an architecture that attempts to address this [10], commonly referred to as “ResNet”. They observed that after a certain point, adding additional layers degraded the final performance of the overall network as a result of limitation in current solvers/optimizers. In addition to alternating layers of convolution and pooling, ResNet provides connections between subsequent blocks of layers, as illustrated in Figure 1.7. It has been shown that this allows ResNet to act as a universal function approximator as a function of network depth [11]. This contrasts with the proof of traditional feed forward neural networks which achieve universal function approximation using width.

If feed forward architectures can be viewed as representing a function by approximating the small region to which each neuron is receptive, ResNet can be seen as an iterative reconstruction of the input/output mapping using a weighted sum of simpler functions. A conventional multilayer perceptron attempts to find $f_0 \circ f_1 \circ \dots \circ f_{n-1} \circ f_n(x)$ where each layer i attempts to transform the output of the previous layer to minimize the error of the entire network by approximating a function f_i . Each layer is an independent function that maps the output of the previous layer onto a space more predictive of the output. The ResNet architecture instead attempts to find $f_n(f_{n-1}(x) + x)$ for $n \in N$ for a network

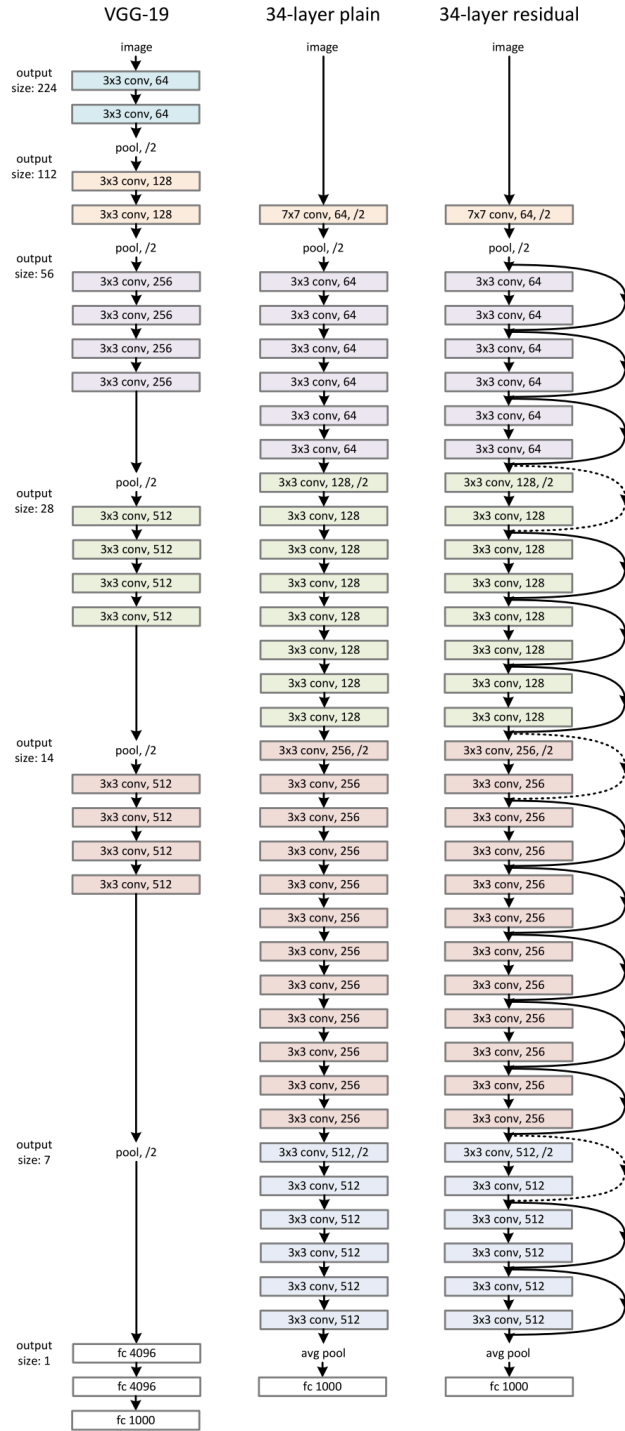


Figure 1.7: ResNet architecture (right) compared with previous architectures. The skip connections are represented as curved arrows and allow each set of layers to operate on the residuals of the previous block [10].

with N blocks. Because of these additional connections, each layer instead attempts to recover some information not captured by the earlier layers. Because of their flexibility, ResNet architectures represent a useful starting point for data that can be presented in an imagelike² form. Further, previously published weight values derived from training on large sets of images can accelerate training as the early layers of networks with these weights already respond to primitive features (e.g. edges and gradients).

1.5 Motivation

The goal of this work is to determine the position of the neutron detector array relative to the D-T generator. This serves two purposes: first, the position of the array determines the time-of-flight window to reject scattered neutrons, and, second, the array position is required to convert the angular data acquired from multiple projections from a sinogram into an image.

²i.e. a 2-d array of values on a common scale bound within a finite range

Chapter 2

Review of Prior Work and Statement of Original Contributions

2.1 Nuclear Materials Identification System (NMIS)

Variants of the NMIS system have been built or proposed since 1994 [2]; even these reference prior work in the field. Early versions employed ^{252}Cf and detected the time of emission using a cloud chamber. The first version of the modern NMIS system was described by Mihalcz et al. [12]. The authors describe using the time correlation between multiple channels in the same detector system to classify radiation detection events. This method was extended by Mullens [13], introducing the potential for tomographic reconstruction from NMIS measurements by using a larger number of smaller detectors to provide additional angular resolution. Their tomographic reconstructions were performed using backprojection, but they recommended the use of the algebraic reconstruction technique (ART, described in [6]) as it was more suitable to the small number of projections expected to be available to an NMIS measurement. They also proposed several different modes for the NMIS system employing both untagged (no API) and tagged (API capable) neutron sources, as well as a variety of neutron detector elements which might provide the ability to differentiate between neutron and photon events (a.k.a. pulse shape discrimination). One of these sources was developed and described by Chichester et al. [14]. The new source, a deuterium-tritium generator, allowed for a neutron source that could be turned off, could be directionally

tagged, and could be tagged in time using an integrated alpha particle detector. This source was integrated into NMIS by Grogan et al. [15]. Previous versions of NMIS distinguished between transmitted and scattered neutrons only by time of flight. Neutrons that scatter off of high-Z nuclei, however, tend to retain most of their energy and therefore may arrive at essentially the same time as unscattered neutrons. At the time that this functionality was added, the timing resolution of NMIS was 1 ns, so it was possible for a neutron to scatter in the imaging stage then interact with a neutron detector not on its original path. This would cause an underestimation of the attenuation between the source and the pixel that detected the scattered neutron. With the addition of a pixelated alpha detector to the neutron source, the NMIS software could reject these scattered neutrons, improving the accuracy of the attenuation measurement and the resultant reconstructed image.

In 2008, Grogan et al. [16] used MCNP-PoliMi (a variation of the Los Alamos National Labs Monte Carlo program MCNP) to demonstrate a possible measurement using NMIS. They simulated a stainless steel trough filled with UO_3 powder and were able to consistently predict the mass of the powder in the trough with a root mean squared error of 0.66%. They used multiple MCNP simulations to approximate the coincidence between alpha and neutron detectors. Their results were consistent even when short measurement times were simulated due to the low aggregate uncertainty when all simulated detector positions were taken into account.

In 2012, Swift [17] successfully combined tomographic and maximum-likelihood reconstruction methods using both gamma- and neutron-based simulated measurements to show how a high-fidelity reconstruction of a cylindrical object could be generated from a version of the NMIS system known as FNMIS (which incorporated a stage to rotate the imaged object for tomographic reconstruction). They demonstrated that in this well-characterized detector it was possible to reconstruct the radii of the cylinders that made up the test object with an absolute error of 1.58% to 4.25%. This established a floor for minimum error in a measurement under near ideal measurement scenario, as well as extending the simulation work started by Grogan et al. [16].

Pena [18] applied the Kolmogorov–Smirnov test to measurements taken with this version of the NMIS system as a metric matching the statistical distance between two distributions

to determine whether or not two measurements from the NMIS system were of the same object. As a part of this research she sought to determine how invariant her matching distance was to rotations of the examined objects. She analyzed both rotationally symmetric and asymmetric objects, with testing showing that this method was able to match some template measurements to the object in question in spite of rotations up to 48° while others only successfully matched at rotations up to 24° . These measurements were taken from between thirty and sixty projection angles, which would likely be sufficient for reasonable two dimensional CT reconstruction of the object. This demonstrated statistical inference methods were consistent across multiple measurements of the same object.

Troxell [19] was the first to examine the effects of detector misalignment in the NMIS system. He developed a neutron detector array consisting of seven blocks of four detectors each, mounted in plastic enclosures. These blocks were mounted on a metal arm in a pseudo-curved array, with block angled to be tangent to a circle with a radius of 115.1 cm from the neutron generator. He conducted a series of measurements with this detector array aligned and with axial, lateral, and rotational offsets present, both with and without the presence of annuli composed of aluminum, PVT plastic, and depleted uranium. Offsets examined were in the range -15 cm to 15 cm at 5 cm intervals while rotational offsets ranged from -6.789° to 6.789° in steps of approximately 2.25° . He determined that offsets in this range still produced sinograms with recognizable features and demonstrated a qualitative relationship between those offsets and differences in measurements.

2.2 Computed tomography

Analysis of errors in tomographic imaging can be divided into two categories: measurement errors (errors in the direct radiographic measurements) and reconstruction errors (artifacts). Nearly all discussion of both of these focuses on x-ray methods.

Tomographic reconstruction of radiographic images suffer from reconstruction errors (artifacts) that are characteristic of specific real-world imperfections in detector systems. Boas and Fleischmann provide a (relatively) recent survey of the types of reconstruction errors, their causes, and some methods of reducing their impact in the context of medical

imaging [20]. One class of artifacts, beam hardening, is caused by the fact that typical x-ray sources are not monoenergetic. The high energy portion of the beam is attenuated less than the low energy portion of the beam. This results in a flux which is greater than $I = I_0 \exp(-\mu x)$, indicating that there is less scattering mass present along that path. This appears as darker streaks in the final image on the detector side of the scattering object. NMIS addresses this in transmission imaging by attempting to measure only unscattered neutrons using a monoenergetic source. Jelinek and Overton [21] attempt to quantify a number of these effects in fan beam radiography. Of specific note is their discussion of reconstruction errors introduced by angular offsets during measurement. They note that errors in detector angular offset of as little as $\frac{1}{3}^\circ$ can have a significant impact on the reconstructed image, though this was in reference to multiple-rotation fan-beam tomography. Their theoretical work was extended by Wenig and Kasperl [22]. They simulated cone-beam tomographic measurements of a phantom with various departures from ideal geometry, then quantified the deviations in seven measurements of parts of the object. They identified the two largest alignment factors that had the largest impact on the final reconstruction as “tilt about the [source-detector] axis and [shift of object] rotation about the [lateral] axis”. They identify a threshold of one-half pixel width as sufficiently accurate to produce useful measurements, though they arrived at this conclusion not from translations of the detector plane (examined in this work) but translations of the rotation axis¹.

Techniques to quantify and account for geometric misalignment in measurement error are well summarized by Ferrucci et al. [23]. This work describes alignment primarily for cone-beam x-ray tomographic systems. They classify attempts to measure these alignments into two categories: methods based on reference objects (calibrated imaging phantoms of various types) and methods based on reference instruments (e.g. laser or interferometer distance measurements). The work identifies four conditions that constitute an aligned imaging system:

¹For an investigation of the impact of rotation axis shift on NMIS measurements in the context of template matching please see the work of Pena [18]

1. The intersection of the magnification axis with the detector (also known as the principal point—a term used in geometrical modeling of cameras) is coincident with the centre of the detector,
2. the magnification axis² is normal to the detector,
3. the magnification axis intersects the axis of rotation at a 90° angle, and
4. the projection of the axis of rotation is parallel to the detector’s columns.

With a phantom present for each geometric projection, the procession of the phantom as the detector system performs multiple projections can be used to predict the geometric alignment of the source, detector array, and rotation axis of the imaging stage using optimization methods to minimize error [24]. Other proposed methods use more complex phantoms to predict alignment from only a single projection that also guard against the potential error introduced by local minima in the optimization process, therefore converging on a global solution [25]. A single-projection method also allows for reconstruction in the case where the array may move between projections. These alignment values can then be used to improve the reconstruction of the final image using algorithms designed to account for the array position [26].

Some methods have been proposed to perform geometric calibration without a phantom present. Kyriakou et al. [27] examined such a technique for a near-circular scan path by minimizing the entropy in the paths between eight points selected from an imaged object. This method had the additional advantage of not requiring an application-specific phantom, meaning it could be applied to ad-hoc scanning paths (the combination of geometries between projections). Repeated projections and reconstructions were performed on each measurement while iteratively adjusting the geometry until the entropy in the reconstructed images was minimized. Medical images reconstructed via this method were closer to the ideal calibrated reconstructions, though this method assumes that erroneous reconstructions are lower-entropy than the correct image. Zheng et al. [28] used a related assumption, but instead of examining the entropy of the reconstructed image they sought to maximize the consistency between projections. This method was able to predict η (rotation of the detector

²The magnification axis corresponds to the x-axis in this work.

plane about the source-detector axis) and μ_c (the detector offset to the left or right of the source-detector axis) to high accuracy and provide estimates of ν_c (vertical detector offset) and the source-detector distance. The accuracy of these parameters was dependent on the selection of the cost function which measured the consistency between projections, so a different selection of cost function would be expected to provide different trade-offs.

2.3 Original contributions

Previous work on neutron imaging has assumed that the source and neutron detector array are well aligned, or at least that the alignment is constant for measurement with an inspected object absent (void measurement) and with the object present. Previous work on geometric alignment of tomographic systems has focused on the use of calibration phantoms viewed from multiple projection angles. This work uses features of the NMIS system to determine geometric alignment from a single projection with no reference object present. It also shows that neural networks are a viable tool for analysis of NMIS data. A new simulation framework has been developed allowing for simulation of large numbers of detector array alignments. This framework allows for simulation of different measurement times in order to test the sensitivity to Poisson error in the measurement. No previous work has attempted to address geometric misalignment in a neutron imaging system or proposed a method of recovering alignment information from a single projection in a system capable of tomographic imaging. This work takes advantage of the associated particle imaging technique to determine alignment even in the case of very short (on order of seconds) simulated measurements.

Chapter 3

Description of Associated Particle Imaging System

3.1 Detector system hardware components

The neutron generator in the NMIS system is a VNIIA ING-27 continuous neutron generator. It has a nominal neutron output of 2×10^8 neutrons per second over 4π steradians, though this varies significantly from generator to generator and as a function generator age and operating time. During the time of these experiments its output was consistently limited to 60% of maximum beam current. This value was selected as a compromise between measurement time and component wear on the tritiated target.

The generator housing also contains an integrated silicon semiconductor alpha detector. This alpha detector is positioned 10 cm from the center of the target spot. It consists of 15 square 6 mm by 6 mm pixels arranged in a line parallel to the ground spaced such that the center of each pixel was 7 mm from its nearest neighbor (a pixel pitch of 7 mm). The center pixel (designated dtp08) is offset 6 mm such that the line at 90° to the deuterium beam axis intersects pixel dtp07. This is illustrated in Figure [3.1](#).

Neutrons emitted by the generator are detected by thirty-two Scionix Holland V25.4B102/1-E1-PXNEG plastic scintillation detectors. Each has a block of Eljen EJ-200 fast scintillating plastic $2.54\text{ cm} \times 2.54\text{ cm} \times 10.2\text{ cm}$ in size coupled to a photomultiplier tube. The times of the pulses are assigned using six Ortec CF8000 Octal constant-fraction discriminators

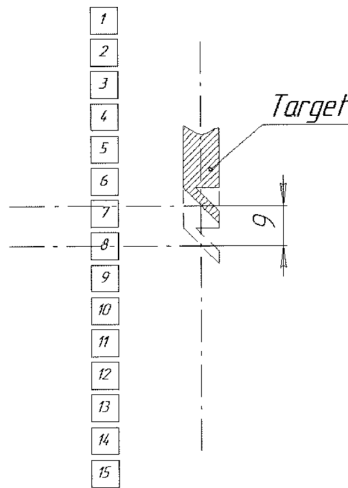


Figure 3.1: Position of alpha detectors relative to the tritiated target in the ING-27 generator.

(CFD), each of which has eight channels. A CFD assigns a time to a pulse by adding a delayed version of the original pulse to an inverted and scaled (by a “constant fraction”) copy of itself. The resultant sum has a steep slope which intersects the zero voltage axis at a steep angle and this intersection is assessed to be the time of the pulse. An illustration of this process is shown in Figure 3.2. Each CFD is capable of processing up to eight detector pixels. Up to thirty-two are used for neutron pixels, fifteen for alpha detector pixels (integrated with the ING-27 generator), and one is used for a Phillips Scientific Model 417 NIM pocket pulser which provides a reference during long periods with no recorded counts in order to maintain a constant time frame.

The outputs of the CFDs are sent to a Caen model V1190A 2ESST 128 channel multihit time to digital converter (TDC), which records the times from each channel. These are read by a laptop via a Caen model V1718 USB/VME bridge by custom software developed by Jim Mullens of Oak Ridge National Laboratory. This software both records the time/channel numbers to disk and tallies the correlation counts between each pair of detector pixels for time lags from 0 ns to 204.8 ns in increments of 0.1 ns (defined henceforth as 1 tick).

A laptop computer records the data from the USB/VME bridge while another runs the software to control the ING-27 generator. Generator output is controlled by adjusting the current of the deuterium beam. This current varies over time and may rise or fall over the course of several measurements. Data can be recorded in several formats; these are selected using a configuration text file stored on the laptop or as command-line flags when executing the data recording program. The first data file records the channel numbers from the TDC as well as the total counts recorded in each channel. The second data file records the raw “list mode” data where each record consists of a time code and a channel number. Earlier versions of this software used a 24 bit time stamp, a 7 bit channel ID number, and a single “continue” bit to indicate that 2^{24} tick had elapsed without a recorded event in any channel. Later versions of the software instead record this information in a 64 bit format, with 8 bits assigned to detector channel and 56 bits to record the time stamp. The earlier format recorded the amount of time elapsed between events while the later format records the time since the start of data collection. The third data format was the one employed in the analysis steps described later in this work. This format records the number

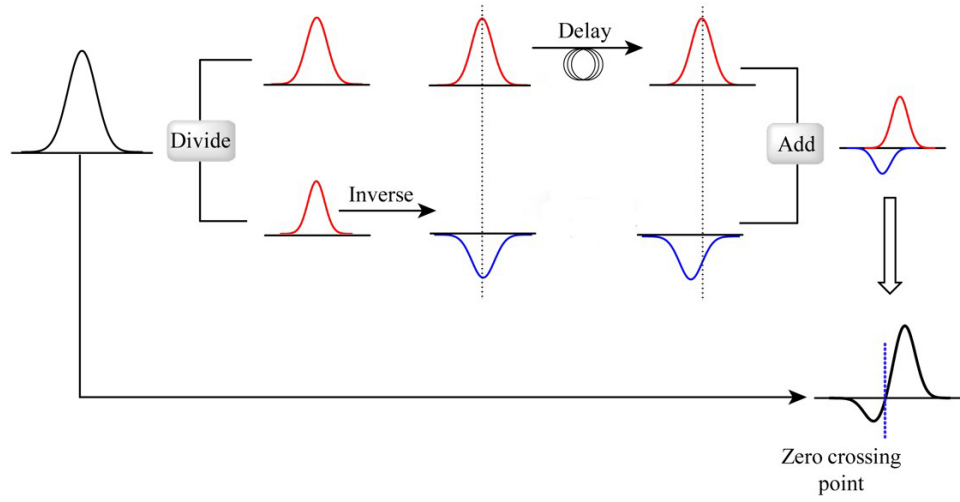


Figure 3.2: Illustration of the operation of a constant fraction discriminator (CFD). [29]

of coincident counts between each pair of detectors as a function of time lag. The time lag is configurable based on the TDC resolution setting (typically 100 ps) and command line flags which control the total number of bins to include (4097 or 2049 bins). By default the software only performs the coincidence calculation between alpha and neutron detectors, though it is capable of recording coincidence between neutron-neutron and alpha-alpha pairs as well. This coincidence calculation happens in real time—important, as the volume of data precludes the calculation of coincidence using traditional fourier methods. An example of this third data format is shown in [Figure 3.3](#).

3.2 Time-alignment of measured data

The correlation plots produced by the NMIS software are offset by some amount of time, distinct to each alpha pixel/neutron pixel pairing. This is primarily driven by the signal transport from the detectors themselves to the constant fraction discriminators as well as (to some extent) differences in the discriminators themselves. Before the correlation plots can be used to determine alignment information the channels must be aligned to be on the same time scales.

In order to perform this correction a void measurement is taken with all of the neutron detector pixels positioned at some known distance from the detector (in this case 115.1cm). While the detector array used for this calibration was not a perfect arc, this approximation is accurate to within 0.15 tick. For these measurements the generator was run at 60 % nominal power for 10 min.

The columns of the correlation measurement (each representing correlated counts between one alpha and one neutron pixel as a function of time lag) were filtered to only include pairs with at least 100 correlated counts in one time bin. This value was found to match pairs that included all of the neutron pixels and all but two of the alpha pixels (the emission cone for these alpha pixels did not correspond with any of the neutron pixels with the array at its nominal position). The time bin corresponding to the peak of each of these curves was recorded.

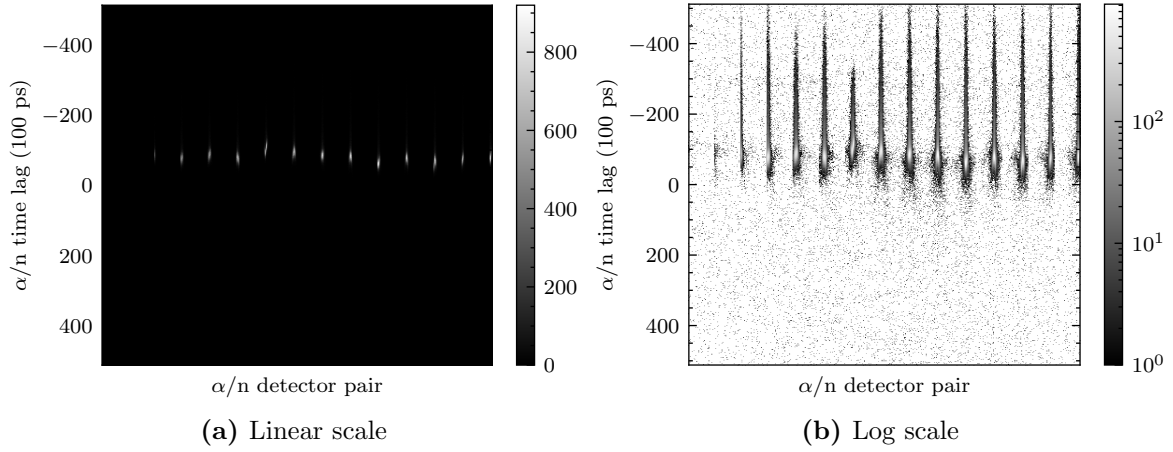


Figure 3.3: Coincident counts between each pairing of alpha and neutron detectors as a function of time lag. Time lag is on the vertical axis while the each column represents the pairing of one alpha and one neutron pixel. Each of the fourteen contiguous shapes represents several neutron pixels in coincidence with one alpha pixel.

It was assumed that each channel had a fixed offset that would be consistent across the pairings. Gradient descent was then used to minimize the quantity

$$\vec{E} = (\vec{t} - \mathbb{S}\vec{o} + \vec{p})^2, \quad (3.1)$$

where $\mathbb{S}$ is a filtering matrix to match offsets to combinations of specific pixels, $\vec{o}$ is the vector of offset parameters being adjusted, $\vec{p}$ is recorded peak locations from the correlation matrix, and $\vec{t}$ is the vector of target times, arrived at by calculating the nominal distance from the origin to the center of the front edge of each pixel divided by the nominal speed of the respective particle.

The vector of offsets was then adjusted through 500 000 iterations using equation 3.2 where $\frac{dE}{do_i}$ is the derivative of equation 3.1 with respect to offset o_i and η is a hyperparameter that defines the learning rate. A value of $\eta = 1 \times 10^{-3}$ was found to produce convergence in a few seconds of computer time.

$$o'_i = o_i - \eta \frac{d|\vec{E}|}{do_i} \quad (3.2)$$

In the case of alpha channels *ntp01* and *ntp02*, insufficient correlated counts were recorded to use this method. For these two channels an offset was assigned that was the average of the other six channels connected to the same constant fraction discriminator.

In order to validate this method, void measurements were taken on five days spread out over the course of four months. The calculation described above was performed on each void measurement and the resultant offsets examined for consistency. As shown in Figure 3.4, all channels converged to within an error of less than 1 ns, less than the standard deviation of the timing assignment from the constant fraction discriminators, which at the time of this measurement had a standard deviation on order of 20 ns—an increase in standard deviation of 0.06 %. This also suggests that a single void measurement taken for this purpose should be valid for at least several months so long as the same physical setup (e.g. specific cable lengths) is used.

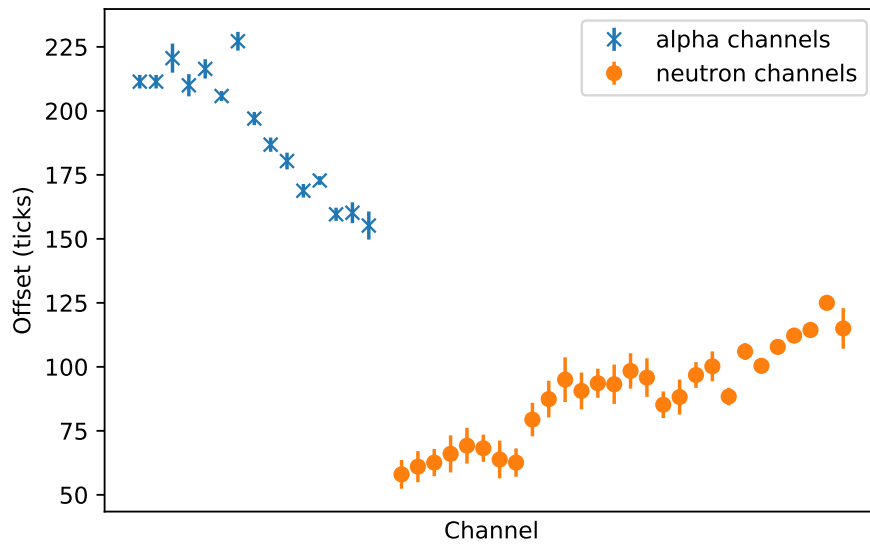


Figure 3.4: Calculated offsets for each detector with standard deviation from five void measurements.

These offsets were then used to align the detector timing for all of the measurements taken with this array. The python script used to calculate these alignments and save them to a json formatted text file is included as appendix [C](#).

3.3 Impact of timing accuracy in alpha and neutron detectors

The expected behavior of the system was a sharp rise in the correlation curve for each neutron/alpha detector pair at opposite or near-opposite angles corresponding to time t_{corr} described in equation [1.8](#). Since this represents the minimum transit time from the source to the neutron pixels (less the travel time from the source to the alpha pixels) there should have been no correlated counts before that. Experimental results showed a more gradual rise with an area under the curve before the peak comprising approximately 20% of the total counts. Further analysis indicated that this was due to inconsistent timing from the alpha pixels—some detection events were recorded later following a D-T fusion event than others.

In order to further investigate this problem a series of pulses was recorded using a digital oscilloscope. The minimum threshold amplitude to trigger recording was set to 80 mV, 100 mV, 120 mV and 150 mV and 1000 pulses were recorded at each setting. A wide variation in pulse amplitudes was observed, as demonstrated in Figure [3.5](#). Figure [3.6](#) indicates that there is a relationship between the pulse widths (FWHM) and amplitude, which gave a clue to the timing problem. The constant fraction discriminator (CFD) algorithm is resistant to variations in pulse amplitude but sensitive to changes in pulse shape.

To investigate this further the ORTEC CF 8000 CFDs were simulated in software to assign a time to each of the recorded pulses. A delay of 10 ns was used, matching the setting on the laboratory CFDs. The pulses were then filtered by amplitude and the standard deviation of the timing assigned by the CFD was plotted as a function of the filtering threshold (shown in Figure [3.7](#)). This plot identifies a minimum amplitude of approximately 315 mV above which the timing from the CFD is stabilized to a standard deviation of 2 ns. The cumulative distribution of these pulses was plotted to determine what fraction of pulses

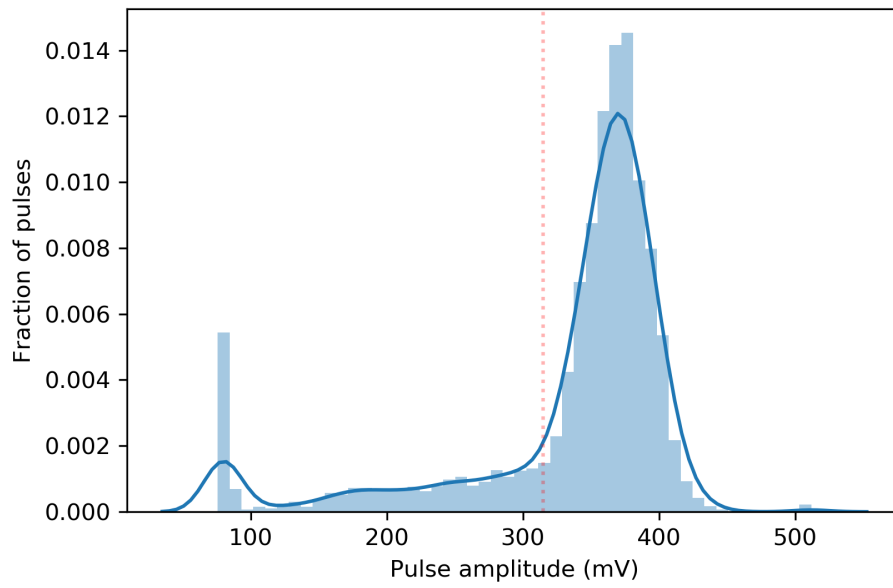


Figure 3.5: Amplitude of 1000 pulses recorded from alpha pixel *dtp08* with a minimum threshold of 80 mV. The cluster of pulses at the left corresponds to signal noise.

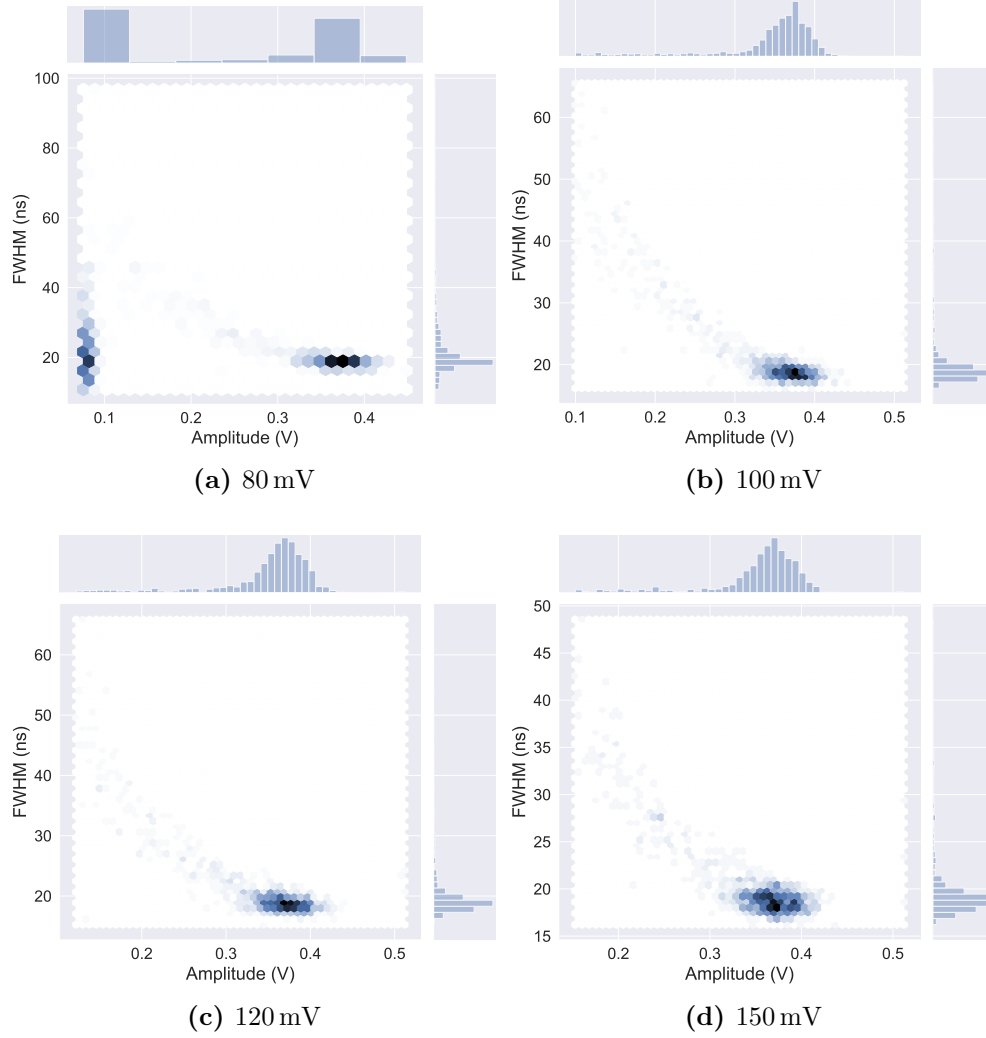


Figure 3.6: Pulse width vs. amplitude distribution for various triggering thresholds. Each shows an apparent tail of wider pulses at lower amplitudes.

would be rejected as below this threshold value (Figure 3.8). Approximately 20 % of the recorded pulses were found to be below this threshold amplitude. This was considered to be a reasonable tradeoff for the improved timing consistency.

3.4 Computed tomography and required accuracy

As stated above, the array alignment information is used for both noise rejection and the construction of two-dimensional images. Computed tomography requires the calculation of the inverse radon transformation of a sinogram. A sinogram is an image where the coordinates of each pixel correspond to a projection angle and a detector position. In the case of transmission imaging the intensity of each pixel is assigned based on the measurement of that pixel at that projection angle. This is equivalent to the Radon transform of the image:

$$\mathcal{R}_\theta f(s, i) = \int f(s \cos(\theta_i) - t \sin(\theta_i), s \sin(\theta_i) + t \cos(\theta_i)) dt, \quad (3.3)$$

where each projection s “corresponds to the collection of line integrals through f in the direction perpendicular to θ_i ” [30]. While equation 3.3 describes parallel-beam tomography, the same mathematics apply to fan-beam tomography, the technique used with the NMIS system. In this, instead of the rays projected through an object being parallel to each other, each has its origin at a common point. The inverse transformation assumes a mapping between each pixel and two known angles (the projection angle and the detector angle relative to the source). The lack of this information means that artifacts are introduced into the final processed image. Some of these artifacts are well understood: too few views, for example, introduces “streaking,” or lines where all of the information along a path comes from a single projection. The severity of streaking artifacts is illustrated in Figure 3.9.

Less literature exists regarding the distortions introduced by mis-positioning the detector relative to the source. In order to determine the effect of these misalignments some simulations were made using the astra tomography toolkit [31]. A simplified phantom object (figure 3.10a) was generated that shared some of the characteristics of the object that would eventually be modeled in MCNP—an annulus with a hollow core surrounded by a thin box.

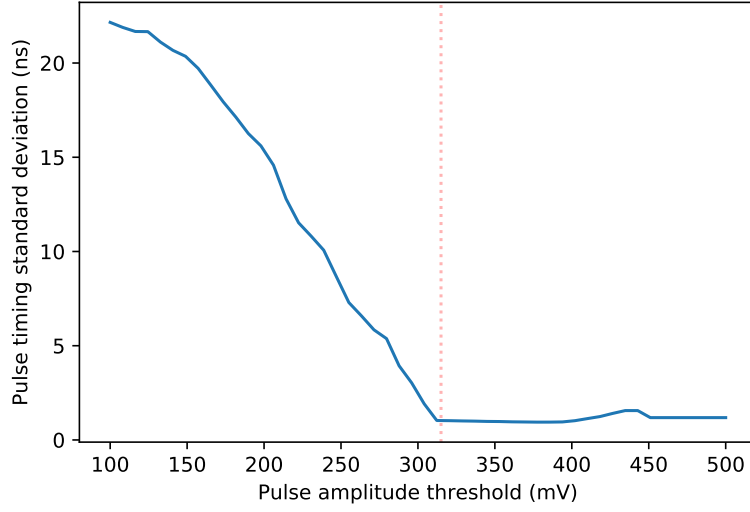


Figure 3.7: Standard deviation of 1000 pulses recorded from alpha pixel *dtp08* vs. threshold amplitude. Above approximately 315 mV the pulse timing becomes much more self-consistent.

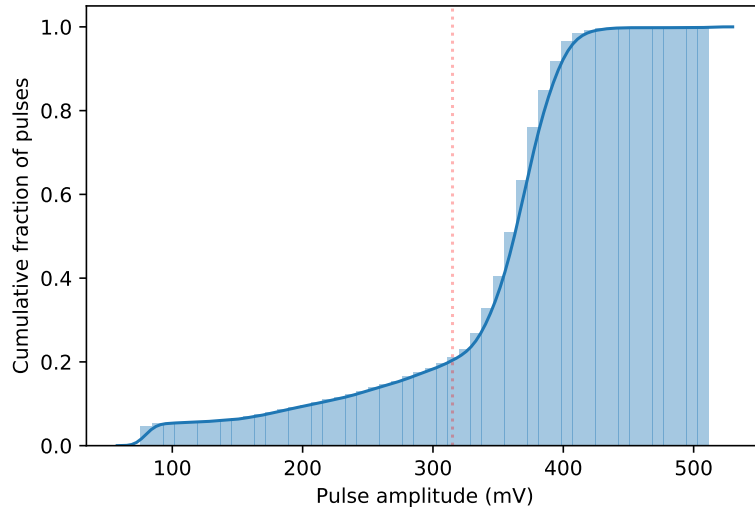


Figure 3.8: Cumulative distribution for amplitude of pulses from detector *dtp08*. A threshold of 315 mV rejects 20 % to 25 % of detected events.

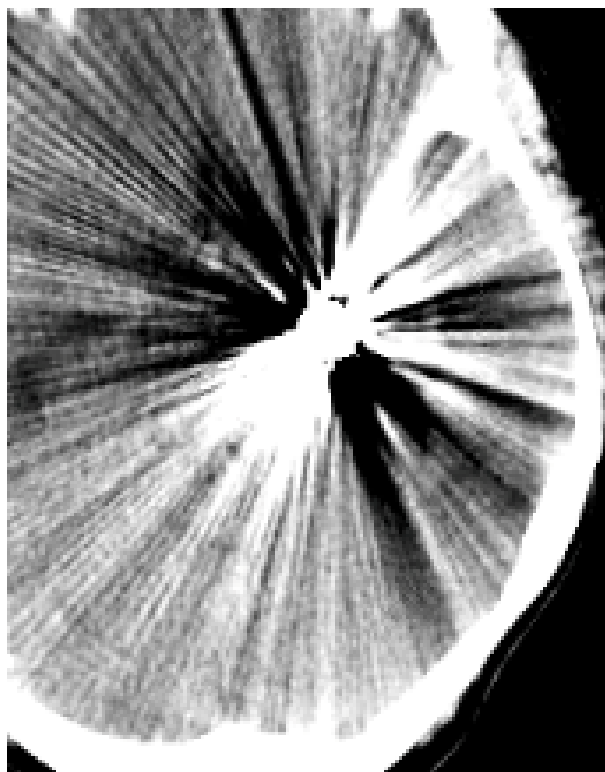


Figure 3.9: Streaking artifacts attributed to motion and undersampling [20].

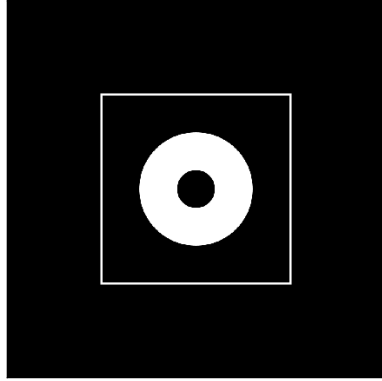
This image was transformed into a sinogram using a flat array of 32 pixels at 2.9 cm spacing positioned with some offset or rotation. The inverse transformation was then applied to the sinogram under the assumption that the array was in its nominal position, producing an image. The simultaneous iterative reconstruction technique (SIRT) was employed with 1000 iterations for each reconstruction as this produced the best compromise between computation time and image quality. The reconstructions are shown in Figure 3.10.

It is clear that a number of reconstruction artifacts are present in each projection. The low angular resolution of the pixels contributes to image artifacts. The low resolution, however, is a compromise with the geometric efficiency of the detector pixels. Since the neutron mean free path is low in the detector material (0.098 cm^{-1}), there is an inherent compromise between geometric efficiency (the likelihood of detecting a neutron given the size and orientation of the pixel) and resolution.

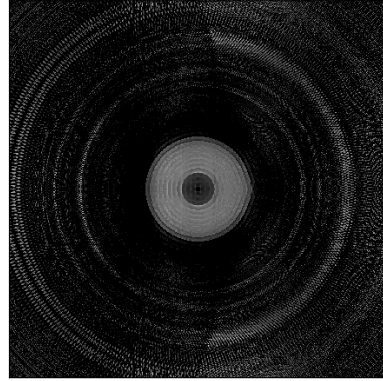
Figure 3.10b shows the smallest difference from the original phantom. The reconstructed image shows a difference in scale. A measurement of the outer diameter of the annulus gave a width of 27.81 cm and an inner diameter of 8.54 cm compared to the true values of 30 cm and 10 cm respectively. Though this represents an 8% difference in the ratios of these two measurements and a 10% to 15% difference in absolute measurements, the fact that there is an inner core inside of an outer annulus is visible. The reduction in contrast might affect annuli with similar inner and outer materials, however.

Figure 3.10c shows the largest deviation from the original phantom. The outer diameter of the reconstruction is 18.56 cm while the inner core is arguably missing. Identifying this object as a cylinder vice an empty annulus would be challenging using this information alone, as would assessing its size.

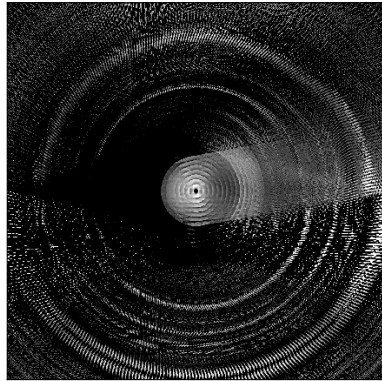
In order to quantify the difference between the true object and the NMIS reconstruction, the earth mover's distance was used. The earth mover's distance is a distance between two distributions corresponding to the minimum amount of work to transition from one distribution to the other. An analogy would be to treat each distribution as a pile of dirt with the height at each location equal to the pixel value at that location. One unit of work would be to decrease the value of one pixel by one and increase the value of the adjacent pixel by one while two units of work could either be decreasing the value of one pixel by



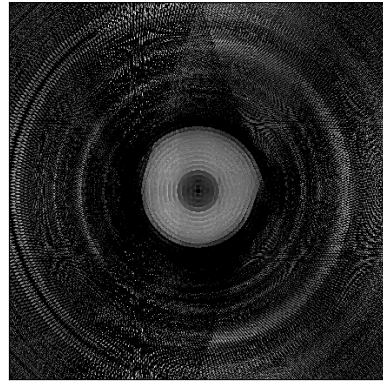
(a) Original image



(b) 15 cm closer to source



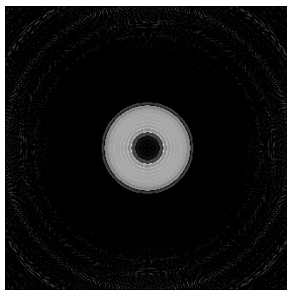
(c) 15 cm left of imaging axis



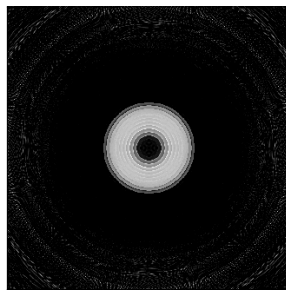
(d) 7.5° clockwise

Figure 3.10: Distortions introduced in image reconstruction from array positioning errors. Ring artifacts are a function of the comparatively small number of projections.

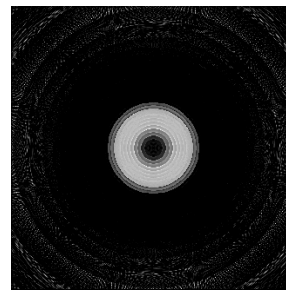
one and increasing the value of a neighbor two spaces away by one, or of decreasing the value of one pixel by two and increasing the value of the nearest neighbor by one. Rubner et al. [32] proposed this as a metric for assessing the distance between two images, specifically citing its connection to perceptual similarity between images. To verify that this property holds for tomographic reconstruction in the presence of array location errors, six images were reconstructed of sinograms of a simple annulus with the array laterally offset by 1 cm, 2 cm, 3 cm, 5 cm, 8.5 cm and 15 cm. These reconstructions are shown in Figure 3.11. The earth mover’s distance was then calculated between each pair of images, shown as a heat map in Figure 3.12. As desired, the reconstruction error is small when comparing images with similar lateral offsets (for example, 1 cm and 2 cm) and increases as the offset is increased. This indicates that the earth mover’s distance between reconstructed images preserves the ordering of errors induced by increasing offsets.



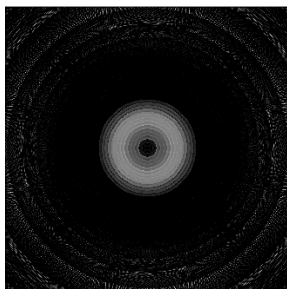
(a) 1 cm offset



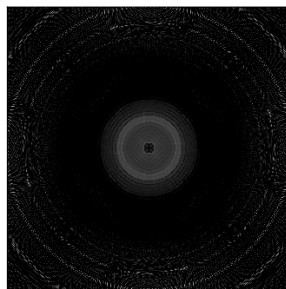
(b) 2 cm offset



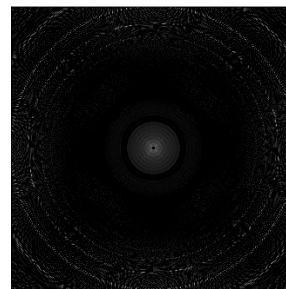
(c) 3 cm offset



(d) 5 cm offset



(e) 8.5 cm offset



(f) 15 cm offset

Figure 3.11: Reconstructions with various lateral array offsets.

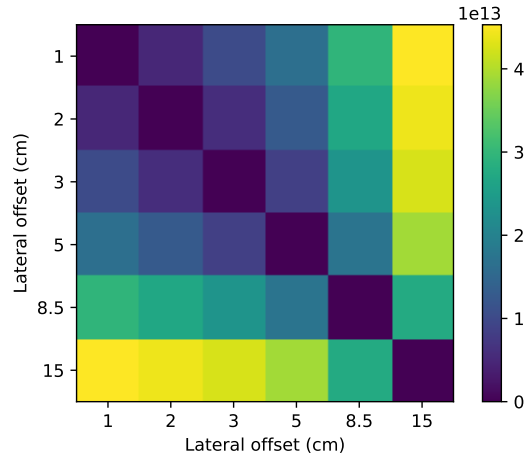


Figure 3.12: Earth mover's distance in arbitrary units for each pair of images from Figure 3.11. The figure is symmetric: both the bottom left and top right square show the distance between reconstructions with 1 cm and 15 cm. The distance increases monotonically with the magnitude of the offset.

Chapter 4

Description of Simulation

4.1 Design of Experiment

Eight materials were modeled for this experiment. Their linear attenuation coefficients (the amount of travel to attenuate $1/e$ of the incident particles) are shown in Figure ?? . Polyvinyl toluene is a highly scattering plastic used in scintillators. Comprised of hydrogen and carbon, it has a high microscopic cross section for 14.1 MeV neutrons but a lower density than other materials included in the simulation, bringing its macroscopic cross section to 10 cm^{-1} . Troxell [19] and Valkovic et al. [33] identify DT generators as useful for detecting and classifying explosives, so two such materials are included in these simulations. TNT and RDX are hydrogenous explosives with densities around 1.6 g/cm^3 and macroscopic cross sections from 0.13 cm^{-1} to 0.14 cm^{-1} . Two structural materials (carbon steel and aluminum) were included. These are commonly used in industrial applications and are likely to be employed in manufactured components where rigid supports are required. Aluminum is significantly less dense than steel (2.7 g/cm^3 vs. 7.85 g/cm^3) and has a macroscopic cross section for DT neutrons of 0.105 cm^{-1} (compared to 0.22 cm^{-1} for steel). Two dense materials (tungsten and depleted uranium) were included in the model. These materials are highly scattering for photons, especially in the x-ray energy domain. These were included to represent situations where the object could not be effectively imaged with x-rays, necessitating a different imaging modality. While they are less attenuating for high energy neutrons than for typical x-rays, they still are highly interacting for neutrons, with macroscopic cross sections of 0.33 cm^{-1}

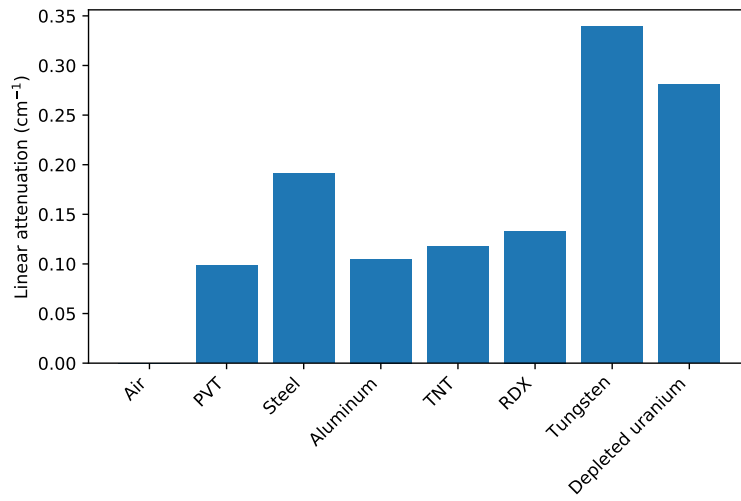


Figure 4.1: Linear attenuation coefficients for the eight simulated materials for 14.1 MeV neutrons. Four materials (PVT, Aluminum, TNT, and RDX) are within 0.035 cm^{-1} of each other.

and 0.28 cm^{-1} respectively. The density, isotopic makeup, and atom fractions for each of these materials were taken from [34].

The neutron array’s position was varied in three dimensions: x (along the axis between the generator and nominal position of the detector), y (lateral, or left/right when observing the position of the array from the generator), and θ (rotation of the array about a point centered on the detector’s front face). These offsets were selected from the range -15 cm to 15 cm , -15 cm to 15 cm , and -7.5° to 7.5° respectively. These values were selected based on conservative estimates of accuracy for a moderately-trained operator setting up the detector system. They were confirmed based on work by Troxell [19], who conducted a series of setup exercises using a neutron array mounted on a tripod and subsequently measured the accuracy using sub-mm accuracy laser distance sensors. His work indicated that the boundaries selected in this work encompassed the accuracy of a competent operator.

In order to capture the maximum input variation in a comparatively small number of samples, an orthogonal latin hypercube-based design of experiment was employed. The range of each parameter (δx , δy , $\delta\theta$, r_{inner} , r_{outer} , and inner and outer material) was established as described above and a series of random vectors whose component values were constrained within these ranges were generated. These vectors were selected such that no two points lay on the same hyperplane in this cube, i.e. no two sample points share more than one value in common [35]. This represents an improvement over both random and stratified sampling [36] and represents an effort to maximize variance over a sample space within a fixed number of samples. A quantity of 1000 samples was generated in this arrangement. Of these, 2 samples suffered from errors due to computer reset, leaving a total of 998. Approximately 15 % (157) were separated and used to test the performance of the trained networks leaving 841 for network training and validation. The distribution of the input parameters for the training, validation, and testing sets was examined qualitatively and determined to have the same uniform distribution. Because the inner radius of the annulus was required to be less than the outer radius, the uniform values generated were expressed as a fraction of the outer radius. This resulted in the inner radius following an approximately $1/r^2$ distribution. As the network was not expected to predict this parameter and the same distribution was followed for both the training and testing sets, this was deemed acceptable.

4.2 MCNP Modeling

In order to accurately simulate the real-world measurements using the portable NMIS system, the Los Alamos National Labs software tool MCNP (Monte Carlo N-Particle) [37] was employed. This software attempts to simulate the neutron field by generating particles and tracking the material through which these particles travel. At each unit of distance traveled or boundary between two materials, the program calculates the probability of that particle interacting along the previous path. By simulating many such particles and tracking events that the user specifies, the simulation should converge to the distribution of those quantities that approximates the underlying physical system.

The geometry was specified for a steel container with 0.5 cm thick walls and external dimensions of 92 cm deep by 2 m wide by 2 m tall. This meant that the neutrons generated would always have some amount of scattering material in the path between the generator and the neutron detectors, producing a scattered neutron field and x-rays from inelastic collisions. Inside of the box an annulus was simulated, roughly centered in the box. The inner and outer radii of the annulus were varied between 1 cm to 30 cm and 0 cm to 29 cm respectively, with an inner radius of 0 cm representing a solid cylinder of the outer material. The materials for the outer annulus and inner core were also selected at random from all of the materials listed above with the exception that the outer annulus could not be made of air.

The array of neutron detectors was modeled to match the description of the NMIS system in Chapter 3 with the neutron detectors arranged in a single line having a pitch of 2.9 cm. An example geometry is shown in Figure 4.2. The plastic holders for the neutron detectors were not modeled nor were the supporting tripods or room, all of which would be present for an actual measurement.

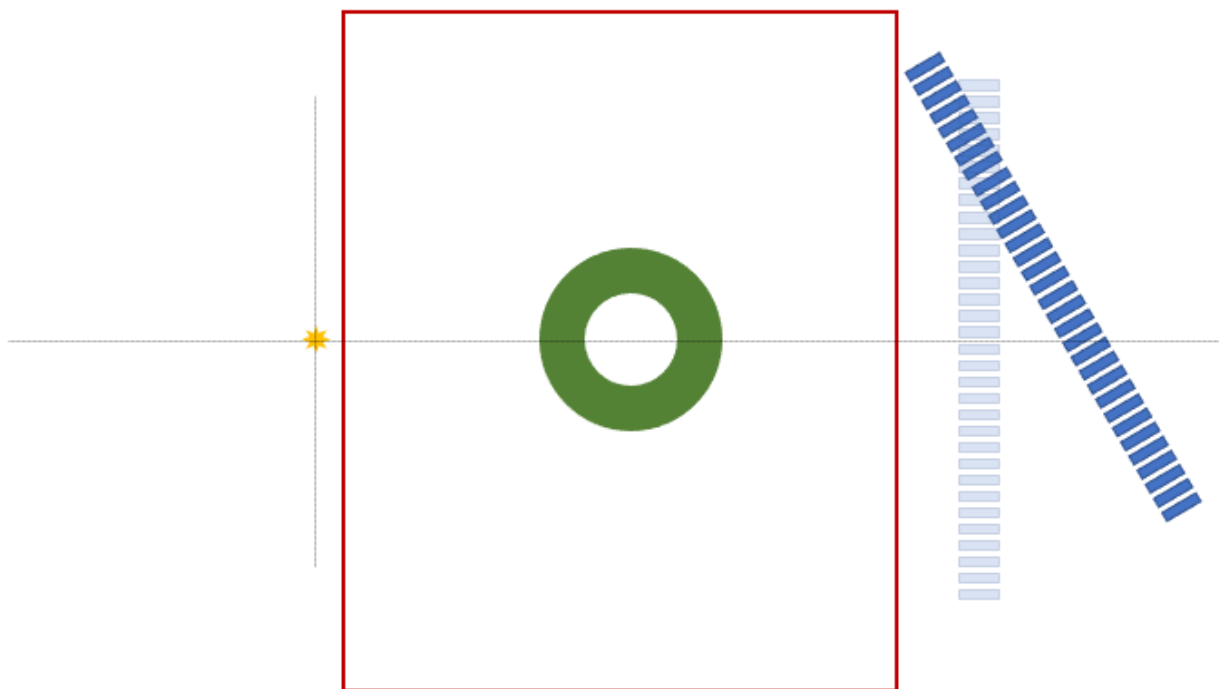


Figure 4.2: Example of model geometry. The nominal position of the array is shown in light blue. The array (dark blue) shown here has been translated in the $+X$ (rightward), $+Y$ (upward), and $+\theta$ (counter-clockwise) directions. The magnitude of the array offsets has been exaggerated for illustration.

4.3 Angular emission distribution

The fifteen alpha detectors integrated into the the ING-27 generator are each 6 mm by 6 mm at a distance of 100 mm from the center of the deuteron beam target spot. This gives them each an average geometric efficiency of approximately 4.3×10^{-3} . In order to improve computational efficiency, only neutrons that corresponded to an alpha detection event were simulated, meaning that the simulations converged on order of three orders of magnitude faster. Previous work with related generators has attempted to simulate the DT reaction and subsequent kinematics from first principles [38]; for this work a more direct measurement was employed. The neutron detector array was used to measure counts at 47 heights ranging from 62.38 cm to 99.85 cm while the generator was fixed at 78 cm. At each height the generator was run for 25.6 s for each of four subsampling positions. Neutron counts were correlated to an alpha pixel by binning them in time.

These values were used to fit angular distributions to each alpha pixel in both the horizontal and vertical angles. Historically the functional form used was the sum of two Gaussian functions [15]: one to account for the large peak and the other for the much smaller wings. While this accurately fits the peak of the distribution, it does not precisely fit the “wings” of the distribution. Instead, a Voigt profile was used. The Voigt profile is the convolution of a Lorentzian and Gaussian distribution. A Lorentzian distribution is described by the X-intercept of a ray emitted from a point (x_0, γ) , described by the probability density function

$$p(x|\mu, \gamma) = \left(\pi\gamma \left[1 + \left(\frac{x - \mu}{\gamma} \right)^2 \right] \right)^{-1}, \quad (4.1)$$

while the probability density function of a Gaussian distribution is described by

$$p(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp \left[-\frac{(x - \mu)^2}{2\sigma^2} \right]. \quad (4.2)$$

A physical intuition for why the Voigt profile might better model these curves arises from the fact that the intensity of the deuterium beam incident on the tritium target is likely to be normally distributed about its center while the direction of the neutron following a fusion event should be uniformly distributed. Two channels (dtp14 and dtp15) had insufficient

correlated counts for fitting. The alpha pixels integrated in the ING-27 cover an arc of approximately 54° , centered at 3° . The neutron pixels, however, only cover an arc of 46.2° , centered about 0° . As a result, the neutron emission cone correlated to alpha pixel dtp15 is absent and the cone correlated with alpha pixel dtp14 is nearly so. Since the position of the neutron detector would be varied enough to enter these cones, their properties were modeled by performing a linear regression of the mean angle, full width at half max, and intensity vs alpha pixel number. The emission angles for dtp14 and dtp15 were generated by sampling from this regression equation and the results were validated qualitatively. Since the dtp14 partially intersected the neutron detectors, it was possible to verify that this technique provided results that were at least reasonable (see Figure 4.3).

The same functional form was used to fit the vertical angle ϕ . Since the neutron arm was moved vertically, the counts for each height were normalized to a distance of 115.1 cm using a $1/r^2$ relationship before fitting. Referring to Figure 4.4, the ϕ fits are more obviously asymmetric, which is ascribed to the presence of neutrons scattered into the detector pixels from the floor of the room. The parameters of these fits (in horizontal, θ , and vertical, ϕ , angle) are given in Table 4.1 and are shown in Figures 4.3 and 4.4.

Neutrons are detected in the NMIS system using blocks of polyvinyl toluene (PVT) scintillator. This material detects neutrons by having a high concentration of hydrogen nuclei per unit volume. The incident neutron scatters off of the hydrogen, imparting some fraction of its energy. The hydrogen nucleus decelerates in the plastic, depositing energy as it goes. This energy is emitted by fluors integrated into the plastic with a characteristic wavelength, rise time, and decay time. Photons can also scatter in the scintillator material registering events. In the scintillator material used in NMIS it is impossible to differentiate between an event caused by a neutron scatter from a pulse caused by a photon scatter.

Since alpha particles are relatively easy to detect, the simulation focused on neutron and photon detections that followed the detection of an alpha particle. One MCNP simulation was run for each of the fifteen alpha pixels for each detector array configuration. The initial direction for the neutron was selected using normalized versions of the curves described above. The MCNP program then transported the neutrons, keeping track of their energy and what secondary particles were generated by them. MCNP tracked the average length

Table 4.1: Fitting parameters for emission probability vs θ

Alpha pixel number	A (arb.)	μ ($^{\circ}$)	σ ($^{\circ}$)	γ ($^{\circ}$)
dtp01	67445.07	18.78	0.63	0.63
dtp02	62298.17	15.86	0.60	0.60
dtp03	85129.21	12.95	0.70	0.70
dtp04	76060.03	9.89	0.72	0.72
dtp05	95465.28	6.58	0.84	0.84
dtp06	85634.13	3.32	0.78	0.78
dtp07	93312.13	0.08	0.83	0.83
dtp08	92919.05	-3.25	0.92	0.92
dtp09	76251.28	-6.61	0.91	0.91
dtp10	89965.98	-9.84	1.00	1.00
dtp11	74117.69	-13.08	0.97	0.97
dtp12	81097.46	-16.06	1.09	1.09
dtp13	26954.65	-19.05	1.20	1.20
dtp14	19466.27	-22.04	1.32	1.32
dtp15	13367.66	-25.02	1.43	1.43

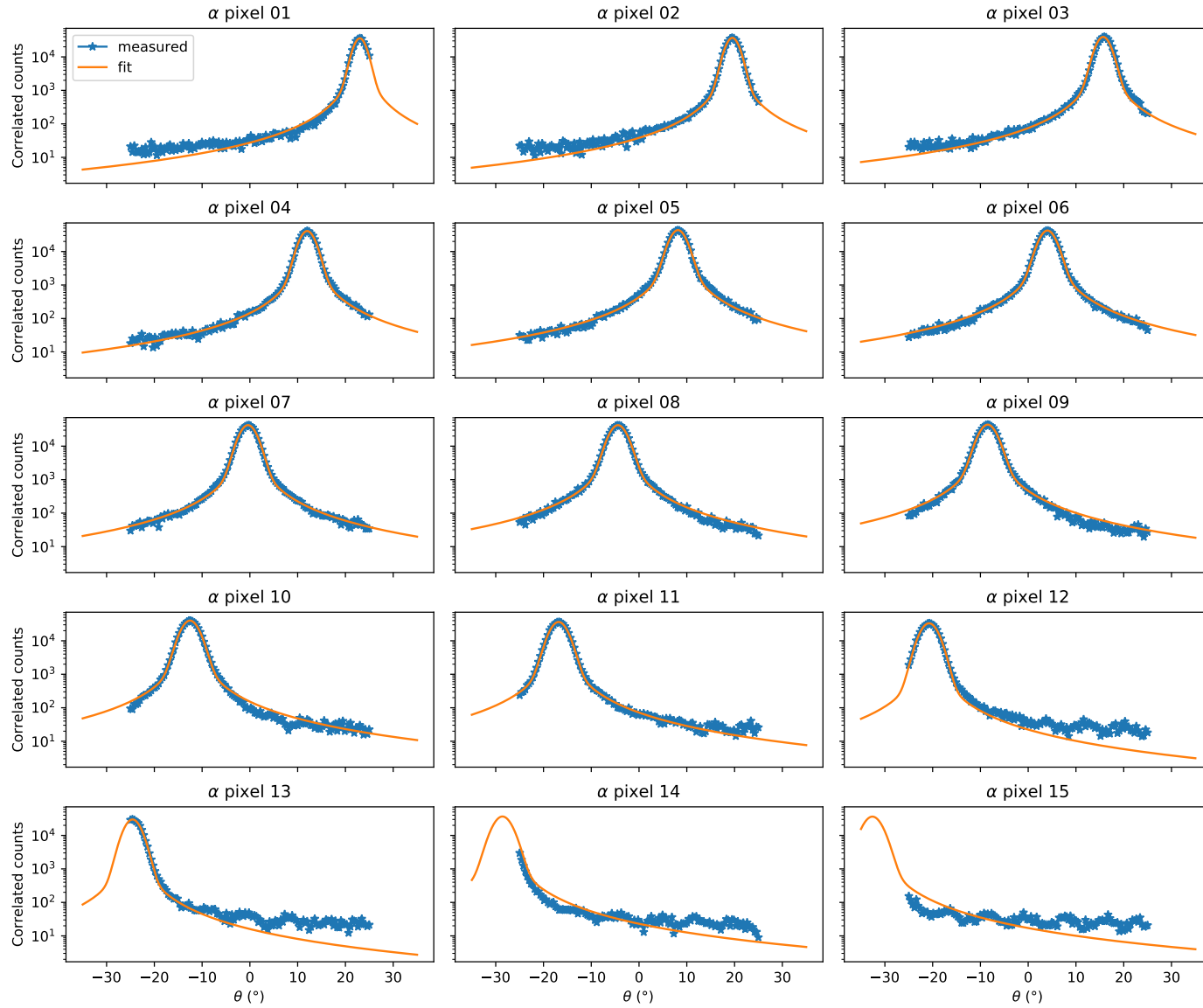


Figure 4.3: Number of counts recorded as a function of horizontal angle θ associated with each α pixel. Measured counts (blue stars) are counts in each neutron pixel that have been correlated to a detection in the relevant alpha pixel. Orange lines are curves fit to the measured counts.

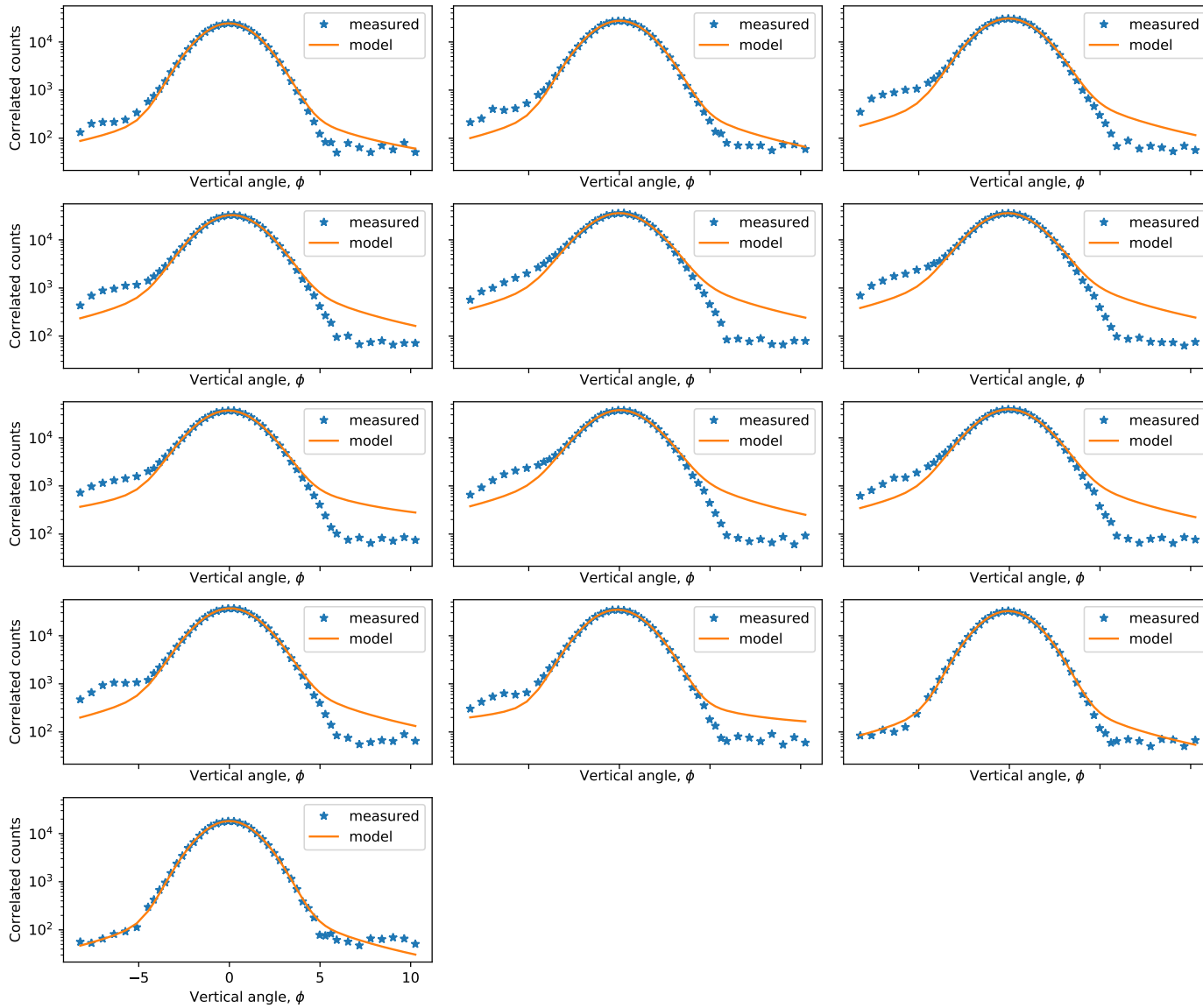


Figure 4.4: Number of counts recorded as a function of vertical angle ϕ associated with each α pixel. The similarity of the curves indicate minimal differences between the alpha pixels with respect to vertical angle. The asymmetry between positive and negative angles was attributed to neutron scatter off of the floor of the room.

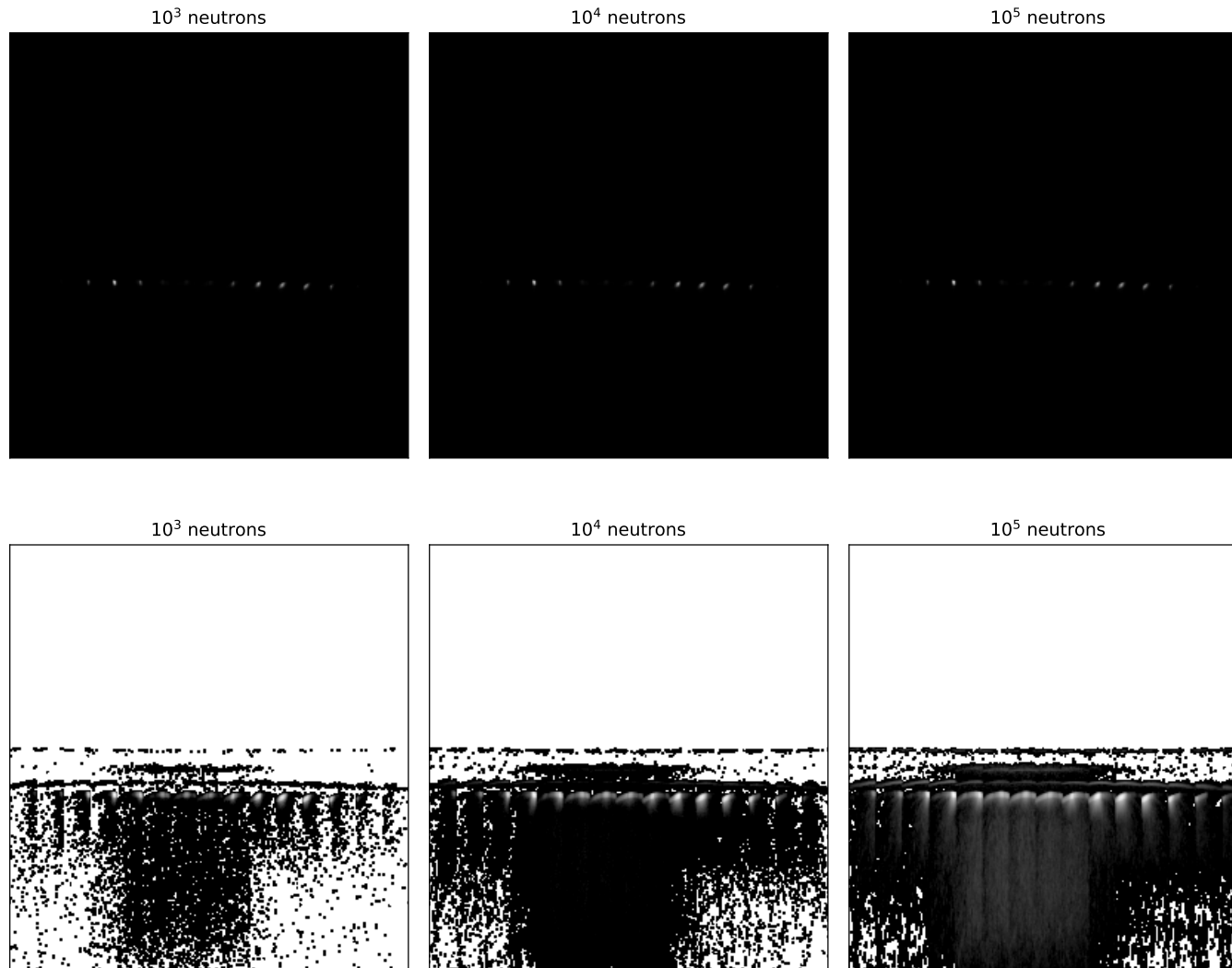


Figure 4.5: Example simulation output, bottom row in log scale. The simulated annulus was 17.22 cm in diameter with an inner core of TNT and a 2.32 cm thick outer ring of depleted uranium. As increasing numbers of neutrons are simulated additional features become apparent.

of travel through the neutron detectors as a function of time, then multiplied this value by the energy-weighted probability of scattering in the pixels. The fifteen MCNP input files each produce a text-formatted output giving the probability of a neutron- or photon-induced scatter event (given a detection in the relevant alpha pixel) in of the 32 neutron pixels versus time in 1 tick intervals.

A scatter in a neutron detector produces scintillation photons which must be collected in order to register as an event. This introduces a delay between the energy deposition by the scattered particle and the time recorded by the time to digital computer. In order to approximate this delay, the event probabilities were blurred in time, using the distribution of times assigned by the simulated CFD described in chapter 3. These values represent the probability of registering an event at time t in neutron detector i given a detection in alpha pixel j at time 0. The rate of alpha detections is treated as deterministic-it is a property of the geometry of the generator and alpha pixels, the beam voltage, and the beam current¹. Using the same calibration data that was used to find the angular distributions above, an expected number of alpha detections per second is scaled by the simulated measurement time, giving n_j detections in pixel j . The probabilities above are then used for a binomial sample with n_j trials to determine the number of counts in each time bin for each pair of alpha and neutron pixels. The randomized nature of this sampling permitted the generation of multiple valid measurements for a single MCNP run. This meant that the neural network could learn not to falsely respond to normal poisson measurement variation. An example of the output for simulations of 6 s, 60 s, and 6000 s is given in Figure 4.5. Longer measurements show increased detection of scattered neutrons and gammas.

The process of generating a simulation for one position of the neutron array with an object present is illustrated in Figure 4.6. A python script (`render.py`) takes array offsets (δx , δy , and $\delta \theta$) and object parameters (inner and outer radii and material) and uses these to generate an MCNP input file. An example of a rendered input file is presented in Appendix 7. These are combined at runtime with the normalized angular probability curves discussed above to generate fifteen text-based output files (`mctal` file) that gives the probability (given a

¹This is true for time scales on the order of months, but not on the order of years. As the generator is used the tritium in the target becomes depleted, lowering the neutron/alpha production rate.

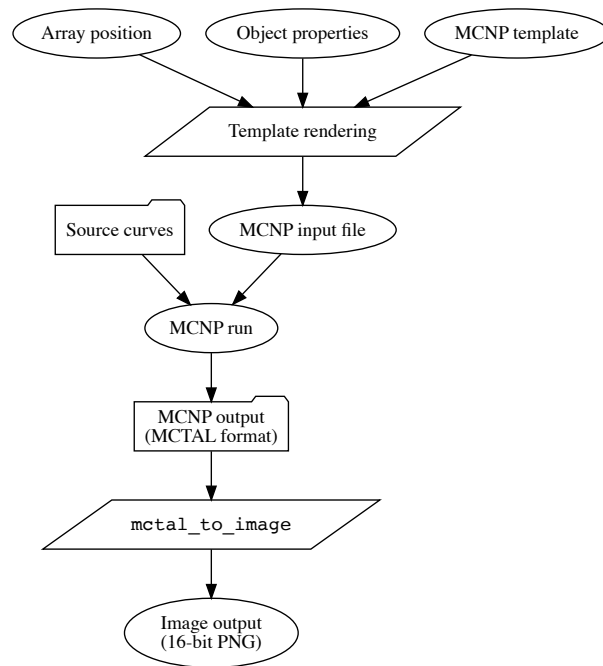


Figure 4.6: Conceptual workflow for simulation.

detection of a particle in the respective alpha pixel) of a neutron or photon scattering in each of the neutron detectors given a detection in one of the fifteen alpha pixels as a function of time. A final program (`mctal_to_image`, included as Appendix A) combines these probabilities and performs a binomial sample with n trials for each time/neutron pixel/alpha pixel combination, where n is determined by the generator run time. The program then stores these values as a compressed image file suitable for training the neural network described in Chapter 5.

4.4 Quasi-curved array

In order to test whether this method would work in different array configurations, a second neutron array was simulated. This array consisted of twenty-eight neutron detector pixels, each of which matched those used in the linear array described above. This array matches that described by Troxell [19]. The same detector response blurring kernel was used as in the linear array. The array was designed such that each pixel subtended the same arc when at the nominal position. With the array misaligned, however, this design goal was no longer satisfied, making this a reasonable test for the general applicability of the final neural network architecture.

4.5 Differences between simulation and reality

The MCNP simulation of the NMIS system departed from the real detector system in several ways. The ING-27 generator is a 4π source, meaning that it emits both neutrons and alpha particles in all possible directions. In order to speed up the MCNP simulation, only neutrons corresponding to an alpha detection were simulated, as described above. These angular probability curves were fit to a void measurement to minimize the impact of neutrons scattering in the detector stage, but scattering between the floor and neutron detectors was still present, as was scattering between neutron detector pixels. The latter of these has the effect of doubly sampling these particles in the simulation, causing them to be overrepresented in the simulated measurement. The simulation did not include the production of detector

pulses in either the neutron or alpha detectors; instead these were accounted for using a blurring kernel formed by averaging across a number of pulses. This destroyed any possible correlation between the time of arrival in the alpha pixel and the angle of emission. It also had the effect of eliminating the possibility of observing a neutron detection event from a subsequent source event less than 100 ns later, but in laboratory measurements at the simulated source levels this was rarely observed.

Another difference is in the minimum energy threshold required to produce a pulse in the detector pixel. The detector electronics system only records pulses that exceed a certain amplitude threshold. The amplitude of the pulse is a function of the number of scintillation photons produced in the detector volume and the time for those photons to be collected in the photomultiplier tube. The number of scintillation photons is, in turn, a function of the energy deposited by the neutron and the particle which scattered the incident neutron (for example, a hydrogen nucleus with 1 MeV produces 2.64 times more photons in EJ200 than a carbon nucleus with the same energy). MCNP gains computational efficiency by calculating an average of these values over the particles produced. As a result, its intermediate calculations do not necessarily conserve physical quantities such as total energy or boson number for non-deterministic calculations. Because of this, all neutron and photon scatter events within the neutron pixels were considered to have produced a detector event. This had the effect of overcounting detector events. This effect was partially mitigated by removing from the simulation any neutron whose kinetic energy was below 1 MeV (the threshold used for minimal energy deposition for scatter off of hydrogen in previous MCNP simulations of NMIS [19]). Further mitigation was not employed as the number of counts per neutron pixel matched to within 25 % of the values from the calibration measurement described in Section 4.3 and other methods had the effect of drastically slowing the simulation.

Chapter 5

Neural Network Analysis

5.1 Data pre-processing

In order to normalize the data, each pixel value was scaled such that the largest value was equal to 1.0. This normalization was applied to each image individually. This had the effect of removing some of the variability in the simulated measurements related to measurement time, effectively matching the network input to a count rate rather than to absolute number of counts. Additionally, it emphasized the difference between neutron pixels which might have been harder to differentiate in cases where all of the neutron pixels were partially obscured. The offset values were scaled and re-centered such that the training set had a mean of 0.0 and unit variance. The test set was scaled with the mean and variance from the training set.

5.2 Learning rate optimization

Arguably, the most important hyperparameter in neural network training is the learning rate. In each training step, each network weight w_i is updated according to equation 1.10. If the learning rate is too low, the training process will take an unacceptably long time or may converge at a local minimum and never reach the global minimum loss. If it is too high, the network may oscillate about a minimum loss value; higher still and the training process may cause loss to increase as training proceeds. This is illustrated in Figure 5.1. A cyclic

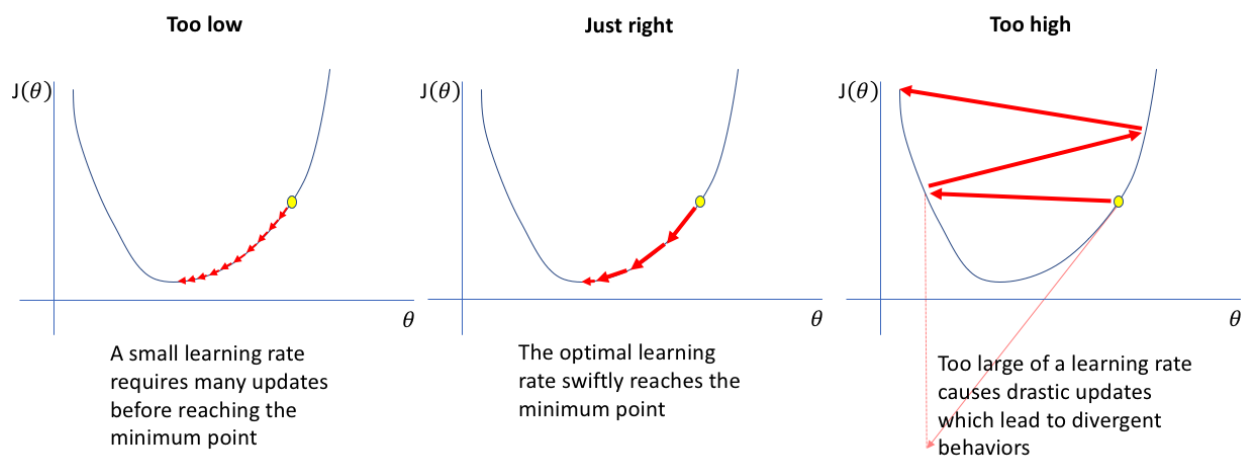


Figure 5.1: Consequences of poor learning rate choices. Here, J represents the network cost as a function of the trainable parameters θ . The network updates for a small learning rate will converge slowly (left) while high learning rates may oscillate about a minimum error. Image reproduced from [8].

learning rate provides a compromise between these two extremes. The network is initialized and trained using a low learning rate that is steadily increased. Some data are withheld from the network during weight updates; these data (referred to as validation data) are instead used to evaluate the network performance during training. The loss function is independently evaluated at the end of each training epoch, or one cycle of the network seeing each sample in the training set. The loss on the validation set is plotted versus this learning rate. A typical result is a reduction in loss at an increasing rate followed by a plateau (where small changes in learning rate have little effect), followed by a sudden increase in loss. The highest learning rate for the network is the one with the most negative slope on this graph [39]. This learning rate is high enough to step over local minima. It may, however, be too high to converge on a true global minimum as well. To address this, the learning rate is reduced over the course of training. This may be performed on a scheduled basis; an alternative method is to reduce the learning rate when the network begins to under-generalize as indicated by an extended period without reduction in the validation loss. A number of learning rates were examined as a part of this work, with the optimal maximum value found to be around 1×10^{-3} and a minimum of 1×10^{-5} . During training, each time the validation error did not improve after 20 epochs (a cycle where the network has been updated with every value from the training set once), the learning rate was reduced by a factor of 5. Validation set loss is used to ensure that the network is learning the true distribution of the underlying data rather than fitting the specific noise patterns in the training data.

Neural networks are sometimes reputed to require significant amounts of training data in order to converge to a useful, general solution. For some tasks the required training volume data can be reduced by using a network trained to perform a related task. Goodfellow, *et al*, describe this as a regularization method conditioned on the prior belief that “among the factors that explain the variations observed in the data associated with the different tasks, some are shared across two or more tasks” [9]. Baxter demonstrated that the shared layers could enforce a more general internal representation for the learned tasks, effectively increasing the accuracy for each of the regression outputs [40]. Conceptually, this takes advantage of the fact that a neural network builds up features over the course of several network layers. Early layers respond to primitive features (e.g. color gradients or edges

of a certain angle) [8]. This means that a pre-trained network can provide benefits even in seemingly unrelated tasks. The benefits are especially pronounced when a significantly larger volume of training data exists for the pre-training task than for the final task to which the network will be applied [9], as it does for published networks trained on the ImageNet recognition task (a database of over 12 000 000 images labeled with over 12 000 categories [41]).

In the case of this task, however, there was insufficient overlap between the information in ImageNet and the output of the NMIS simulation. As shown in Figure 5.2a, the training error did decrease more steeply when the network was initialized with pre-trained weights. This was a result of the network overfitting the training data, as the validation error reached a plateau at approximately 100 epochs and remained constant. The network with randomly initialized weights, by contrast, converged more slowly and showed a measure of instability (evidenced by the irregular spikes in the loss plot in 5.2b), but continued to improve with a final loss value comparable to that of the pre-trained network.

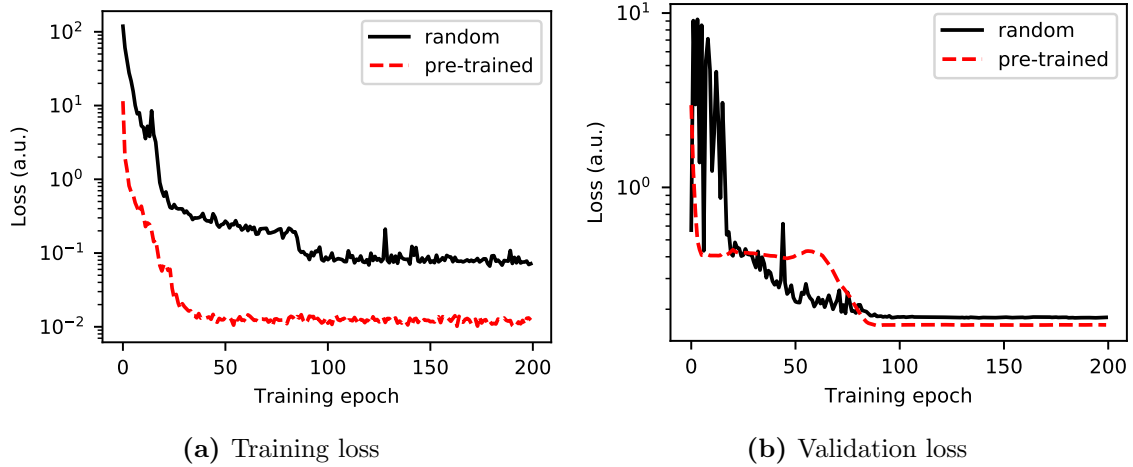


Figure 5.2: Comparing the use of pre-training on a general image classification data set.

Table 5.1: Network performance as function of number of layers (depth)

Network Depth	Training Time (s)	Time to Stability (s)	Training loss (arb)	Validation loss (arb)	Free Parameters	$\delta\theta$ RMSE ($^\circ$)
101	21032	93	0.0009	0.0230	43013633	1.34
50	3050	>500	0.0062	0.0230	2560000	1.35
10	2010	152	0.0226	0.0510	547633	1.33
8	2103	100	0.0662	0.1638	1160433	2.11
6	2108	30	0.3989	0.4118	3724997	4.38

Chapter 6

Results

6.1 Error in prediction

The final regression network was evaluated using the 157 samples generated using the same process as the training data. The distribution of the seven input parameters was examined and determined to have summary statistics comparable to the training and validation data. The network gave root-mean-squared errors (RMSE) of 0.49 cm, 0.66 cm, and 0.66° for δx , δy , and $\delta \theta$, respectively.

The final network had a small systematic bias when predicting the lateral (y) offset (distribution of residuals shown in Figure 6.1). The source of this bias is unclear, but a possible explanation is the low resolution of the pixels. The systematic bias is small ($\mu \approx -0.13$ cm) (corresponding to 0.06°), which is only 5 % of a pixel width. Since each alpha coincidence cone covers approximately 4 to 5 neutron pixels, this should have minimal impact and was therefore deemed acceptable. Based on the results reported in [22], this should be sufficiently accurate for tomographic reconstruction.

6.2 Analysis of errors

The distribution of errors in the test set was examined in an attempt to identify trends that might suggest bounds on the applicability of this method. In this case, however, no correlation between object size (inner and outer radius, see Figure 6.2) and error was

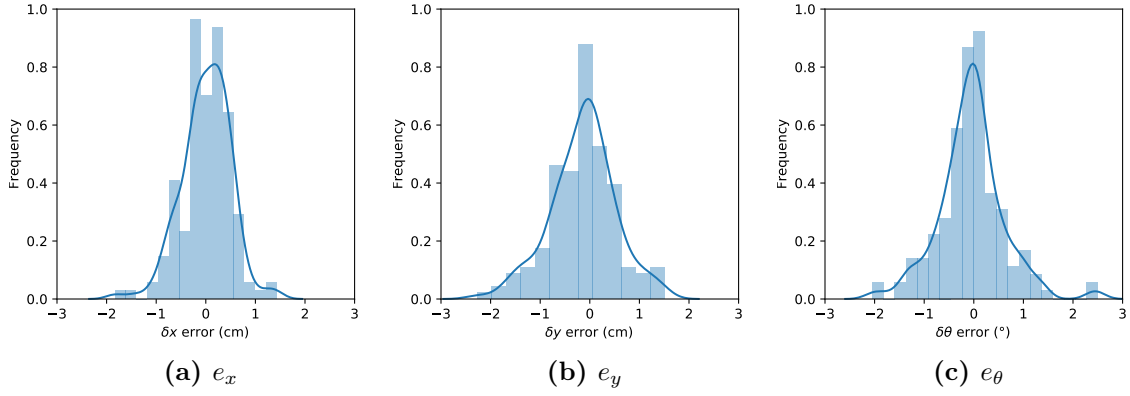


Figure 6.1: Distribution of errors in the final network. While the distribution in Figure 6.1a appears multi-modal, this is a consequence of histogram binning.

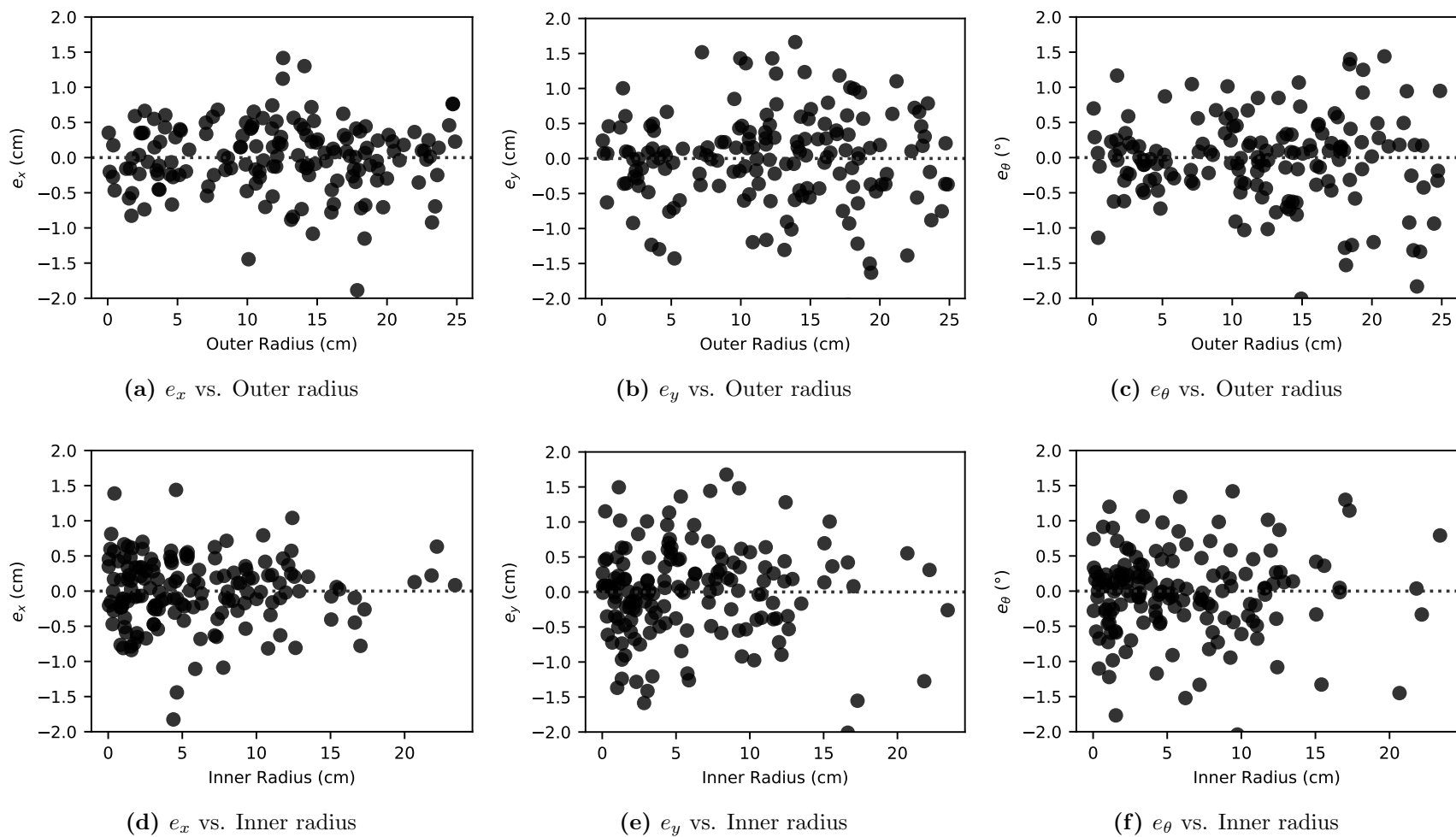


Figure 6.2: Dependence of error on radius of object imaged.

found—the network was accurate across the entire testing range, even when no unobstructed path existed between the source and any neutron pixel. Similarly, no relationship was found as a function of material, even when large cylinders of highly scattering material (depleted uranium and tungsten) were present, as is shown in Figure 6.3.

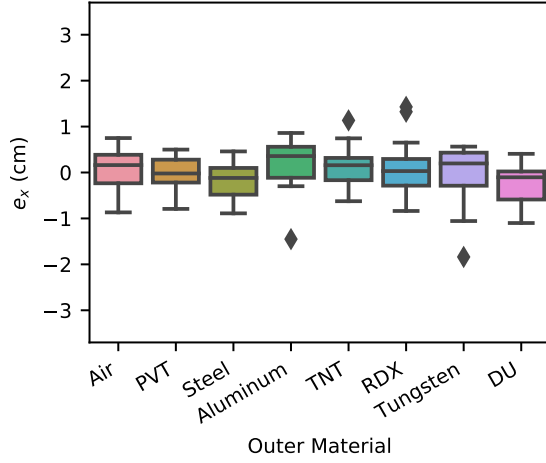
6.3 Measurement time sensitivity

Measurement times of 30 s, 60 s, 120 s, 300 s and 600 s were simulated by sampling from the binomial distribution in proportion to alpha count rates scaled linearly with time. The network exhibited good invariance to simulated measurement time, as shown in Table 6.1. The measurement error was extremely consistent—less than the precision of the table above. The simulated measurements were checked to ensure that their distribution was consistent with expected Poisson error. While the individual time and pixel bins did show variations consistent with Poisson error, it is possible that the network responded to the overall distribution of errors across either the entire input image or that one of the convolution kernels employed by the network served as a local average, denoising the image as a part of the network’s overall structure. Nazaré et al. [42] explored the ability of convolutional neural networks to generalize across different noise distributions than those present in their training data and determined that some networks exhibited this ability. Zhang et al. [43] went further, showing that a deep convolutional network employing residual connections, rectified linear unit activation functions, and batch normalization (all of which are features of the ResNet architecture employed in this work) can use the information in the entire image to separate signal and noise.

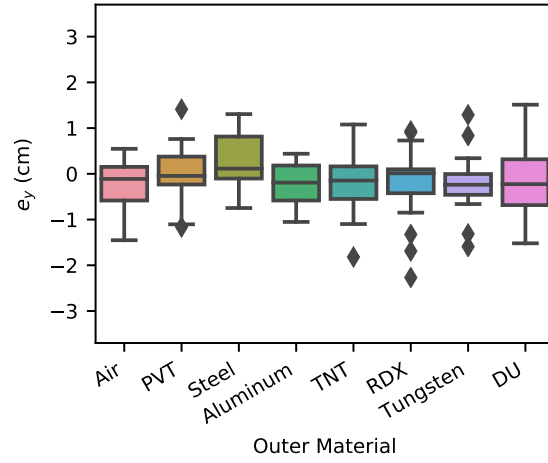
The errors for the alternative, quasi-curved array were comparable to those for the straight array. Residuals for δx and δy had mean values at 0 cm and standard deviations of 0.70 cm and 0.88 cm, respectively. Some systematic error was observed in the rotational offset prediction. The error in these predictions was still centered at 0°, but showed a tendency to under-predict the magnitude of large offsets. This drove the standard deviation of the residual to 1.0°. This difference was attributed to the edge pixels subtending a smaller arc as compared to the straight array. These edge pixels presented less variation with respect

Table 6.1: Error in predictions vs simulated measurement time.

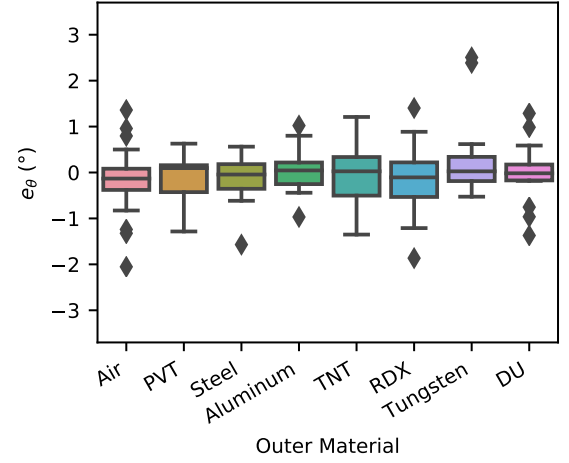
Simulated Measurement Time (s)	e_x (cm)	e_y (cm)	e_θ ($^\circ$)
600	0.4912	0.6479	0.6776
300	0.4904	0.6485	0.6774
120	0.4904	0.6485	0.6774
60	0.4904	0.6485	0.6774
30	0.4904	0.6485	0.6774



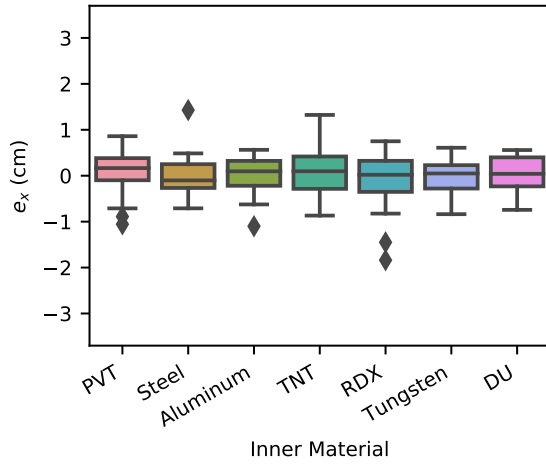
(a) e_x vs. Outer material



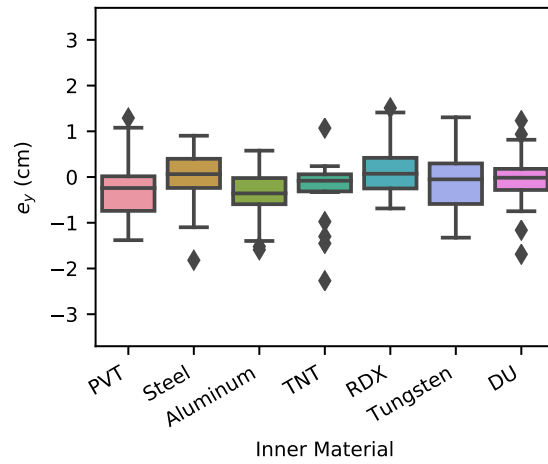
(b) e_y vs. Outer material



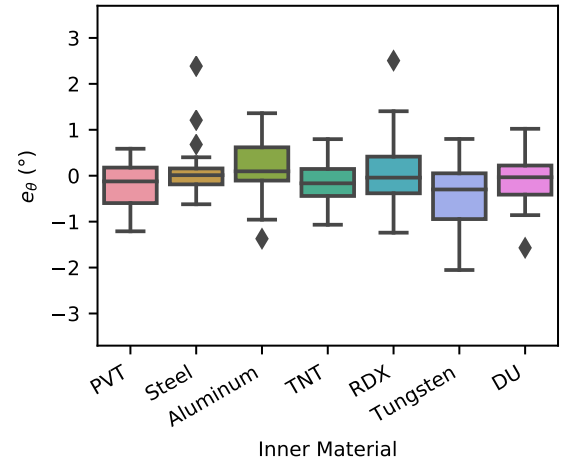
(c) e_θ vs. Outer material



(d) e_x vs. Inner material



(e) e_y vs. Inner material



(f) e_θ vs. Inner material

Figure 6.3: Dependence of error on material of object imaged.

to rotational offsets. The overall array also covered a smaller total volume and subtended a smaller arc (due to the lower number of pixels).

6.4 Tomographic reconstruction

As discussed in Section 1.3, image reconstruction requires knowledge of the array position both to determine the time window to classify detector events as resulting from transmitted neutrons versus scattered neutrons or photons and to perform the tomographic reconstruction from the reconstructed sinogram. Image reconstruction was performed to determine whether the neural network was sufficiently accurate. The reconstruction process is illustrated in Figure 6.6. The simulated measurement was scaled and used as an input to the neural network model to estimate the position of the simulated array. This position was used to determine the time and angle bounds corresponding to transmitted neutrons for each neutron pixel. The time bounds were determined by calculating the time of flight from the source to the front and back edge of the neutron pixel. In order to account for timing error, counts arriving after 1.5 ns before the time of flight to the front edge of the pixel to those arriving 1.0 ns before the time of flight to the back edge of the pixel were included; other counts were rejected. A simulated measurement with no object present was used to determine the angle bounds for each alpha pixel. Each neutron pixel was assigned an angle matching its center. A Gaussian function was fit the number of counts recorded at each angle for each alpha pixel. The angular bounds for each pixel were then set at $\pm 1.9\sigma$, a value selected to most closely match the calibration measurement described in Chapter 4. Pixels that partially overlapped these boundaries were scaled in proportion to the area of overlap with the angles selected above.

Once the count rate corresponding to each pixel was determined by filtering detected events outside of the bounds determined in the previous paragraph, a void simulation was performed in MCNP with the array at the position predicted by the neural network and the void count rate at this position via using the same time and angle boundaries. When combined with the angle from the source to the center of each pixel this produced two plots of count rate vs angle: one with and one without the object present, as shown in Figure 6.4.

The logarithm of the ratio of the void count rate to the object-present count rate was then calculated in accordance with Equation 1.1 to determine the attenuation between the source and the neutron pixel as a function of angle (shown in Figure 6.5). Since the simulated object was rotationally symmetric, a sinogram was formed by repeating this plot for each simulated projection through the object.

This attenuation information was then used with the simultaneous algebraic reconstruction technique (SART) to generate images of the test object. These were compared to the original image, as were reconstructions generated from the idealized sinograms (formed by ray integrals from the source to the center of each pixel) and from sinograms made from the correlation plots assuming that the array was at the nominal position. These reconstructions were compared quantitatively by taking a cross section through the center of each reconstruction and calculating the earth mover’s distance from each cross section to the original simulated object. The steel container was omitted from the comparison object as it was not captured in any of the reconstructions (including those generated from the mathematically ideal sinograms). Some sinograms produced errors and could not be reconstructed. The simulated objects that could not be reconstructed had large outer radii (all but one were greater than 14 cm) and negative axial array offsets, causing the array to be completely obscured and placing parts of the object out of the field of view of the neutron detector array.

Reconstructed images using the neural network predicted array position showed significant improvement over the reconstructions generated assuming that the array was at the nominal position. The median distance from the simulated reconstruction to the original object cross section was 5.9×10^{-4} . This was greater than the distance ray-traced reconstruction to the original object cross section (1.32×10^{-4}), but showed significant improvement when compared to the uncorrected reconstructions (2.6×10^{-3}). The distributions of the distances (see Figure 6.7) shows that the reconstructions without alignment corrections applied deviate significantly from the idealized reconstruction.

Some of the error in the corrected images appears to be caused by high-frequency oscillations in the cross section, as exemplified in Figure 6.8. Jelinek and Overton [21] identify as a possible cause of artifacts of this type miscalibration of one or more detector

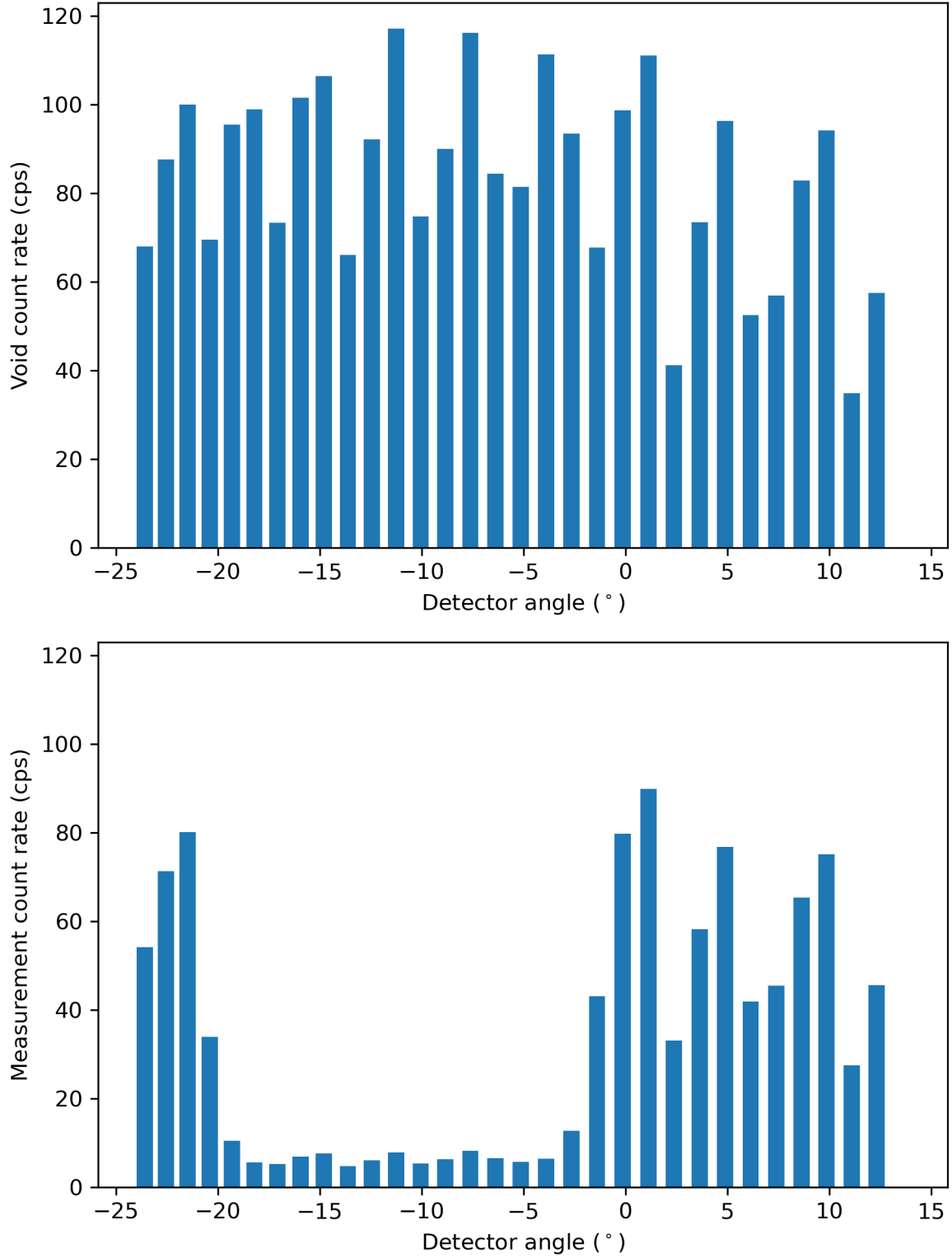


Figure 6.4: Count rate as a function of source-detector angle for a simulated void measurement (top) and with a 82 kg cylinder of depleted uranium present. For this simulation the array was shifted 12 cm in the $-X$ direction and 13 cm in the $+Y$ direction.

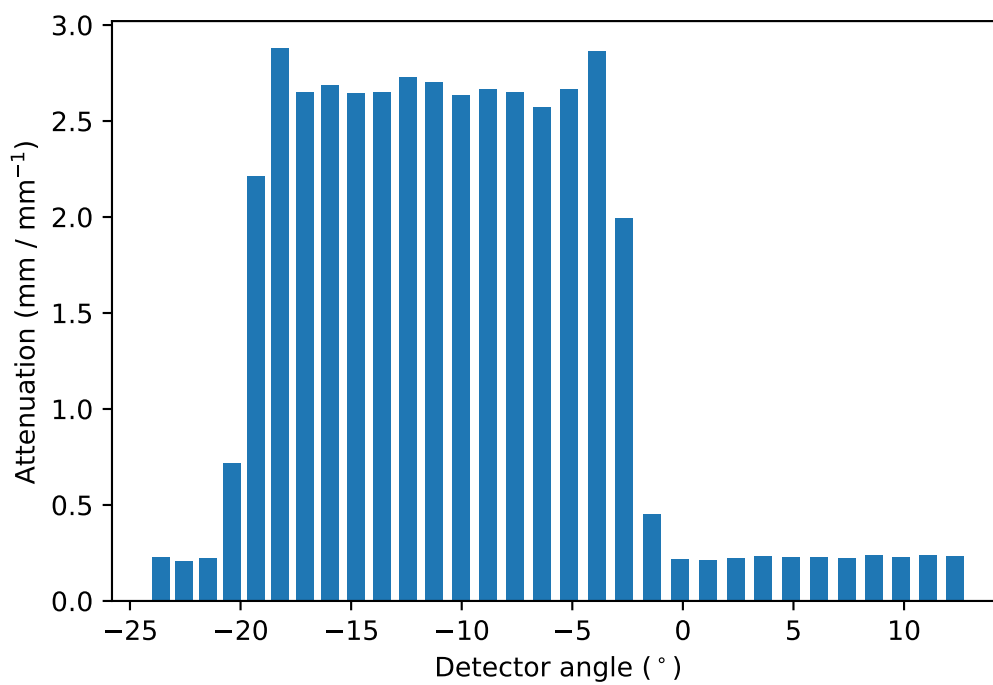


Figure 6.5: Sinogram for the object described in Figure 6.4.

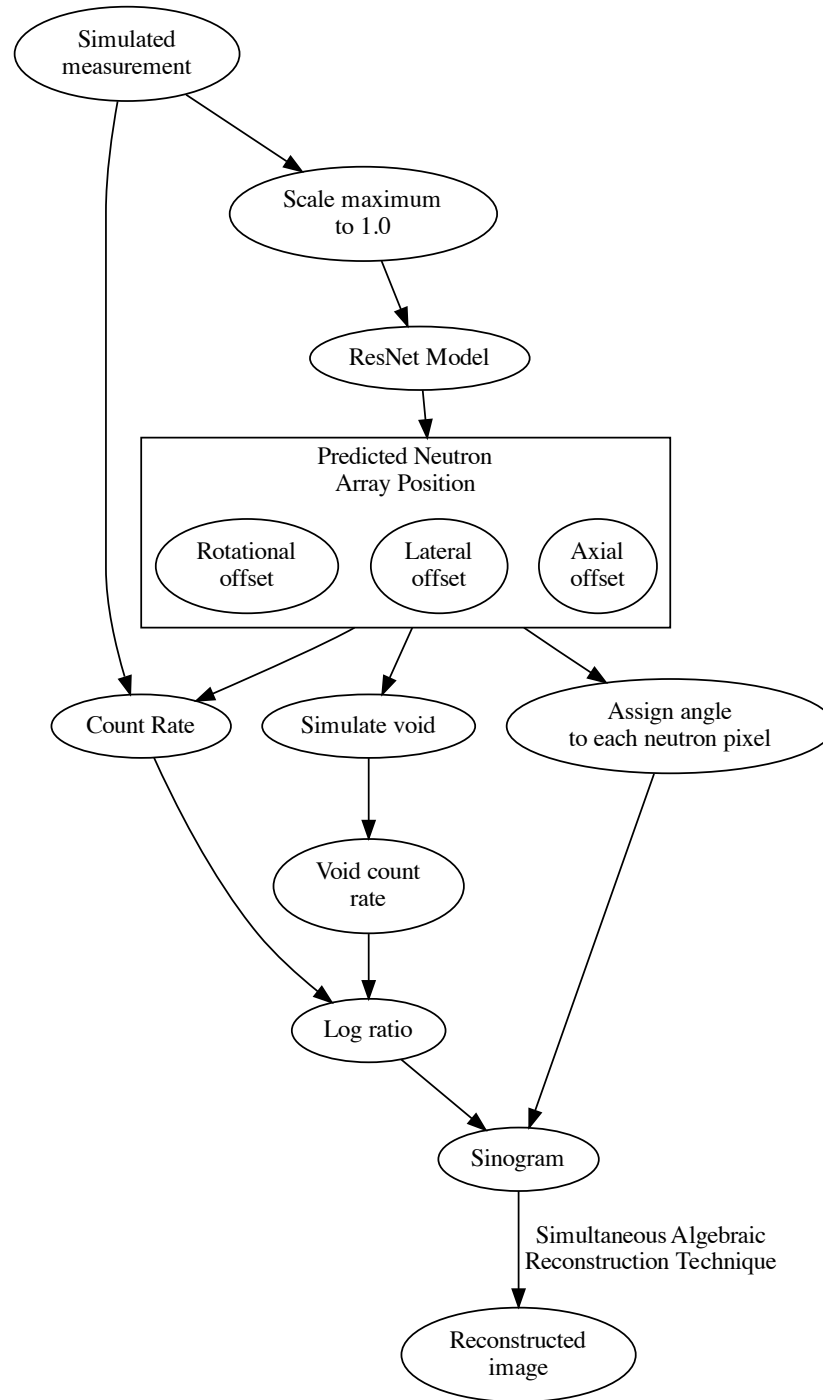


Figure 6.6: Flow chart to reconstruct image from simulated measurement.

pixels. Since the pixels in this case were identically “calibrated” (a result of the simulation), a more likely cause of this error is the SART algorithm attempting to reconstruct a high-resolution image from low angular resolution data. Statistical differences between two detector elements observing similar paths through the object appear as discontinuities in the sinogram. The SART algorithm attempts to numerically approximate a solution to the two-dimensional Fourier transform of the image using a one-dimensional transform of the projection through the image in accordance with the Fourier slice theorem [44]. Because of the discontinuities between adjacent pixels, high frequency components are introduced which appear as the rapid oscillations shown in Figure 6.8.

The factor with the most significant impact on the reconstruction error in the corrected images was the outer radius of the imaged object. This error was present in reconstructions of both the MCNP simulations and the idealized ray traces. Since the outer radius was uncorrelated with the neural network error and the outer radius was also the most correlated with the error in the reconstructed ray-traced images, this error was attributed to the reconstruction process. As Figure 6.9 illustrates, the distribution of errors grows with the object radius. The minimum error, however, stays low until around $r = 20$ cm. This corresponds to the point at which the object outer radius is larger than the support region for the reconstruction. The support region is the area seen by the detector array in every projection. In traditional x-ray CT this is a function of the detector size and position. In associated particle imaging it depends on the intersection of the angles seen by the alpha detectors and the neutron detectors. If the neutron array moves (as in these simulations) while the alpha detectors remain fixed, the overlap between the regions that they see is reduced. When this effect is extended over multiple projections, the effect is a support region that covers a smaller area, as illustrated in Figure 6.10. Objects with scattering material outside of the support area attenuate neutrons in a way that the reconstruction algorithm cannot account for, introducing error in the eventual reconstruction.

The other factor that was most correlated with cross-sectional error was the absolute error in the y-offset prediction (lateral error). The test simulation with the highest lateral error (1.8 cm in the $+y$ direction) was of a 2 cm thick ring of TNT around a 34.6 cm diameter core of steel. The reconstructed cross section under-predicted the object diameter by 18%

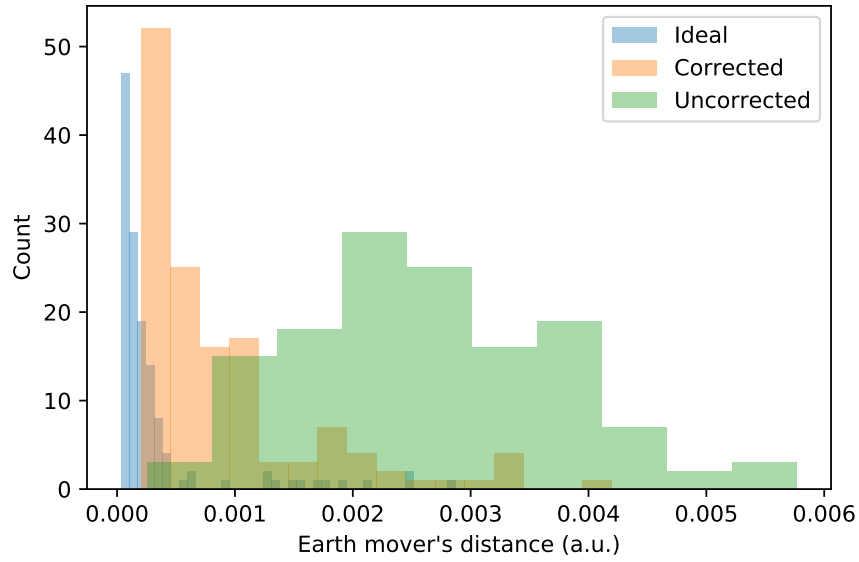


Figure 6.7: Distribution of cross sectional error for the idealized reconstruction as well as the corrected and uncorrected MCNP simulations.

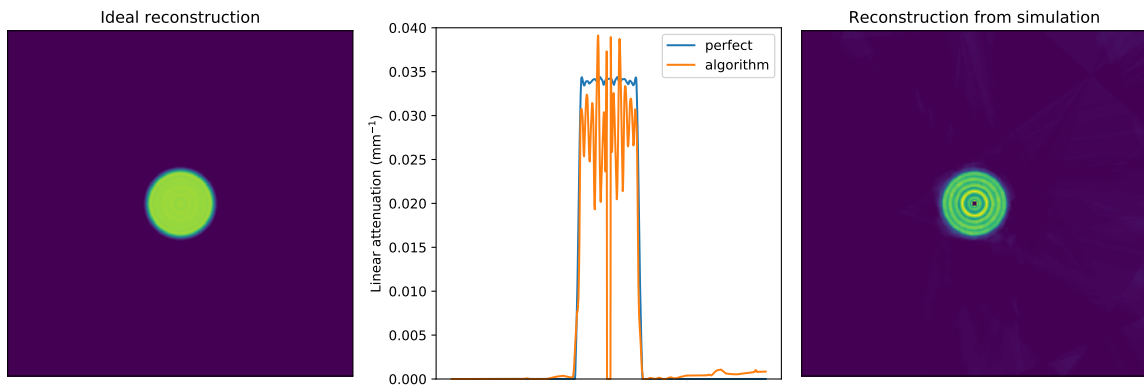


Figure 6.8: Reconstruction based on ray trace (left) and MCNP simulation (right). The two cross sections (center) show that the MCNP reconstruction has significant oscillations not present in the ray trace-based reconstruction. The overall radius is a close match in both reconstructions.

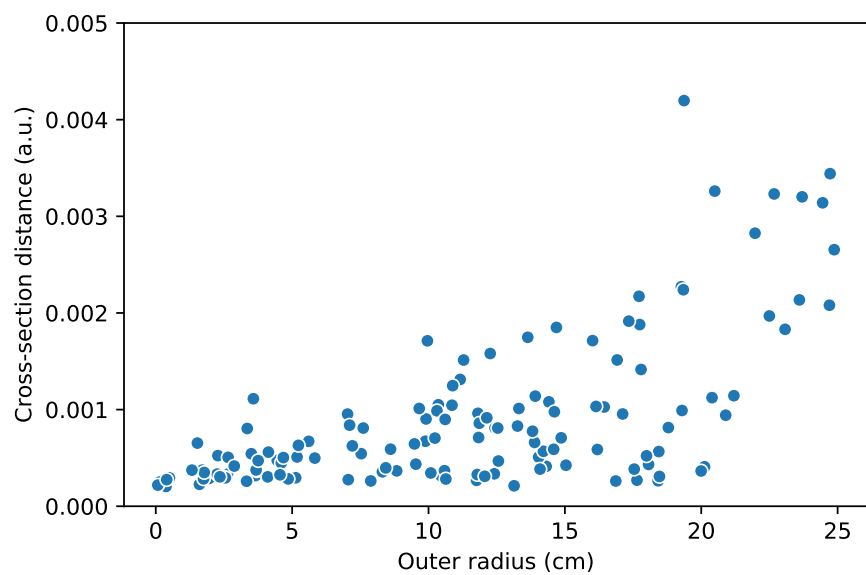


Figure 6.9: Cross sectional error in reconstructed images vs. radius of imaged object.

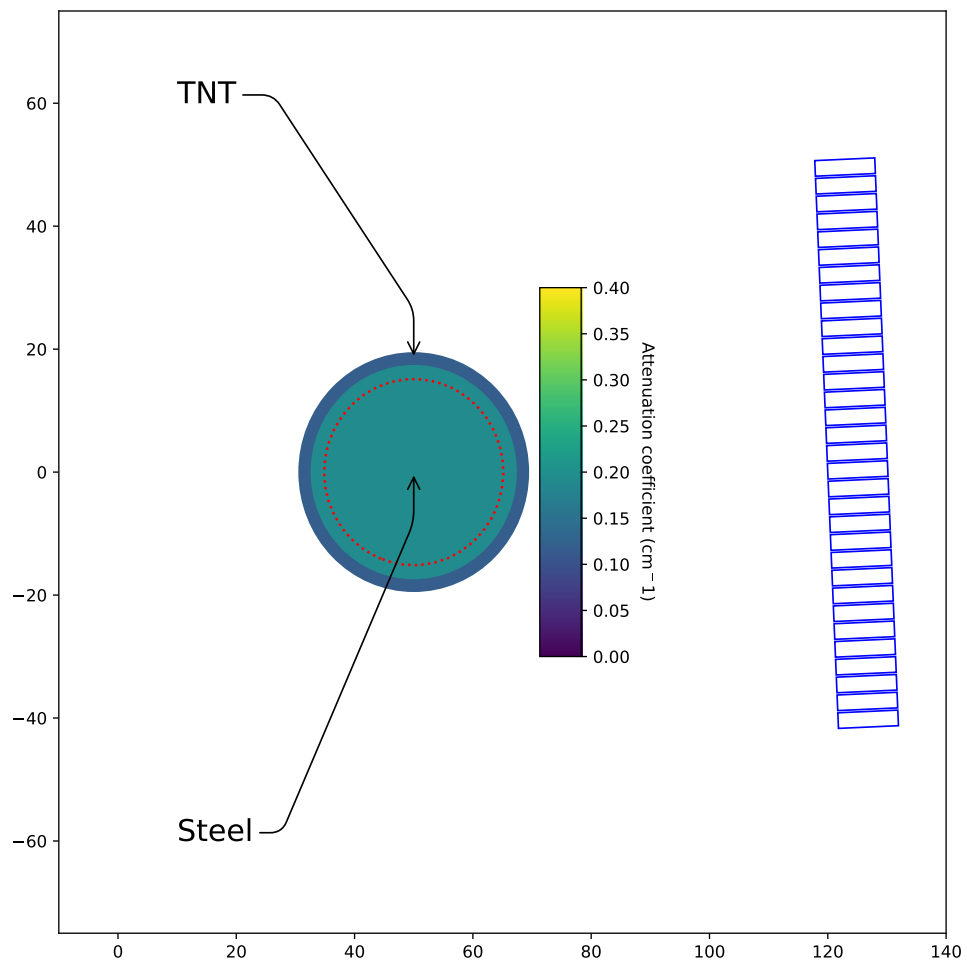


Figure 6.10: Support area for an example measurement of a steel cylinder surrounded by a jacket of TNT. The object extends beyond the support area (shown as a red dashed line), resulting in insufficient information to perform tomographic reconstruction.

and showed significant ring artifacts. The mean attenuation in the reconstruction was more consistent with depleted uranium (0.28 cm^{-1}) than steel (0.22 cm^{-1}). It is worth noting that even in the ray-traced reconstruction the outer ring of explosive is not visible, so the main contributor to the error score is the high attenuation.

Chapter 7

Conclusions and future work

This work has presented a neural-network based method that can predict the position of a 2-d array of neutron detectors with sufficient accuracy for tomographic reconstruction. Unlike other methods for aligning radiographic detectors, it can make this prediction using only a single projection through the object of interest. Previous methods required either multiple projections from different angles, the use of calibration phantoms, or both. The method presented here instead uses the time-dependent measurement method of associated particle imaging.

The method presented here uses neural networks to infer the dependencies between the simulated time correlation data and the array position. The network was able to accurately predict the simulated array position to a standard deviation of 0.67 cm, within the half-pixel-width identified by Wenig and Kasperl [22] as sufficient for image reconstruction. This corresponded in an improved mean cross-sectional earth mover’s distance in reconstructed images from 2.6×10^{-3} to 5.9×10^{-4} , accounting for an expected five-fold improvement when compared to the reconstructions based on idealized ray traces. The distribution of errors remaining qualitatively matched the distribution of errors in the ray trace reconstructions, indicating that much of the remaining error was the result of properties inherent to the NMIS system itself.

This model has potential for improvement by using more physics-based modeling of the detector response. The current simulation necessarily averages over all pulse shape distributions in the alpha and neutron detectors. Higher-fidelity simulations might recover

more angular information useful to the network. While the simulations used here were representative of a likely portable detector system design, this has not been finalized. A sensitivity study examining different neutron array configurations could better inform decisions on final system development. Two different array configurations (one straight array, one pseudo-curved array) were simulated. The same network architecture, when trained on simulated data, converged to similar accuracy in each case. The objects modeled in simulation (simple annuli) were representative, but it is possible that networks trained on application-specific objects (e.g. simulated improvised explosive devices for a bomb disposal detector system) could perform better in their specific regimes. Certainly applications that involve imaging non rotationally-symmetric objects merit additional simulation work to ensure that the network produces accurate output for multiple projections. Neural networks depend on sufficient training data whose underlying distribution matches that of the data used to generate predictions. Care should be taken to ensure that the simulations used to generate training data span the full range of potential applications of the actual detector, as well as the full range of operating parameters that might be used. Specifically, this method is sensitive to detector response for each of the alpha and neutron pixels. Training data that does not incorporate this variability may cause unexpected error in the network output. While the decision was made to neglect the energy dependence of detector efficiency, it may be possible to more realistically incorporate this effect at minimal computational cost. Verbinski et al. [45] present a method for calculating the material efficiency of an organic scintillator as a function of energy. If similar efficiency experiments were performed for the detectors used in the NMIS system, it should be possible to incorporate this detector response without sacrificing MCNP's ability to perform calculations in parallel. If so, it should be possible to better match the simulation framework to laboratory measurements. These improved detector response simulation methods could be incorporated into both the simulations used to train the neural network and into the MCNP void simulations required to perform tomographic reconstruction when the measured object cannot be moved.

While the method presented here proved capable of predicting the position of the simulated neutron detector array, attempts at a simple scaling of a void measurement were not accurate enough for tomographic image reconstruction. In order to reconstruct images

when a void measurement is not feasible, a true-to-life simulation framework (better than the proof-of-concept simulation presented in this work) should be developed. Lessons learned during the development of these simulations indicate that the simulation environment must be tuned for the specific generator, generator power, and detector settings as changes to these values affect the detector system in a nonlinear fashion. Current void simulations require tens of minutes of processing time on reasonably high-performing computers. Since a void measurement primarily depends on measurement time and array position, it may be possible to use generative neural networks to reduce computation costs at measurement time.

The successful image reconstruction indicates that the current neural network is reasonably accurate, but it may be that some unforeseen uses of the detector require more accuracy (perhaps in a detector system that provides more image resolution). In this case it is possible that more complicated networks or ensembles of networks merit investigation; they are omitted here in the interest of minimizing complexity. This work only examined neutron array movement in three dimensions, leaving three remaining dimensions (vertical/Z-axis translation and rotation in the Y-Z or X-Z planes) unexamined. This work also did not focus on improved tomographic reconstruction methods. In Chapter 6 out-of-field artifacts were identified as a significant source of error. Recent work by Yang et al. [46] may provide a mechanism to correct for these deficiencies and should be investigated. In the meantime, the NMIS system is best suited for measurement scenarios where the object of interest subtends a smaller angle (relative to the neutron generator) than the neutron array itself.

Bibliography

- [1] Jason M Crye. *Enrichment Determination of Uranium Metal in Shielded Configurations Without Calibration Standards*. PhD thesis, University of Tennessee, 2013. URL http://trace.tennessee.edu/utk_graddiss/1711/. 1
- [2] J T Mihalcz, V K Pare, and T E Valentine. Nuclear weapons identification system. In *1996 IEEE Nuclear Science Symposium. Conference Record*, volume 1, pages 44–45 vol.1, nov 1996. doi: 10.1109/NSSMIC.1996.590888. 5, 17
- [3] Seth M. Mcconchie and Matthew A. Blackston. Personal communication, 2019. viii, 7
- [4] David W Anderson. *Absorption of ionizing radiation*. University Park Press, Baltimore, MD, 1984. ISBN 083911821. 8
- [5] David A Brown, MB Chadwick, R Capote, AC Kahler, A Trkov, MW Herman, AA Sonzogni, Y Danon, AD Carlson, M Dunn, et al. Endf/b-viii. 0: The 8th major release of the nuclear reaction data library with cielo-project cross sections, new standards and thermal scattering data. *Nuclear Data Sheets*, 148:1–142, 2018. viii, 10
- [6] Avinash C. Kak and Malcolm Slaney. *Principles of Computerized Tomographic Imaging*. Number 1. SIAM, Philadelphia, PA (United States). ISBN 978-0-898714-94-4. 11, 17
- [7] LeCun Yann and Bengio Yoshua. Convolutional Networks for Images, Speech, and Time-Series. 4(April 2016):2571–2575, 1995. ISSN 1098-7576. doi: 10.1109/IJCNN.2004.1381049. URL https://www.researchgate.net/profile/Yann_Lecun/publication/2453996_Convolutional_Networks_for_Images_Speech_and_Time-links/0deec519dfa2325502000000.pdf?%0Ahttps://www.researchgate.net/profile/Yann_Lecun/publication/2453996_Convolutional_Networks_fo. 12
- [8] Matthew D. Zeiler, Graham W. Taylor, and Rob Fergus. Adaptive deconvolutional networks for mid and high level feature learning. *Proceedings of the IEEE International Conference on Computer Vision*, pages 2018–2025, 2011. ISSN 1550-5499. doi: 10.1109/ICCV.2011.6126474. viii, x, 13, 58, 60

- [9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT press, Cambridge, MA, 2016. ISBN 9780262035613. URL <http://www.deeplearningbook.org>. 14, 59, 60
- [10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, jun 2016. ix, 14, 15, 116
- [11] Hongzhou Lin and Stefanie Jegelka. ResNet with one-neuron hidden layers is a Universal Approximator. In S Bengio, H Wallach, H Larochelle, K Grauman, N Cesa-Bianchi, and R Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6169–6178. Curran Associates, Inc., 2018. URL <http://papers.nips.cc/paper/7855-resnet-with-one-neuron-hidden-layers-is-a-universal-approximator.pdf>. 14
- [12] J. T. Mihalczko, J. A. Mullens, J. K. Mattingly, and T. E. Valentine. Physical description of nuclear materials identification system (NMIS) signatures. *Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 450(2):531–555, 2000. ISSN 01689002. doi: 10.1016/S0168-9002(00)00304-1. 17
- [13] J.A. Mullens. Addition of Tomographic Capabilities to NMIS (Y/LB-16,160). Technical report, Oak Ridge Y-12 Plant (Y-12), Oak Ridge, TN, mar 2003. URL <http://www.osti.gov/servlets/purl/809868-JHACXg/native/>. 17
- [14] D. L. Chichester, J. D. Simpson, and M. Lemchak. Advanced compact accelerator neutron generator technology for active neutron interrogation field work. *Journal of Radioanalytical and Nuclear Chemistry*, 271(3):629–637, mar 2007. ISSN 0236-5731. doi: 10.1007/s10967-007-0318-7. URL <http://link.springer.com/10.1007/s10967-007-0318-7>. 17
- [15] Brandon R Grogan, Seth M McConchie, John T Mihalczko, and James Allen Mullens. Alpha Detector Pixelization Effects

- on NMIS Imaging. URL <https://www.osti.gov/biblio/941040-alpha-detector-pixelization-effects-nmis-imaging>. 18, 47
- [16] B. R. Grogan, J. T. Mihalczo, and J. A. Mullens. MCNP-PoliMi Simulation of Neutron Radiography Measurements for Mass Determination for a Trough of UO₃. 2008. 18
- [17] AL Swift. *Materials and Configuration from NMIS Type Neutron Imaging and Gamma Spectroscopy*. PhD thesis, University of Tennessee, 2012. URL https://trace.tennessee.edu/utk_gradthes/1210/. 18
- [18] Kirsten Elise Pena. *Template-Based Imaging Analysis of Arbitrary Nuclear Material Configurations Using Time and Directionally-Tagged Fast Neutrons*. PhD thesis, University of Tennessee, 2015. URL https://trace.tennessee.edu/utk_graddiss/3456/. 18, 20
- [19] Benjamin Troxell. Fieldable Neutron Imaging System: Simulation and Experimentation into the Viability of Associated Particle Imaging in Austere Field Environments. Master’s thesis, University of Tennessee, 2020. 19, 42, 44, 55, 56
- [20] F. Edward Boas and Dominik Fleischmann. CT artifacts: causes and reduction techniques. *Imaging in Medicine*, 4(2):229–240, apr 2012. ISSN 1755-5191. doi: 10.2217/iim.12.13. URL <http://www.futuremedicine.com/doi/abs/10.2217/iim.12.13>. ix, 20, 36
- [21] J. Jelinek and T. R. Overton. Simulation of CT reconstruction artifacts associated with multiple-rotation fan-beam data collection. *Computerized Radiology*, 10(1):23–36, 1986. ISSN 07304862. doi: 10.1016/0730-4862(86)90016-8. 20, 69
- [22] Peter Wenig and Stefan Kasperl. Examination of the measurement uncertainty on dimensional measurements by X-ray computed tomography. *European Conference on Non-Destructive Testing*, pages 1–10, 2006. 20, 62, 78

- [23] Massimiliano Ferrucci, Richard K. Leach, Claudiu Giusca, Simone Carmignato, and Wim Dewulf. Towards geometrical calibration of x-ray computed tomography systems - A review, 2015. ISSN 13616501. [20](#)
- [24] G. T. Gullberg, B. M.W. Tsui, C. R. Crawford, and E. R. Edgerton. Estimation of geometrical parameters for fan beam tomography. *Physics in Medicine and Biology*, 32(12):1581–1594, 1987. ISSN 00319155. doi: 10.1088/0031-9155/32/12/005. [21](#)
- [25] Yi Sun, Ying Hou, Fengyong Zhao, and Jiasheng Hu. A calibration method for misaligned scanner geometry in cone-beam computed tomography. *NDT and E International*, 39(6):499–513, 2006. ISSN 09638695. doi: 10.1016/j.ndteint.2006.03.002. [21](#)
- [26] Yi Sun, Ying Hou, and Jiasheng Hu. Reduction of artifacts induced by misaligned geometry in cone-beam CT. *IEEE Transactions on Biomedical Engineering*, 54(8):1461–1471, 2007. ISSN 00189294. doi: 10.1109/TBME.2007.891166. [21](#)
- [27] Y. Kyriakou, R. M. Lapp, L. Hillebrand, D. Ertel, and W. A. Kalender. Simultaneous misalignment correction for approximate circular cone-beam computed tomography. *Physics in Medicine and Biology*, 53(22):6267–6289, 2008. ISSN 00319155. doi: 10.1088/0031-9155/53/22/001. [21](#)
- [28] Liang Zheng, Lujie Chen, Shuang Luo, and Shouhua Luo. Self-geometric calibration of circular cone beam CT based on epipolar geometry consistency. In Hilde Bosmans, Guang-Hong Chen, and Taly Gilat Schmidt, editors, *Medical Imaging 2019: Physics of Medical Imaging*, number May, page 90. SPIE, mar 2019. ISBN 9781510625433. doi: 10.1117/12.2511127. URL <https://www.spiedigitallibrary.org/conference-proceedings-of-spie/10948/2511127/Self-geometric-calibration-of-circular-cone-beam-CT-based-on/10.1117/12.2511127.full>. [21](#)
- [29] Tdudziak. CFD_Diagram1, 2019. URL https://commons.wikimedia.org/wiki/File:CFD{__}Diagram1.jpg. ix, [26](#)

- [30] S Basu and Y Bresler. Feasibility of tomography with unknown view angles. *IEEE Transactions on Image Processing*, 9(6):1107–1122, 2000. ISSN 1057-7149. [34](#)
- [31] Wim van Aarle, Willem Jan Palenstijn, Jan De Beenhouwer, Thomas Altantzis, Sara Bals, K Joost Batenburg, and Jan Sijbers. The ASTRA Toolbox: A platform for advanced algorithm development in electron tomography. *Ultramicroscopy*, 157:35–47, 2015. ISSN 0304-3991. doi: <https://doi.org/10.1016/j.ultramic.2015.05.002>. URL <http://www.sciencedirect.com/science/article/pii/S0304399115001060>. [34](#)
- [32] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The Earth Mover’s Distance as a Metric for Image Retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000. doi: [10.1023/A:1026543900054](https://doi.org/10.1023/A:1026543900054). [39](#)
- [33] V. Valkovic, D. Sudac, J. Obhodas, C. Eleon, B. Perot, C. Carasco, G. Sannié, K. Boudergui, V. Kondrasovs, G. Corre, S. Normand, R. Woo, and J. M. Bourbotte. The use of alpha particle tagged neutrons for the inspection of objects on the sea floor for the presence of explosives. *Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 703:133–137, 2013. ISSN 01689002. doi: [10.1016/j.nima.2012.11.096](https://doi.org/10.1016/j.nima.2012.11.096). [42](#)
- [34] Ronald J McConn, Christopher J Gesh, Richard T Pagh, Robert A Rucker, and Robert Williams III. Compendium of Material Composition Data for Radiation Transport Modeling. Technical report, Pacific Northwest National Laboratory (PNNL), Richland, WA (United States), mar 2011. URL <http://www.osti.gov/servlets/purl/1023125-QtpCud/>. [44](#)
- [35] Boxin Tang. Orthogonal array-based latin hypercubes. *Journal of the American Statistical Association*, 88(424):1392–1397, 1993. ISSN 1537274X. doi: [10.1080/01621459.1993.10476423](https://doi.org/10.1080/01621459.1993.10476423). [44](#)
- [36] M D Mckay, R J Beckman, and W J Conover. A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output From a

- Computer Code. *Technometrics*, 42(1):55–61, 2000. doi: 10.1080/00401706.2000.10485979. URL <https://amstat.tandfonline.com/doi/abs/10.1080/00401706.2000.10485979>. 44
- [37] Christopher John Werner, Jeffrey S Bull, C J Solomon, Forrest B Brown, Gregg Walter McKinney, Michael Evan Rising, David A Dixon, Roger Lee Martz, Henry G Hughes, Lawrence James Cox, Anthony J Zukaitis, J C Armstrong, Robert Arthur Forster, and Laura Casswell. MCNP Version 6.2 Release Notes. feb 2018. doi: 10.2172/1419730. URL <http://www.osti.gov/servlets/purl/1419730/>. 45
- [38] B R Grogan. *The Development of a Parameterized Scatter Removal Algorithm for NMIS Imaging*. PhD thesis, University of Tennessee, 2010. URL https://trace.tennessee.edu/utk_graddiss/692/. 47
- [39] Leslie N. Smith. A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay. pages 1–21, 2018. URL <http://arxiv.org/abs/1803.09820>. 59
- [40] Jonathan Baxter. Learning internal representations. *Proceedings of the 8th Annual Conference on Computational Learning Theory, COLT 1995*, 1995-Janua:311–320, 1995. doi: 10.7551/mitpress/2906.003.0006. 59
- [41] J Deng, W Dong, R Socher, L.-J. Li, K Li, and L Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. 60
- [42] Tiago S Nazaré, Gabriel B Paranhos da Costa, Welinton A Contato, and Moacir Ponti. Deep convolutional neural networks and noisy images. In *Iberoamerican Congress on Pattern Recognition*, pages 416–424. Springer, 2017. 65
- [43] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017. doi: 10.1109/TIP.2017.2662206. 65
- [44] Ronald N Bracewell. Strip integration in radio astronomy. *Australian Journal of Physics*, 9(2):198–217, 1956. 73

- [45] V V Verbinski, W R Burrus, T A Love, W Zobel, N W Hill, and R Textor. Calibration of an organic scintillator for neutron spectrometry. *Nuclear Instruments and Methods*, 65(1):8–25, 1968. URL http://shop-pdp.net/efhtml/nim{_}065{_}1968{_}8-25{_}verbinski.pdf. 79
- [46] Min Yang, Xu Han, Xiaojun Wu, Gang Zhao, Dongtao Wei, Tian Lang, and Shunli Zhang. Artifacts correction method for fan-beam CT with projections asymmetrically truncated on both sides. *NDT and E International*, 87(January):24–30, 2017. ISSN 09638695. doi: 10.1016/j.ndteint.2017.01.002. URL <http://dx.doi.org/10.1016/j.ndteint.2017.01.002>. 80
- [47] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958. ISSN 0033295X. doi: 10.1037/h0042519. 116
- [48] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computatiojn*, 9(8), 1997. 116
- [49] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998. ISSN 1558-2256. doi: 10.1109/5.726791. 116
- [50] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359 – 366, 1989. ISSN 0893-6080. doi: [https://doi.org/10.1016/0893-6080\(89\)90020-8](https://doi.org/10.1016/0893-6080(89)90020-8). URL <http://www.sciencedirect.com/science/article/pii/0893608089900208>. 116
- [51] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, may 2017. ISSN 0001-0782. doi: 10.1145/3065386. URL <https://dl.acm.org/doi/10.1145/3065386>. 116
- [52] Matthew D. Zeiler. Adadelta: An adaptive learning rate method, 2012. URL <https://arxiv.org/abs/1212.5701>. 116

Appendices

A Example of rendered MCNP input deck

message: name=0a5a5636-d474-4c11-95e3-4078cdd038d0.dtp01. tasks 30

Title: dx: -11.01 dy: -11.58 dt: -4.74 or: 18.09 om: 1 ir: 15.52 im: 6

```
c ++++++
c +
c +          Cells          +
c +
c ++++++
c Neutron pixels
c -----
1101    2      -1.023  -101    imp:n=4 imp:p=4
1102    2      -1.023  -102    imp:n=4 imp:p=4
1103    2      -1.023  -103    imp:n=4 imp:p=4
1104    2      -1.023  -104    imp:n=4 imp:p=4
1105    2      -1.023  -105    imp:n=4 imp:p=4
1106    2      -1.023  -106    imp:n=4 imp:p=4
1107    2      -1.023  -107    imp:n=4 imp:p=4
1108    2      -1.023  -108    imp:n=4 imp:p=4
1109    2      -1.023  -109    imp:n=4 imp:p=4
1110    2      -1.023  -110    imp:n=4 imp:p=4
1111    2      -1.023  -111    imp:n=4 imp:p=4
1112    2      -1.023  -112    imp:n=4 imp:p=4
1113    2      -1.023  -113    imp:n=4 imp:p=4
1114    2      -1.023  -114    imp:n=4 imp:p=4
1115    2      -1.023  -115    imp:n=4 imp:p=4
1116    2      -1.023  -116    imp:n=4 imp:p=4
1117    2      -1.023  -117    imp:n=4 imp:p=4
1118    2      -1.023  -118    imp:n=4 imp:p=4
1119    2      -1.023  -119    imp:n=4 imp:p=4
1120    2      -1.023  -120    imp:n=4 imp:p=4
1121    2      -1.023  -121    imp:n=4 imp:p=4
1122    2      -1.023  -122    imp:n=4 imp:p=4
1123    2      -1.023  -123    imp:n=4 imp:p=4
1124    2      -1.023  -124    imp:n=4 imp:p=4
```

```

1125      2      -1.023  -125      imp:n=4 imp:p=4
1126      2      -1.023  -126      imp:n=4 imp:p=4
1127      2      -1.023  -127      imp:n=4 imp:p=4
1128      2      -1.023  -128      imp:n=4 imp:p=4
1129      2      -1.023  -129      imp:n=4 imp:p=4
1130      2      -1.023  -130      imp:n=4 imp:p=4
1131      2      -1.023  -131      imp:n=4 imp:p=4
1132      2      -1.023  -132      imp:n=4 imp:p=4
c -----
c Steel Box
c -----
1201      3      -7.8      -201 202 imp:n=4 imp:p=4
c -----
c Imaged Annulus
c -----
1301      1      -0.001      -301 302 imp:n=4 imp:p=4
1302      6      -1.800      -302      imp:n=4 imp:p=4
c -----
c The world and void
c -----
1901      1      -0.001225  -900  #1101 #1102 #1103
                                   #1104 #1105 #1106
                                   #1107 #1108 #1109
                                   #1110 #1111 #1112
                                   #1113 #1114 #1115
                                   #1116 #1117 #1118
                                   #1119 #1120 #1121
                                   #1122 #1123 #1124
                                   #1125 #1126 #1127
                                   #1128 #1129 #1130
                                   #1131 #1132 #1201
                                   #1301
                                   #1302 imp:n=1 imp:p=1
9999      0                                   900      imp:n=0 imp:p=0

c ++++++
```

c	+							+
c	+	Surfaces						+
c	+							+
c	+++++							
c	Neutron pixels							
c	-----							
c	tr		xmin	xmax	ymin	ymax	zmin	zmax
c	---		-----	-----	-----	-----	-----	-----
101	100	rpp	0.00	10.16	-46.22	-43.68	-1.27	1.27
102	100	rpp	0.00	10.16	-43.32	-40.78	-1.27	1.27
103	100	rpp	0.00	10.16	-40.42	-37.88	-1.27	1.27
104	100	rpp	0.00	10.16	-37.52	-34.98	-1.27	1.27
105	100	rpp	0.00	10.16	-34.62	-32.08	-1.27	1.27
106	100	rpp	0.00	10.16	-31.72	-29.18	-1.27	1.27
107	100	rpp	0.00	10.16	-28.82	-26.28	-1.27	1.27
108	100	rpp	0.00	10.16	-25.92	-23.38	-1.27	1.27
109	100	rpp	0.00	10.16	-23.02	-20.48	-1.27	1.27
110	100	rpp	0.00	10.16	-20.12	-17.58	-1.27	1.27
111	100	rpp	0.00	10.16	-17.22	-14.68	-1.27	1.27
112	100	rpp	0.00	10.16	-14.32	-11.78	-1.27	1.27
113	100	rpp	0.00	10.16	-11.42	-8.88	-1.27	1.27
114	100	rpp	0.00	10.16	-8.52	-5.98	-1.27	1.27
115	100	rpp	0.00	10.16	-5.62	-3.08	-1.27	1.27
116	100	rpp	0.00	10.16	-2.72	-0.18	-1.27	1.27
117	100	rpp	0.00	10.16	0.18	2.72	-1.27	1.27
118	100	rpp	0.00	10.16	3.08	5.62	-1.27	1.27
119	100	rpp	0.00	10.16	5.98	8.52	-1.27	1.27
120	100	rpp	0.00	10.16	8.88	11.42	-1.27	1.27
121	100	rpp	0.00	10.16	11.78	14.32	-1.27	1.27
122	100	rpp	0.00	10.16	14.68	17.22	-1.27	1.27
123	100	rpp	0.00	10.16	17.58	20.12	-1.27	1.27
124	100	rpp	0.00	10.16	20.48	23.02	-1.27	1.27
125	100	rpp	0.00	10.16	23.38	25.92	-1.27	1.27
126	100	rpp	0.00	10.16	26.28	28.82	-1.27	1.27
127	100	rpp	0.00	10.16	29.18	31.72	-1.27	1.27
128	100	rpp	0.00	10.16	32.08	34.62	-1.27	1.27

```

129 100 rpp    0.00  10.16   34.98  37.52  -1.27   1.27
130 100 rpp    0.00  10.16   37.88  40.42  -1.27   1.27
131 100 rpp    0.00  10.16   40.78  43.32  -1.27   1.27
132 100 rpp    0.00  10.16   43.68  46.22  -1.27   1.27
c -----
c Steel Box
c -----
c   tr      xmin   xmax   ymin   ymax   zmin   zmax
c   ---      -
201   rpp    1.00   93.0  -100.0  100.0 -100.0  100.0
202   rpp    1.50   92.5   -99.5   99.5  -99.5   99.5
c -----
c Imaged Annulus
c -----
c      x      y      z      vx      vy      vz      r
c      -
301 rcc  50.00    0.00 -20.00   0.00   0.00  40.00 18.09
302 rcc  50.00    0.00 -20.00   0.00   0.00  40.00 15.52
c -----
c The world
c -----
900 rpp -1 150 -101 101 -101 101

c ++++++
c +
c +      Data Cards
c +
c ++++++
c Source
c -----
read file=../templates/src_dtp01.txt
c -----
c Materials
c -----
C name: Air (dry, near sea level)
C density = 0.001225

```


C Always used (atmosphere)

m1

6012 -1.2264e-04
6013 -1.4374e-06
7014 -7.5280e-01
7015 -2.9460e-03
8016 -2.3130e-01
18036 -3.8552e-05
18038 -7.6721e-06
18040 -1.2789e-02

C name: Polyvinyl Toluene (PVT)

C density = 1.023

C Always used (neutron detectors)

m2

1001 -8.4980e-02
1002 -1.9533e-05
6012 -9.0440e-01
6013 -1.0600e-02

C name: Steel, Carbon

C density = 7.8

C Always used (B-25 box)

m3

6012 -4.9421e-03
6013 -5.7921e-05
26054 -5.6173e-02
26056 -9.1442e-01
26057 -2.1496e-02
26058 -2.9108e-03

C name: Aluminum

C density = 2.7

m4

13027 -1.0000e+00

C name: Explosive compound, TNT

C density = 1.6

m5

1001 -2.2206e-02

```

1002 -5.1039e-06
6012 -3.6623e-01
6013 -4.2922e-03
7014 -1.8446e-01
7015 -7.2188e-04
8016 -4.2191e-01
8017 -1.7081e-04
C name: Explosive compound, RDX
C density = 1.8
m6
1001 -2.7248e-02
1002 -6.2629e-06
6012 -1.6050e-01
6013 -1.8811e-03
7014 -3.7726e-01
7015 -1.4764e-03
8016 -4.3145e-01
8017 -1.7467e-04
C name: Tungsten
C density = 19.3
m7
74180 -1.1746e-03
74182 -2.6227e-01
74183 -1.4241e-01
74184 -3.0658e-01
74186 -2.8757e-01
C name: Uranium, Depleted, Typical
C density = 19.0
m8
92234 -5.0000e-06
92235 -2.5000e-03
92238 -9.9750e-01
c -----
c Translation
c -----
tr100* 104.094 -11.577 0.000

```

```

-4.737  85.263  90.000
94.737  -4.737  90.000
90.000  90.000  0.000
1
c -----
c Tallies
c -----
f14:n 1101 30i 1132
fm14 -1 2 (2:3)
sd14 1 30r 1
fc14 Neutron flux, integrated over volume, in each detector cell, times XS
f24:p 1101 30i 1132
fm24 -1 2 (502:504)
sd24 1 30r 1
fc24 Photon flux, integrated over volume, in each detector cell, times XS
t0 -10.24 3071i 20.48 NT
c -----
c Physics
c -----
mode n p
cut:n,p 20.48 0.5 3j
phys:n 15 j j 3j j j j 2j j j
phys:p 15 1 j 0 j j
c -----
c Problem control and output
c -----
nps 2183466
prdmp -10 -10 1 1 0

```

B Conversion from MCNP output to image

B.1 Library code

```
"""Main module."""

import numpy as np
from numpy.random import binomial
from PIL import Image
from mcnpertools import Mctal, MctalTally
from pathlib import Path
import logging

def read_one_dtp(mctal):
    """Process one of the 15 dtp files from an MCNP run"""
    logger = logging.getLogger(__name__)

    try:
        neutrons = mctal.GetTally(14)
        photons = mctal.GetTally(24)
    except Exception as me:
        logger.warn(f'could_not_access_tally_for_{mctal}')
        raise me

    ntimes = len(neutrons.GetTBins())
    nfacets = len(neutrons.GetFBins())
    tfcv = [MctalTally.TFC for v in range(6)]

    nvals = [[neutrons.GetValue(fi, *tfcv, ti) for fi in range(nfacets)] \
              for ti in range(ntimes)]
    pvals = [[photons.GetValue(fi, *tfcv, ti) for fi in range(nfacets)] \
              for ti in range(ntimes)]
    vals = np.array([nvals, pvals])

    # Quick sanity check
```

```

assert vals.shape == (2, 3073, 32)

return vals


def mctal_to_prob(guid, directory):
    """For one simulation set (15 files), create one probability matrix"""
    logger = logging.getLogger(__name__)
    logger.debug(f'processing_{guid}')

    fns = sorted([directory / f'{guid}.dtp{i:02d}.m' for i in range(1, 16)])

    mctals = [Mctal(str(fn)) for fn in fns]

    probs = [read_one_dtp(m) for m in mctals]
    vals = np.array(probs)

    # Quick sanity check
    assert vals.shape == (15, 2, 3073, 32)

    logger.debug(f'finished_{guid}')
    return vals


def make_measurements(probs, time=600):
    """Given a (15, 2, 3073, 32) probability matrix, generate a measurement

    Parameters
    -----

    probs
        numpy array containing binomial probability of detection for an event
        in each alpha detector. Shape is (dtp, particle, times, img), giving
        (15, 2, 3073, 32).

    time : int
        The measurement time (seconds). Defaults to 600s (10 minutes)

```

Returns

numpy.ndarray

Counts in each img detector corresponding to each dtp. Shape is (15, 3073, 32).

"""

```
# arates_file = Path(__file__).parents[0] / 'kernels' / 'alpha_geometric_efficiency.npy'
```

```
# arates = np.load(arates_file)
```

```
arates = np.array([
```

```
    3.64E+03,
```

```
    3.85E+03,
```

```
    4.13E+03,
```

```
    4.33E+03,
```

```
    4.66E+03,
```

```
    4.82E+03,
```

```
    4.91E+03,
```

```
    4.99E+03,
```

```
    5.04E+03,
```

```
    4.86E+03,
```

```
    4.56E+03,
```

```
    4.35E+03,
```

```
    3.87E+03,
```

```
    4.34E+03,
```

```
    4.23E+03
```

```
])
```

```
meas = []
```

```
for i, arate in enumerate(arates):
```

```
    trials = int(time * arate)
```

```
    try:
```

```
        ncounts = binomial(n=trials, p=probs[i, 0, :, :])
```

```
        pcounts = binomial(n=trials, p=probs[i, 1, :, :])
```

```
    except TypeError as te:
```

```
        print(type(probs))
```

```

        raise te

    counts = ncounts + pcounts
    meas.append(counts)

meas = np.array(meas)

# Quick sanity check
assert meas.shape == (15, 3073, 32)

return meas


def blur_measurements(meas):
    """Blurs img measurements to account for alpha arrival times"""
    # Load the blurring kernels
    blur_file = Path(__file__).parents[0] / 'kernels' / '
        final_blurring_kernels.npy'
    blurks = np.load(blur_file)

    vals = []

    for blurk, m in zip(blurks, meas):
        out = blurk @ m
        assert out.shape == (2049, 32)
        vals.append(out)

    vals = np.hstack(vals).astype(np.int)

    # Quick sanity check
    assert vals.shape == (2049, 480)

    return vals


def numpy_to_image(array, logscale=False):
    """Create PNG files from a numpy measurement array

```

Parameters

array

The measurement array. Should be of type (np.int)

logscale, optional

Boolean describing whether the images should be log scaled or not

Returns

PIL.Image

Output image that can be saved as a PNG. Image will be black and white using the 16-bit color depth, scaled to the maximum counts in the measurement.

"""

if logscale:

 array = array.astype(np.float)

 array = np.log(array)

 m = array.min()

 np.nan_to_num(array, copy=False, neginf=0.1*m)

Scale the array to [0, 1]

array = array.astype(np.float)

array /= array.max()

array *= 2**16 - 1

array = array.astype(np.uint16)

img = Image.fromarray(array, mode='I;16')

return img

B.2 Command line tool

"""Console script for mctal_to_image."""

import sys

import click

import logging

from .mctal_to_image **import** *


```

from PIL import Image
import numpy as np
from pathlib import Path
from sys import stdout

@click.group()

def main():
    """Console script for mctal_to_image."""
    logger = logging.getLogger(__name__)
    logger.setLevel(logging.DEBUG)

    formatter = logging.Formatter('%(asctime)s_%(name)s_%(levelname)s_%(
        message)s')

    ch = logging.StreamHandler(stdout)
    ch.setLevel(logging.DEBUG)
    ch.setFormatter(formatter)

    logger.addHandler(ch)

    return 0

@main.command()
@click.argument('input_filepath', type=click.Path(exists=True))
@click.argument('output_filepath', type=click.Path())
def probs(input_filepath, output_filepath):
    logger = logging.getLogger(__name__)
    logger.info('making_probability_matrices_from_mctal_files')

    idir = Path(input_filepath)
    odir = Path(output_filepath)

    # Create output filepath if needed
    if not (odir.exists()):

```

```

    odir.mkdir(parents=True)

guids = [str(f.name)[:8] for f in idir.iterdir() if '.dtp15.m' in str(f)]

for g in guids:
    ofile = odir / f'{g}.probs.npy'
    if not(ofile.exists()):
        try:
            p = mctal_to_prob(g, idir)
            assert p.shape == (15, 2, 3073, 32)
            np.save(ofile, p)
            logger.debug(f'created_{g}.probs.npy')
        except AssertionError as ae:
            print(f'output_shape_was_{p.shape},_not_(15,_2,_3073,_32)')
            raise ae
        except Exception as e:
            logger.debug(f'couldn\'t_process_{g}')
            raise e
    else:
        logger.debug(f'{g}_already_exists')

logger.info('created_all_probability_files')
return 0

```

```

@main.command()
@click.argument('input_filepath', type=click.Path(exists=True))
@click.argument('output_filepath', type=click.Path())
@click.option('--n', '--number', type=click.INT, default=1)
@click.option('--t', '--time', type=click.INT, default=600)
def images(input_filepath, output_filepath, n, t):
    logger = logging.getLogger(__name__)
    logger.info('generating_images_from_probabilities')

    idir = Path(input_filepath)
    odir = Path(output_filepath)

```

```

pfiles = [f for f in idir.iterdir() if '.probs.npy' in str(f)]

for pf in pfiles:
    g = str(pf.name)[:10]
    p = np.load(pf)
    for i in range(n):
        meas = make_measurements(p, time=t)
        blur = blur_measurements(meas)
        img = numpy_to_image(blur)
        logger.debug(f'saving_image_for_{g}')
        img.save(odir / f'{g}.{i:04d}.png')

logger.info('finished_generating_images')

return 0

@click.command()
@click.argument('input_filepath', type=click.Path(exists=True))
@click.argument('output_filepath', type=click.Path())
@click.option('--n', '--number', type=click.INT, default=1)
@click.option('--t', '--time', type=click.INT, default=600)
def measure(input_filepath, output_filepath, n, t):
    logger = logging.getLogger(__name__)
    logger.info('generating_measurements_from_probabilities')

    idir = Path(input_filepath)
    odir = Path(output_filepath)

    pfiles = [f for f in idir.iterdir() if '.probs.npy' in str(f)]

    for pf in pfiles:
        g = str(pf.name)[:10]
        p = np.load(pf)
        for i in range(n):

```

```

    meas = make_measurements(p, time=t)
    blur = blur_measurements(meas)
    logger.debug(f'saving_measurements_for_{g},_max_of_{blur.max()}')
    np.save(odir / f'{g}.{i:04d}.npy', blur)

logger.info('finished_making_measurements')

return 0

if __name__ == "__main__":
    sys.exit(main()) # pragma: no cover

```

C Neural network specification and training

```
import pandas as pd
from pathlib import Path
import numpy as np
from tensorflow.keras import layers, optimizers, callbacks, models, losses
from tensorflow.keras.applications.resnet50 import ResNet50
from PIL import Image
from datetime import datetime as dt

# Load the CSV file with the alignment values
ddir = Path('.') / 'data' / 'processed'
df = pd.read_csv(ddir / 'processed1.csv')
df.head()

# Construct the training and testing data for the network
X_train, X_test = [], []
y_train, y_test = [], []

for dx, dy, dt, guid, test in zip(df.dx, df.dy, df.dt, df.uuid, df.test):
    try:
        if test==1:
            X_test.append(np.array(Image.open(ddir / f'{guid}.0000.png'))))
            y_test.append([dx, dy, dt])
        elif test==0:
            X_train.append(np.array(Image.open(ddir / f'{guid}.0000.png'))))
            y_train.append([dx, dy, dt])
    except FileNotFoundError:
        pass

# normalize the layers
X_train = np.array(X_train) / (2**16-1)
X_test = np.array(X_test) / (2**16-1)
y_train = np.array(y_train)
y_test = np.array(y_test)
```

```

ymean = y_train.mean(axis=0)
ystd = y_train.std(axis=0)

y_train = (y_train - ymean) / ystd
y_test = (y_test - ymean) / ystd

print(X_train.shape, y_train.shape, X_test.shape, y_test.shape)

# Verify training mean is [0, 0, 0] and stddev is [1, 1, 1]
print(y_train.mean(axis=0), y_train.std(axis=0))

# Specify network architecture
# Input data and add color channel
inp = layers.Input(shape=(480, 480), name='Input')
x = layers.Reshape((480, 480, 1))(inp)

# Initial feature layer without residual
x = layers.Conv2D(64, (7, 7), (2, 2), padding='valid')(x)
x = layers.BatchNormalization()(x)
x = layers.MaxPool2D((3, 3), (2, 2), padding='valid')(x)

# First ResNet block
# Processing block
left = layers.Conv2D(64, (3, 3), padding='same')(x)
left = layers.BatchNormalization()(left)
left = layers.Activation('relu')(left)
left = layers.Conv2D(64, (3, 3), padding='same')(left)
left = layers.BatchNormalization()(left)
left = layers.Activation('relu')(left)

# Residual connection
right = layers.Conv2D(64, (1, 1), padding='same')(x)
right = layers.BatchNormalization()(right)

```

```

x = layers.Add()([left, right])

# Second ResNet block
# Processing block
left = layers.Conv2D(64, (3, 3), (2, 2), padding='same')(x)
left = layers.BatchNormalization()(left)
left = layers.Activation('relu')(left)
left = layers.Conv2D(64, (3, 3), padding='same')(left)
left = layers.BatchNormalization()(left)
left = layers.Activation('relu')(left)

# Residual connection
right = layers.Conv2D(64, (1, 1), (2, 2), padding='same')(x)
right = layers.BatchNormalization()(right)

x = layers.Add()([left, right])

# Third ResNet block
# Processing block
left = layers.Conv2D(64, (3, 3), (2, 2), padding='same')(x)
left = layers.BatchNormalization()(left)
left = layers.Activation('relu')(left)
left = layers.Conv2D(64, (3, 3), padding='same')(left)
left = layers.BatchNormalization()(left)
left = layers.Activation('relu')(left)

# Residual connection
right = layers.Conv2D(64, (1, 1), (2, 2), padding='same')(x)
right = layers.BatchNormalization()(right)

x = layers.Add()([left, right])

# Fourth ResNet block
# Processing block
left = layers.Conv2D(64, (3, 3), (2, 2), padding='same')(x)
left = layers.BatchNormalization()(left)

```

```

left = layers.Activation('relu')(left)
left = layers.Conv2D(64, (3, 3), padding='same')(left)
left = layers.BatchNormalization()(left)
left = layers.Activation('relu')(left)

# Residual connection
right = layers.Conv2D(64, (1, 1), (2, 2), padding='same')(x)
right = layers.BatchNormalization()(right)

x = layers.Add()([left, right])

# Fifth ResNet block
# Processing block
left = layers.Conv2D(64, (3, 3), (2, 2), padding='same')(x)
left = layers.BatchNormalization()(left)
left = layers.Activation('relu')(left)
left = layers.Conv2D(64, (3, 3), padding='same')(left)
left = layers.BatchNormalization()(left)
left = layers.Activation('relu')(left)

# Residual connection
right = layers.Conv2D(64, (1, 1), (2, 2), padding='same')(x)
right = layers.BatchNormalization()(right)

x = layers.Add()([left, right])

x = layers.Flatten()(x)

# Enforce feature orthogonality
x = layers.Dropout(0.25)(x)

# Dense layer to combine features nonlinearly
x = layers.Dense(512, activation='tanh')(x)

# Enforce feature orthogonality
x = layers.Dropout(0.25)(x)

```



```

# Output layer
out = layers.Dense(3, activation='linear')(x)

# Compile the model
resnet = models.Model(inputs=inp, outputs=out)

# RMSprop for training
opt = optimizers.RMSprop(learning_rate=1e-3)

# Set up callbacks for logging, LR scheduling
es = callbacks.EarlyStopping(patience=20)
lr = callbacks.ReduceLROnPlateau()
logdir = f'tblogs/{dt.now().strftime('%Y-%m-%d_%H-%M-%S')}'
tb = callbacks.TensorBoard(log_dir=logdir,
                           write_images=True, histogram_freq=25)
cbs = [es, lr, tb]

# LogCosh loss
loss = losses.LogCosh()

# Compile network
resnet.compile(loss=loss)

# Run training for up to 500 epochs (complete view of training set)
# Randomly hold back 20% of training data to validate
hist = resnet.fit(x=X_train, y=y_train, validation_split=0.2, epochs=500,
                 callbacks=cbs, verbose=True, batch_size=32)

print(resnet.summary())

# Save the model
mdir = Path('.') / 'models'
resnet.save(mdir / 'all_three_model_2020-05-05.h5')

# Execute model predictions and verify output shape is correct

```

```

testout = resnet.predict(X_test)
testout.shape  # should be (153, 3)

# Rescale the output
preds = testout * ystd + ymean
preds.shape  # should still be (153, 3)

# Rescale the test targets
targets = y_test * ystd + ymean

# Calculate prediction error/residuals
resids = targets - preds

print(f"Mean_of_errors:_{resids.mean(axis=0)}")
print(f"Standard_deviation_of_errors:_{resids.std(axis=0)}")

```

D Python script to align detectors to the same time scale

```
import numpy as np
from pathlib import Path
import pandas as pd
from sys import argv
from datetime import datetime
import json
import logging

NPIXELS = 28
NALPHAS = 15

def load_corr(file):
    try:
        data = np.loadtxt(file, skiprows=1, usecols=range(1, 421),
                           dtype=np.uint16, delimiter=',')
    except ValueError:
        # For some reason sometimes the second column is blank
        data = np.loadtxt(file, skiprows=1, usecols=range(2, 422),
                           dtype=np.uint16, delimiter=',')

    return data

@np.vectorize
def dtpno(colno, npixels=NPIXELS):
    return colno // npixels

@np.vectorize
def imgno(colno, npixels=NPIXELS):
    return colno % npixels

def time(dtp, img):
    ntime = 115.1 / 0.5126
```

```

centers = np.array([round(.6 + (i-8)*0.7, 1) for i in range(15)])
adists = np.sqrt(centers**2 + 10.0**2)
atimes = adists / 0.354
return np.round(ntime - atimes[ntp]).astype(np.int16) + 1024

def make_df(data):
    dtpnos = np.array(list(map(dtpno, range(data.shape[1]))))
    imgnos = np.array(list(map(imgno, range(data.shape[1]))))

    maxvals = data.max(axis=0)
    maxinds = data.argmax(axis=0)

    df = pd.DataFrame({'dtp': dtpnos, 'img': imgnos, 'maxval': maxvals,
                       'maxind': maxinds})
    df['dtp'] = df['dtp'].astype(np.int8)
    df['img'] = df['img'].astype(np.int8)

    df['time'] = time(df.dtp, df.img)
    df['offset'] = df.time - df.maxind

    return df

def make_filtered_df(df, threshold=100):
    filt = df[df.maxval > threshold].copy()
    return filt

def make_equation_parts(filt):
    s = np.zeros((NPIXELS + NALPHAS, filt.shape[0])).T

    for i, row in enumerate(filt.values):
        s[i, row[0]] = 1
        s[i, row[1]+NALPHAS] = 1

    o = np.zeros(s.shape[1])
    p = filt.maxind.values
    t = filt.time.values

```

```

    return s, o, p, t

def predict(s, o, p):
    return s @ o + p

def error(p, t):
    return 0.5 * (t - p)**2

def derror(p, t, s):
    return (p - t) @ s

def delt(s, o, p, t, lr=0.001):
    do = derror(predict(s, o, p), t, s)
    return lr * do

if __name__ == "__main__":
    log = logging.getLogger(__name__)
    log.addHandler(logging.StreamHandler())
    log.setLevel(logging.DEBUG)

    nowstring = datetime.now().strftime('%Y-%m-%d_%H-%M-%S')

    file = Path(argv[1])
    data = load_corr(file)
    df = make_df(data)
    filt = make_filtered_df(df)
    s, o, p, t = make_equation_parts(filt)

    for _ in range(500000):
        o -= delt(s, o, p, t)

    if o[0] == 0:
        log.debug("Imputing_dtp01_as_{o[3:8].mean()}")
        o[0] = o[3:8].mean()
    if o[1] == 0:

```

```

        log.debug("Imputing_dtp02_as_{o[3:8].mean()}")
        o[1] = o[3:8].mean()
    if o[2] == 0:
        log.debug("Imputing_dtp03_as_{o[3:8].mean()}")
        o[2] = o[3:8].mean()
    if o[-1] == 0:
        log.debug("Imputing_img28_as_{o[-8:-1].mean()}")
        o[-1] = o[-8:-1].mean()

offsets = {'input': file.name}

for dtpno, offset in zip(range(1, 16), o[:15]):
    offsets[f'dtp{dtpno:02d}'] = int(np.round(offset).astype(np.
        int))

for imgno, offset in zip(range(1, NPIXELS+1), o[15:]):
    offsets[f'img{imgno:02d}'] = int(np.round(offset).astype(np.
        int))

log.debug(f"img28_offset:{o[-1]}\tset_to_{offsets['img28']}")

with open(f'offsets_{nowstring}.json', 'w') as f:
    try:
        json.dump(offsets, f)
        log.info(f'Saved_offsets_to_offsets_{nowstring}.json')
    except TypeError as te:
        print(offsets)
        raise(te)

```

E An overview of neural network architectures examined in this work

Four neural network architectures were considered for this work: densely connected networks [47], recurrent networks [48], convolutional networks [49], and the ResNet architecture [10] which was finally employed. Of these, recurrent architectures were rejected after brief experimentation as difficult to train to a stable state. Densely connected networks are also known as feedforward networks or multi-layered perceptrons and were first described by Rosenblatt [47]. These networks are organized into “layers” consisting of some number of “neurons.” The input to each neuron is the weighted sum of the outputs of all of the neurons in the previous layer. At a minimum these networks have two layers: an input layer and an output layer, though this configuration can only model linear systems. With the addition of a “hidden” layer between the input and output, the network can model any function [50]. In practice this may require large numbers of neurons. This was the case with attempts to model the relationship between simulated NMIS data and neutron array position. Networks with progressively larger numbers of neurons were trained on sample data but exhibited errors and were unable to consistently predict even the training data. The largest network examined had eight fully-connected layers (in addition to input and output layers) for a total of 178 294 784 trainable parameters. The network was only able to capture the value of the distribution. Further, it was very sensitive to added noise, making it unsuitable for repeated measurements.

Densely connected networks are very flexible but are generally inferior to convolutional neural networks (CNN) at image recognition tasks given finite bounds of computer memory and training time [49]. CNNs take advantage of two intuitions: that the relationship between nearby pixels in an image is more closely related to the desired output than the relationship between widely separated pixels, and that the nature of the relevant relationship between nearby pixels is invariant over the image. This has the effect of reducing the memory required for networks with large numbers of layers [51]. Two broad convolutional architectures were examined as a part of this work. These networks were trained using the ADADELTA learning method [52], which adjusts the learning rate upward when the gradient in the error surface is

low. This method was selected for evaluation of network prototypes as a broadly applicable with low sensitivity to hyperparameter selection. The first network split the input into three-dimensions: alpha pixel, neutron pixel, and time. It then applied three dimensional convolution windows to the data. The goal was to separate the information correlated to each alpha channel. The second used two-dimensional convolution windows covering the entire input. The size of these windows was varied in both dimensions from 7x7 pixels to 1x20 (width, height). While these networks exhibited less sensitivity to noise, both of them reverted to predicting the mean value of each array and were therefore rejected.

Vita

J.R. Powers-Luhn was commissioned as a naval officer following his undergraduate studies in Physics at the University of Virginia. After initial nuclear power and submarine training he served on board USSS CHARLOTTE and USS HYMAN G RICKOVER in primarily engineering roles, followed by a three year shore tour in Yokosuka, Japan. In 2011, he separated from active duty and worked for an energy efficiency startup designing neural networks to predict building energy demand. In 2016 he returned to school, focusing his research on applying neural networks to detectors with non-proliferation applications.