Analysis of Process Criticality Accident Risk Using a Metamodel-Driven Bayesian Network

by William John Zywiec

BS in Nuclear Engineering, May 2013, Rensselaer Polytechnic Institute
MSE in Systems Engineering, May 2017, The Johns Hopkins University

A Dissertation submitted to

The Faculty of
The School of Engineering and Applied Science
of The George Washington University
in partial satisfaction of the requirements
for the degree of Doctor of Philosophy

November X, 2020

Dissertation directed by

Shahram Sarkani
Professor

Thomas Andrew Mazzuchi
Professor

The School of Engineering and Applied Science of The George Washington University certifies that William John Zywiec has passed the Final Examination for the degree of Doctor of Philosophy as of November X, 2020. This is the final and approved form of the dissertation.

Analysis of Process Criticality Accident Risk Using a Metamodel-Driven Bayesian Network

William John Zywiec

Dissertation Research Committee:

Shahram Sarkani, Professor, Dissertation Director

Thomas Andrew Mazzuchi, Professor, Dissertation Co-Director

Dedication

For my family.

Acknowledgments

I would like to thank my wife, Katrina, for supporting me throughout my undergraduate and graduate education, and for going along with my decision to pursue a PhD. I would also like to thank my advisors, Dr. Shahram Sarkani and Dr. Thomas Mazzuchi, for providing invaluable advice and guidance, and for putting together and managing an excellent program. Additional thanks go to Daniel Falbel, for answering questions about *TensorFlow*, and Dr. Shankar Kulumani, for putting together the LaTeX template I used to write this dissertation. Lastly, I would like to thank David Heinrichs and Dr. Thomas McLaughlin, because "if I have seen further it is by standing on the shoulders of giants."

Abstract

Analysis of Process Criticality Accident Risk Using a Metamodel-Driven Bayesian Network

Since the discovery of fission in 1938, there have been more than 60 criticality accidents throughout the world. These accidents are divided into two categories: those that occur during critical experiments or operations with research reactors, and those that occur in production facilities, more commonly referred to as *process criticality accidents*. The focus of this work is on the development of a methodology that uses a Bayesian network and a neural network metamodel to estimate process criticality accident risk.

List of Figures

List of Tables

List of Abbreviations

API Application Programming Interface

CPU Central Processing Unit

DOE U.S. Department of Energy

ENDF Evaluated Nuclear Data File

FORM First Order Reliability Method

GPU Graphics Processing Unit

IAEA International Atomic Energy Agency

INES International Nuclear and Radiological Event Scale

MSE Mean Squared Error

NNSA National Nuclear Security Administration

SORM Second Order Reliability Method

SS304 304 Stainless Steel

SSE Sum of Squared Errors

TPU Tensor Processing Unit

Chapter 1: Introduction

A criticality accident is an uncontrolled fission chain reaction that occurs when fissionable material (e.g., plutonium, uranium) is inadvertently assembled into a critical or supercritical mass. Since the discovery of fission in 1938 [31], there have been more than 60 criticality accidents throughout the world [50]. These accidents are divided into two categories: those that occur during critical experiments or operations with research reactors, and those that occur in production facilities, more commonly referred to as *process criticality accidents* [50]. The focus of this work is on the development of a methodology that uses a Bayesian network and a neural network metamodel to estimate process criticality accident risk.

For those who are unfamiliar with criticality accidents, there is the International Nuclear and Radiological Event Scale (INES), which was introduced in 1990 by the International Atomic Energy Agency (IAEA), to enable prompt communication of the safety significance of accidents and incidents to a wider audience [6]. INES goes from 0 to 7 and is intended to be logarithmic, with each level representing an accident approximately ten times as severe as the next level down. The Chernobyl disaster, for example, was an INES level 7 event, which is defined as a "major release of radioactive material with widespread health and environmental effects requiring implementation of planned and extended countermeasures" [6]. On the other end of the scale is an INES level 0 event, which is a "deviation" with no off-site consequences [6]. Process criticality accidents are typically reported as INES level 3 or 4 events, which are defined as either a "serious incident" or an "accident with local consequences" [6, 36].

Historically, fault tree analysis has been the main methodology used by the U.S. Department of Energy (DOE) to estimate process criticality accident risk [2]. Fault tree analysis consists of defining one or more initiating events, building logic gates for all foreseeable failure paths, and applying Boolean algebra to calculate the top-level failure probability [59].

Over the years, several improvements have been made to this methodology by incorporating binary decision diagrams [57, 58], dynamic fault trees [29, 60], and Monte Carlo simulations [28]. However, fault tree analysis still relies heavily on identifying events, failure paths, and minimum cut sets, which is difficult to do for fissionable material operations and other systems with a high degree of dimensionality [10, 15, 37].

For operations with fissionable material, there are ten parameters that affect nuclear criticality: mass, absorption, geometry/shape, interaction, concentration/density, moderation, enrichment, reflection, volume, and temperature [39]. Most parameters are observable, but even with a perfect understanding of all ten, a radiation transport code is still needed to calculate the effective neutron multiplication factor (k_{eff}) of the system, in order to determine if a criticality accident will occur. A simplified form of the equation to calculate k_{eff} is shown in 1.1 [43].

$$k_{eff} = \frac{\text{neutrons produced from fission in one generation}}{\text{neutrons produced from fission in the previous generation}} \tag{1.1}$$

If $k_{eff} < 1$, the system is subcritical—and presumably safe. If $k_{eff} \geq 1$, the system will produce either a self-sustaining or divergent fission chain reaction and an accompanying burst of neutron and gamma ray radiation (i.e., a criticality accident) [39, 43]. For nuclear criticality applications, k_{eff} is typically calculated using a Monte Carlo radiation transport code, such as MCNP [3]. A Monte Carlo radiation transport code works by reading input and building a computer model of the system, which consists of a specific geometric arrangement of fissionable and non-fissionable materials. The code then simulates particle interactions and calculates a probabilistic value for k_{eff}, based on the number and types of interactions that occur (e.g., fission, scattering, absorption) [3].

In this work, a Bayesian network is used to model parameters (or variables) that affect nuclear criticality, and a neural network metamodel is trained to predict k_{eff}, based on calculations that were performed using MCNP [3] and ENDF/B-VII.1 nuclear data [17].

Table 1.1: Process criticality accident risk at U.S. Department of Energy sites

	accidents/year	reference
Lawrence Livermore National Laboratory	3.2e-05	DOE/EIS-0348
Los Alamos National Laboratory	$<$ 1e-04	DOE/EIS-0283
Nevada National Security Site	$\leq$ 1e-02	DOE/EIS-0426
Pantex Plant	$<$ 1e-06	DOE/EIS-0098
Sandia National Laboratories	$<$ 1e-07	DOE/EIS-0281
Savannah River Site	1e-02	DOE/EIS-0541
Y-12 National Security Complex	$\leq$ 1e-02	DOE/EIS-0387

This work also describes how this methodology is applied to fissionable material operations in the Plutonium Facility (Building 332) at Lawrence Livermore National Laboratory. Building 332 is a multi-laboratory facility that supports a wide variety of activities that are sponsored by the National Nuclear Security Administration (NNSA) [1]. Although the risk of a process criticality accident in Building 332 is low [1], the consequences are extreme, due to the impact a criticality accident would have on the safety and stewardship of the United States' nuclear stockpile [4, 5].

1.1 Literature Review

The motivation for this work originated while reviewing Environmental Impact Statements for U.S. Department of Energy sites and noting that radiation transport codes were not used to analyze the frequency or severity of process criticality accidents (Table 1.1). At the time, this was considered somewhat strange, since radiation transport codes are commonly used in the field of nuclear engineering to design experiments and evaluate fissionable material operations in nuclear facilities. After further review, it appeared that most DOE Environmental Impact Statements use a simplified form of fault tree analysis, which relies on overly broad and unrealistic assumptions of accident conditions [48].

A summary of the Safety Analysis Report (SAR) at the Rocky Flats Plant [48] provides some insight into how DOE sites estimate process criticality accident risk. The following paragraphs are taken from R.J. Mattson [48]:

> *The SAR provides the frequencies for groups of initiating events but does not provide any information on the causes for individual initiating events within a group. A particular concern is the treatment of human errors. Although the involvement of humans in the process is mentioned, the treatment of human errors in the risk analysis is not addressed. The SAR does not include sufficient detail. At least basic assumptions and data, as well as the process, used to derive the results need to be documented. The example calculation in Appendix C of the SAR is insufficient.*
>
> *Further, in the course of calculating the risk of criticality accidents, there has not been a systematic, detailed study of the specific operations performed at the Rocky Flats Plant, nor an estimate of the associated probability of human error. Such a study might well produce results different from those resulting from generic data for human error that were used. Of even more importance to safe operation at the plant is the fact that a detailed study of the plant-specific operations might identify possibilities for human error that have not yet been observed. This could be a rich source of new safety information.*

Even though this summary was published in 1989, it is still relevant today. The SAR that is referenced was written in 1987, and it estimated the site-wide risk of a criticality accident to be 3e-04 accidents/year—six years before a process criticality accident nearly occurred in Building 771 [49].

To illustrate the complexity of fissionable material operations, the results of two different sets of MCNP calculations are plotted and shown in Figures 1.1 and 1.2. The first plot represents a sphere of alpha-phase plutonium ($\rho = 19.86$ g/cm^3) homogeneously dispersed

in water, with one inch of water reflection (Figure 1.1). The second plot represents a sphere of alpha-phase plutonium homogeneously dispersed in magnesium oxide, with one inch of water reflection (Figure 1.2). Even though the only difference between these plots is moderation, the k_{eff} contours are much different (Figure 1.2)—and these differences compound when other parameters that affect nuclear criticality are changed [39].

1.1.1 Bayesian Network

The modeling of high-dimensional, nonlinear systems led to the selection of a Bayesian network [54] as one-half of the methodology described in this work. One benefit of using a Bayesian network is that complex mechanisms leading to top-level failure do not need to be well understood prior to building the model [37, 54, 63]. The only things that are needed are an understanding of parameters that affect nuclear criticality [39] and the ability to model each one. Another benefit of using a Bayesian network is that a single network can model multiple modes and/or causes of failure [14, 66]. Both of these benefits are ideal for this application, since it is generally easier to model parameters and estimate probabilities when there isn't a need to evaluate low- or mid-level failure. One downside of using a Bayesian network is that its ability to model latent variables (e.g., k_{eff}) is constrained by the size and complexity of the model [40, 66]. Existing methods, such as FORM or SORM, use a Taylor series expansion to approximate the "most probable failure point", while others use Monte Carlo variance reduction techniques to select samples that are in the "vicinity of failure" [15]. Recently, a new class of methods have been developed, which use a sampling algorithm coupled to a surrogate model (or *metamodel*) to approximate a performance function [15].

1.1.2 Neural Network Metamodel

The neural network metamodel described in this work is based on prior research in the field of structural engineering, which focuses on using metamodels to reduce the computational burden of performing direct, simulation-based analysis of physical systems [16, 27, 53, 56,

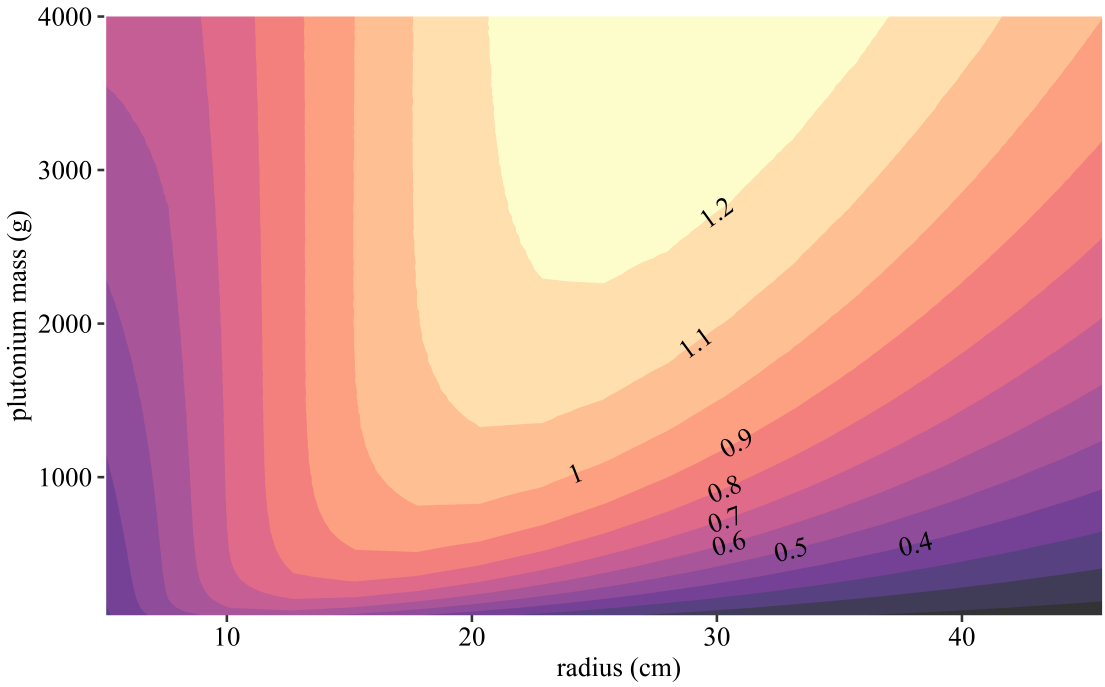

Figure 1.1: *keff* contour plot of a sphere of alpha-phase plutonium (95% ^{239}Pu, 5% ^{240}Pu by weight) with water moderation and one inch of water reflection

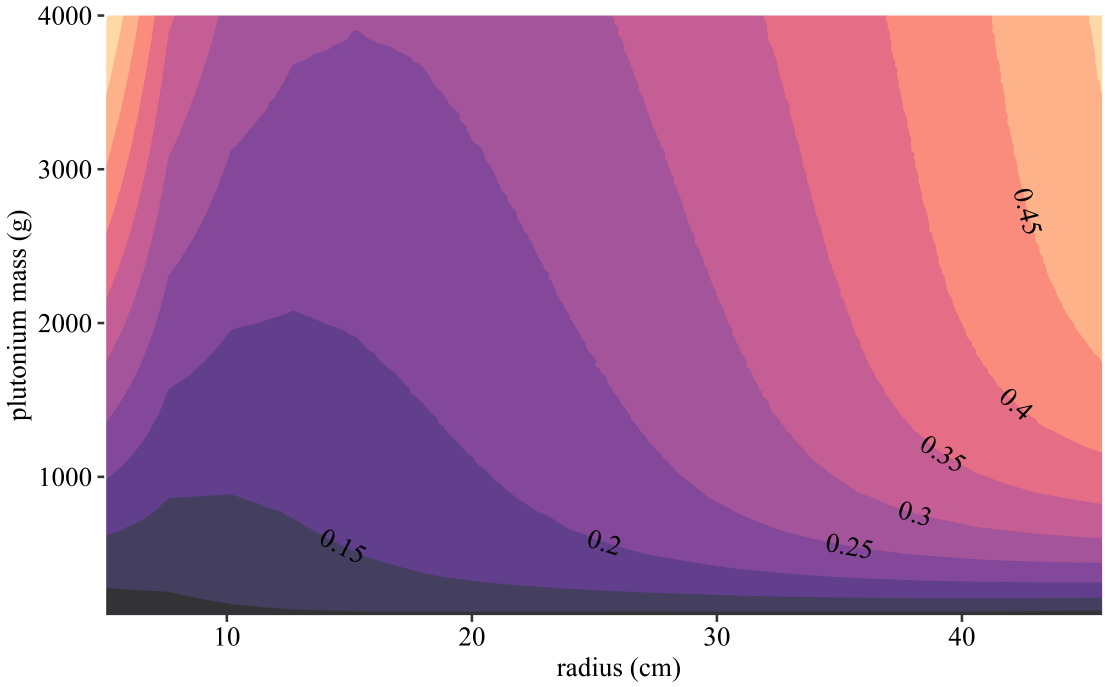

Figure 1.2: *keff* contour plot of a sphere of alpha-phase plutonium (95% ^{239}Pu, 5% ^{240}Pu by weight) with magnesium oxide moderation and one inch of water reflection

69]. There are several different types of metamodels that can be used to approximate a performance function, including Gaussian process regression models [27, 55, 64], neural networks [30, 33], polynomial chaos expansions [13, 21, 52], quadratic response surfaces [8, 32], random forests [42, 46], and support vector machines [11, 61, 65]. Neural networks were selected for further use in this work, based on two white papers [7] that describe GPU-accelerated tensor operations (i.e., vector and matrix calculations), which neural networks are preconfigured to use [7, 18]. After initially renting GPU-hours on the Amazon Elastic Compute Cloud (EC2), I built a computer with an NVIDIA GeForce GTX 1080 8 GB GDDR5X GPU, specifically to work on this dissertation.

1.1.3 Random Sampling Algorithms

The methodology described in this work is based on random sampling algorithms described in F. Cadini and A. Gioletta [15], F. Cadini et al. [16], V. Dubourg et al. [27], D. Koller and N. Friedman [40], and M. Papadrakakis and N.D. Lagaros [53].

1.1.4 Software Packages

The *bnlearn* [62], *Keras* [18], and *TensorFlow* [7] software packages were used throughout this work. These software packages were selected because they are popular, well documented, and written in R, a programming language that has several graphical and statistical software packages that were also used throughout this work [22, 25, 34, 41, 62, 68]. *bnlearn* [62] was used to build the Bayesian network and generate random samples. Other Bayesian network software packages, including BUGS and JAGS, were also considered, but not selected, due to lack of integration with either Python or R. The *Keras* [18] and *TensorFlow* [7] software packages were used to build the neural network metamodel. *Keras* is a high-level API, which manages interactions between Python/R and the *TensorFlow* platform [18]. *TensorFlow* is an end-to-end, open source machine learning platform, which combines four key abilities [7]:

- Efficiently executing low-level tensor operations on CPUs, GPUs, and TPUs
- Computing the gradient of arbitrary differentiable expressions
- Scaling computation to many devices
- Exporting programs to external servers, browsers, mobile, and embedded devices

Caffe and *PyTorch* were also considered, but not selected, due to being relatively new and untested when research for this work began in 2017.

1.2 Contributions

The primary contribution of this work is the development of a methodology that uses a Bayesian network and neural network metamodel to estimate process criticality accident risk. Although neural network metamodels are not new, this work demonstrates the novel use of a metamodel in the context of performing a Bayesian network-based probabilistic risk assessment. The main benefit of this methodology is that it combines the interpretability and random sampling algorithm of a Bayesian network with the high-dimensional, latent variable modeling capability of a neural network metamodel. The secondary contribution of this work is the implementation and testing of a sum of squared errors (SSE) loss function, which is shown to outperform the more commonly used mean squared error (MSE) loss function.

1.2.1 Publications

Chapter 2: Mathematical Background

This chapter provides a mathematical background in nuclear criticality, Monte Carlo radiation transport, Bayesian networks, and neural networks. The equations and figures included in this chapter, while not exhaustive, provide context for later sections of this work, which focus on building a coupled Bayesian network and neural network metamodel and using it to estimate process criticality accident risk.

2.1 Nuclear Criticality

Nuclear fission is an event that occurs when a target nucleus absorbs a neutron, transitions to an excited state, and then de-excites by splitting into two or more fission products and emitting neutron and gamma ray radiation (Figure 2.1). This phenomenon was discovered on December 17, 1938 by Otto Hahn and Fritz Strassmann [31], and later explained on Feburary 11, 1939, in an article written by Lise Meitner and Otto Frisch [51].

Nuclear criticality is a self-sustaining or divergent fission chain reaction that occurs when fissionable material is assembled into a critical or supercritical mass [39]. Nuclear criticality is often expressed as a steady-state calculation of the *effective neutron multiplication factor* (k_{eff}), which is the number of neutrons produced from fission in one generation, divided by the number of neutrons produced from fission in the previous generation (1.1). If the number of neutrons produced from fission in one generation is equal to the number of neutrons produced from fission in the previous generation, then the system is self-sustaining or critical ($k_{eff} = 1$). If it is greater, the system is supercritical ($k_{eff} \geq 1$). If it is less, the system is subcritical ($k_{eff} < 1$).

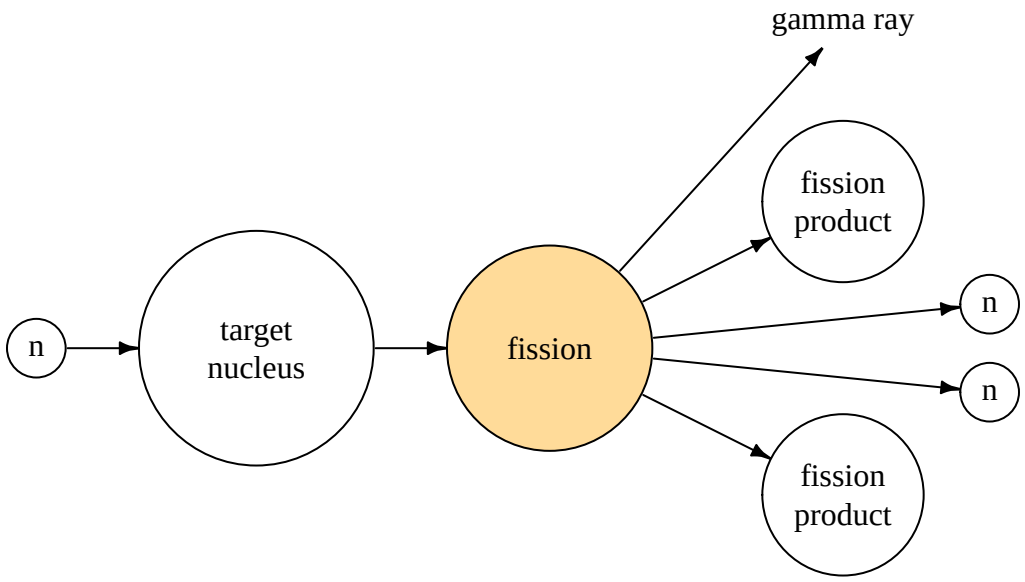

Figure 2.1: Fission diagram

2.1.1 Nuclear Cross Sections

Nuclear cross sections are used to describe the probability that a reaction will occur between a particle (e.g., a neutron) and a target nucleus (Figure 2.2). The concept of a cross section is quantified in terms of "characteristic area", where a larger area means a larger probability of interaction. The standard unit for measuring a nuclear cross section (σ) is the *barn*, which is equal to 10^{-24} cm^2 [43].

Nuclear cross sections depend on the size of the target nucleus, the type of reaction that occurs (e.g., fission, scattering, absorption, etc.), the energy of the incident particle, and, to a lesser extent, the temperature of the target nucleus and the relative angle between the incident particle and target nucleus. Nuclear cross sections are also sometimes referred to as the *microscopic cross sections*, to distinguish them from *macroscopic cross sections*, which are weighted by the number density (N) of the target material (2.1).

$$\Sigma_t(E) = N\sigma_t(E) = \text{macroscopic total cross section (cm}^{-1}) \tag{2.1}$$

where

N = the number density of the target material (atoms/cm^3)

σ_t = the microscopic total cross section (cm^2)

The macroscopic total cross section represents the sum of all macroscopic cross sections for the target material, which includes capture (Σ_c), fission (Σ_f), scattering (Σ_s), and other rare events, such as (n,γ), (n,p), $(n,2n)$, and $(n,3n)$ reactions. If a neutron is captured, it is considered to be removed from the system. If a neutron, traveling in direction Ω at energy E, scatters off a target nucleus, it emerges from the scattering event with a new direction Ω' and energy E'. The distribution of post-collision directions Ω' and energies E' is expressed as $p(\Omega \cdot \Omega', E \rightarrow E')d\Omega' dE'$.

In Monte Carlo radiation transport codes, particle interactions are simulated by sampling the incremental probability of interaction per incremental distance traveled, as well as the angular and energy distributions, depending on the type of interaction that occurs. Plots of these angular and energy distributions are shown in Figure 2.3 and Figure 2.4 for several different incident neutron energies.

2.1.2 Monte Carlo Radiation Transport

MCNP is a general-purpose Monte Carlo radiation transport code that is used throughout this work. MCNP operates by reading input and building a computer model of the system, which consists of a specific geometric arrangement of fissionable and non-fissionable materials. The code then simulates neutron interactions and calculates a probabilistic value for k_{eff}, based on the number and types of interactions that occur (e.g., fission, scattering, absorption) [3]. The MCNP input deck, shown in Figure 2.5, represents 500 grams of alpha-phase plutonium ($\rho = 19.86$ g/cm^3), homogeneously dispersed in a sphere of water, with one inch of water reflection (Figure 2.6). The material cards are formatted by ZAID, which consists of a two-digit atomic number, a two- to three-digit mass number, and an alphanumeric suffix,

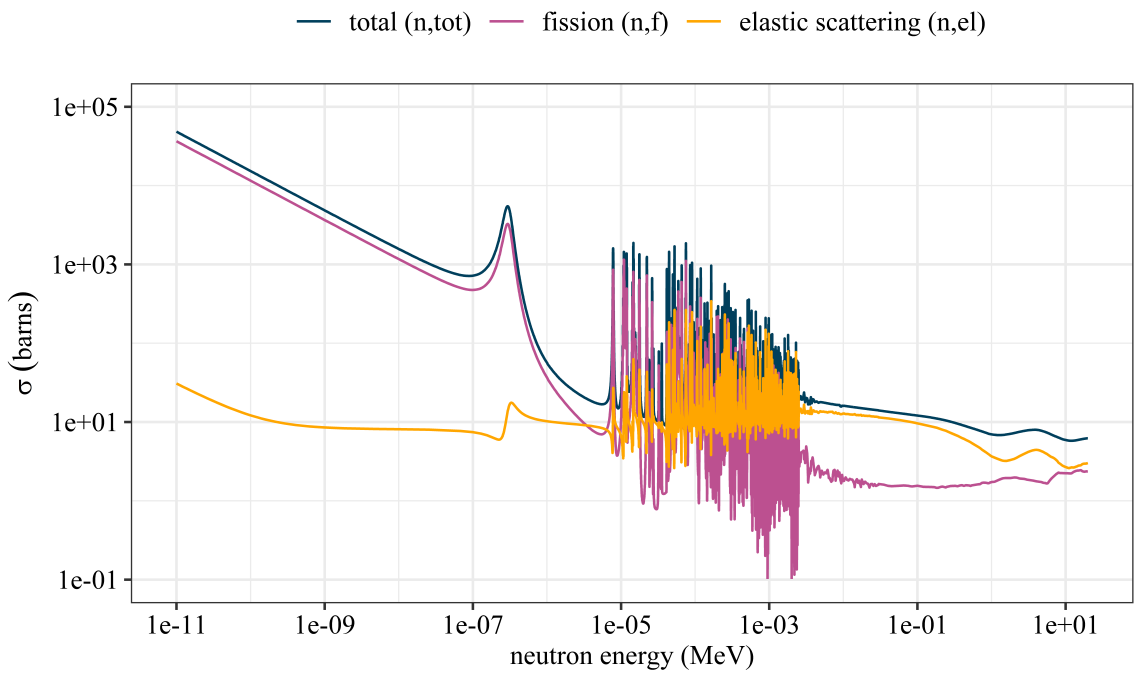

Figure 2.2: ^{239}Pu cross sections

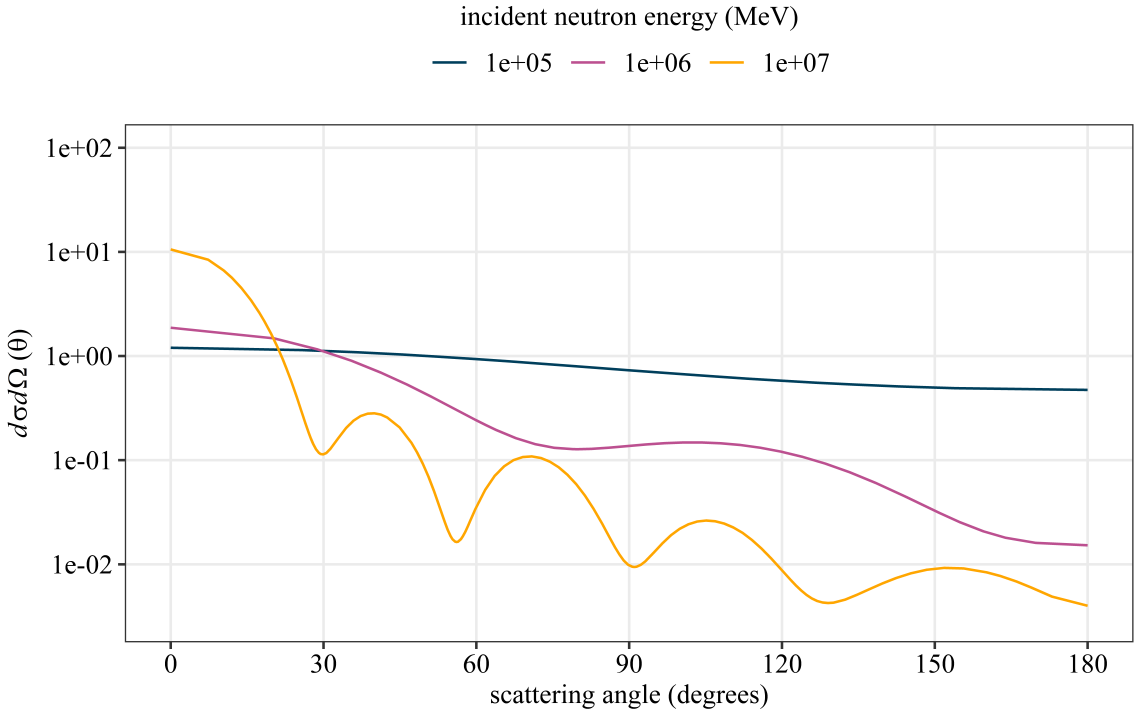

Figure 2.3: ^{239}Pu elastic scattering angular distributions

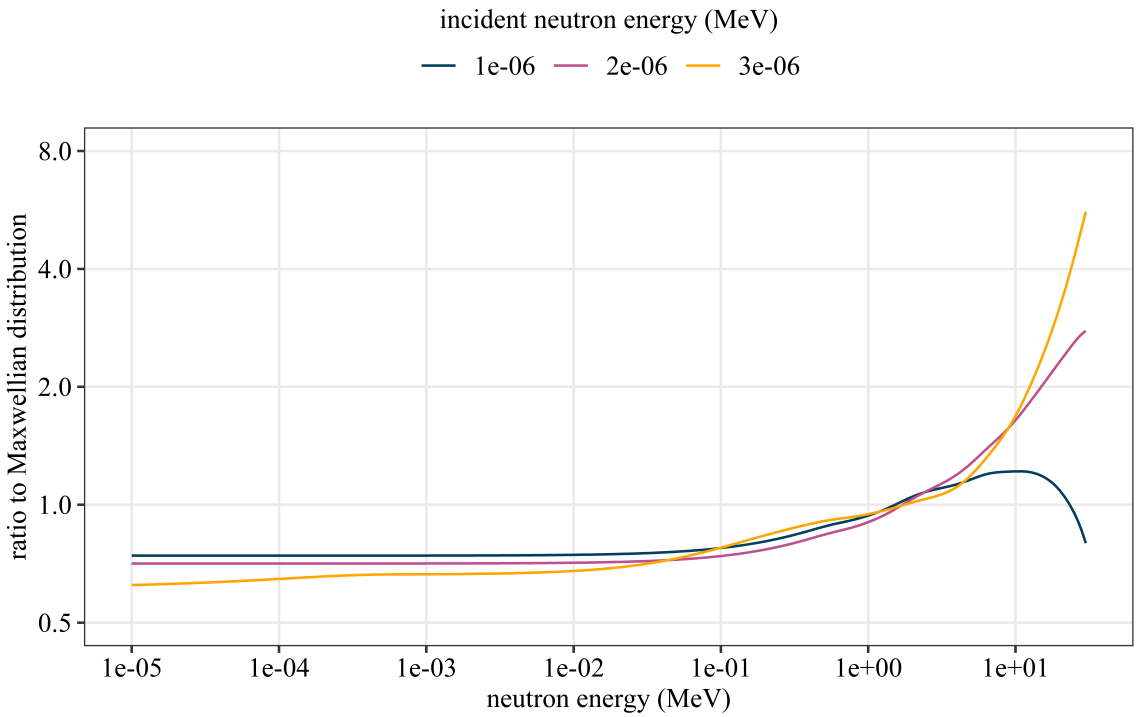

Figure 2.4: ^{239}Pu (n,f) energy distributions

which corresponds to an *xsdir* data table entry and nuclear data files that are installed in the same directory as MCNP. The ".80c" and ".20t" suffixes correspond to nuclear cross sections and S(α, β) thermal scattering treatments based on ENDF/B-VII.1 [17].

There are seven source points specified in the MCNP input decks (Figure 2.5): one at the origin, and six located along the positive and negative x, y, and z axes, at a distance from the origin equal to $2/3$ the radius of the sphere. The *kcode* data card is set to simulate random walks for 500 batches of 10,000 neutrons each (Figure 2.5), from a randomly sampled Watt fission spectrum for ^{239}Pu [3]. After the first batch of random walks, the average number of neutrons produced from fission is tallied, and locations where fissions occur are selected as the new source points for the next batch. The *kcode* data card is set to only calculate k_{eff} for batches 50 through 500, to ensure source points are evenly distributed throughout the system (Figure 2.5).

There are three estimators that MCNP uses to calculate k_{eff}: collision, absorption, and

```
500 alpha h2o 13.97 none sph
c
c cell cards
1  1 -1.03960849037536e+00 -1     $ Pu-H2O sphere density = 1.0396 g/cm^3
2  2 -9.98027000000000e-01 +1 -2 $ H2O reflector density = 0.9980 g/cm^3
3  0 +2

c surface cards
1  so  13.97 $ Pu-H2O sphere radius = 13.97 cm (5.5 in)
2  so  16.51 $ H2O reflector radius = 16.51 cm (6.5 in)

c material cards
m1    1001.80c -1.07201735161097e-01 $ H-1
      8016.80c -8.50684665694693e-01 $ O-16
     94239.80c -4.00079191869996e-02 $ Pu-239
     94240.80c -2.10567995721051e-03 $ Pu-240
mt1  lwtr.20t
m2    1001.80c +2 $ H-1
      8016.80c +1 $ O-16
mt2  lwtr.20t
c
imp:n 1 1 0
c
c source cards
kcode 10000 1 50 500
ksrc  0 0 0
      9.31 0 0
      0 9.31 0
      0 0 9.31
      -9.31 0 0
      0 -9.31 0
      0 0 -9.31
c
print
```

Figure 2.5: MCNP input deck

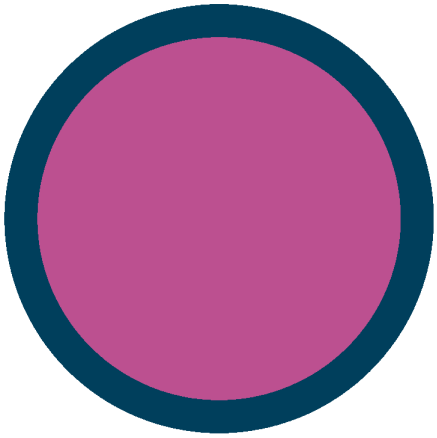

Figure 2.6: MCNP model of 500 grams of alpha-phase plutonium homogeneously dispersed in a sphere of water, with one inch of water reflection

track length. The collision estimator is calculated using 2.2.

$$k_{eff}^{c} = \frac{1}{N} \sum_{i} W_i \frac{\sum_k f_k \bar{\nu}_k \sigma_{f_k}}{\sum_k f_k \sigma_{t_k}} \tag{2.2}$$

where

k_{eff}^{c} = collision estimator

i is summed over all collisions in a cycle

k is summed over all isotopes of the target material involved in the i^{th} collision

N = number of source particles

W_i = weight of collision particle

f_k = atom-fraction of isotope k

$\bar{\nu}_k$ = average number of neutrons produced from fission

σ_{f_k} = microscopic fission cross section

σ_{t_k} = microscopic total cross section

The fraction to the right of W_i represents the number of neutrons produced from all fissions in the i^{th} collision. Multiplying this value by the weight of the collision particle (W_i), summing over all collisions in a cycle, and then dividing by the number of source particles results in the mean number of neutrons produced from fission for each cycle. The absorption estimator is calculated using 2.3.

$$k_{eff}^{a} = \frac{1}{N} \sum_{i} W_i \bar{\nu}_k \frac{\sigma_{f_k}}{\sigma_{c_k} + \sigma_{f_k}} \tag{2.3}$$

where

k_{eff}^{a} = absorption estimator

σ_{c_k} = microscopic capture cross section

The absorption estimator is only summed over the i^{th} collision for the k^{th} fissionable isotope. This is different from the collision estimator, which is summed over all isotopes. The track

length estimator is calculated using 2.4.

$$k^t_{eff} = \frac{1}{N} \sum_i W_i \rho d \sum_k f_k \bar{v}_k \sigma_{f_k} \tag{2.4}$$

where

k^t_{eff} = track length estimator

i is summed over all neutron trajectories

ρ = atomic density of the cell

d = neutron trajectory track length

MCNP takes the average of all k_{eff} values for each estimator, for all active cycles, and uses it to calculate the *maximum likelihood estimate* of k_{eff} (2.5).

$$k^x_{eff} = \frac{1}{C} \sum_{50}^{C} k^x_{eff,c} \tag{2.5}$$

where

C = number of active cycles

$k^x_{eff,\,c}$ = estimate of k_{eff} for each active cycle

Assuming that the $k^x_{eff,\,c}$ for each estimator are independent, the sample standard deviation of the mean estimate of k_{eff} can be obtained by 2.6 [67].

$$\sigma_{\bar{k}^x_{eff}} = \frac{1}{\sqrt{C}} \sigma_{k^x_{eff}} = \left[\frac{1}{C(C-1)} \sum_{c=50}^{C} (k^x_{eff,\,c} - \bar{k}^x_{eff})^2 \right]^{1/2} \tag{2.6}$$

where

$\sigma_{\bar{k}_{eff}}$ = standard deviation of $\bar{k}_{eff}$

$\sigma_{k_{eff}}$ = standard deviation of each estimator's k_{eff} values

The standard deviation of each estimator's k_{eff} values are used to calculate the corresponding population covariance matrix (2.7) [67].

$$\Sigma = \begin{bmatrix} \sigma^2_{k^c_{eff}} & \sigma^2_{k^{ca}_{eff}} & \sigma^2_{k^{ct}_{eff}} \\ \sigma^2_{k^{ac}_{eff}} & \sigma^2_{k^a_{eff}} & \sigma^2_{k^{at}_{eff}} \\ \sigma^2_{k^{tc}_{eff}} & \sigma^2_{k^{ta}_{eff}} & \sigma^2_{k^t_{eff}} \end{bmatrix} \tag{2.7}$$

A square root solution of a 3×3 identity matrix is multiplied by Σ (2.7) and its transpose to obtain a 3×3 matrix, which further reduces to a 2×2 matrix with $i \times j$ (Σ_{zij}) submatrices (2.8) [67].

$$\begin{aligned} \Sigma_z &= A\Sigma A^T \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} \sigma^2_{k^c_{eff}} & \sigma^2_{k^{ca}_{eff}} & \sigma^2_{k^{ct}_{eff}} \\ \sigma^2_{k^{ac}_{eff}} & \sigma^2_{k^a_{eff}} & \sigma^2_{k^{at}_{eff}} \\ \sigma^2_{k^{tc}_{eff}} & \sigma^2_{k^{ta}_{eff}} & \sigma^2_{k^t_{eff}} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \\ &= \left[\begin{array}{c|cc} \sigma^2_c & \sigma^2_c - \sigma^2_{ca} & \sigma^2_c - \sigma^2_{ct} \\ \hline \sigma^2_c - \sigma^2_{ca} & \sigma^2_c + \sigma^2_a - 2\sigma^2_{ca} & \sigma^2_c + \sigma^2_{at} - \sigma^2_{ca} - \sigma^2_{ct} \\ \sigma^2_c - \sigma^2_{ct} & \sigma^2_c + \sigma^2_{at} - \sigma^2_{ca} - \sigma^2_{ct} & \sigma^2_c + \sigma^2_t - 2\sigma^2_{ct} \end{array}\right] \\ &= \left[\begin{array}{c|c} \Sigma_{z11} & \Sigma_{z12} \\ \hline \Sigma_{z21} & \Sigma_{z22} \end{array}\right] \end{aligned} \tag{2.8}$$

Σ_{z12} and Σ_{z22} are converted to sum of squares matrices with $(n-1)$ degrees of freedom (2.9) [67].

$$S_{zij} = (n-1)\Sigma_{zij} \tag{2.9}$$

A (2.8) is also used to calculate $\bar{d}$ (2.10), which is used in conjunction with the collision estimator (2.2) and sum of squares matrices (2.9) to calculate the *maximum likelihood*

estimate of k_{eff} (2.11) [67].

$$A \begin{bmatrix} \bar{k}^c_{eff} \\ \bar{k}^a_{eff} \\ \bar{k}^t_{eff} \end{bmatrix} = \begin{bmatrix} \bar{k}^c_{eff} \\ \bar{d} \end{bmatrix} \tag{2.10}$$

$$k_{eff} = \bar{k}^c_{eff} - S_{z12} S^{-1}_{z22} \bar{d} \tag{2.11}$$

2.11 is essentially the collision estimator (2.2) modified by the variance-weighted residual between the collision estimator and the other two estimators (2.3, 2.4). The k_{eff} from 2.11 is reported as the final result in the MCNP output file.

2.2 Bayesian Networks

A *Bayesian network* is a type of a directed acyclic graph that uses Bayesian inference (2.12) to calculate the posterior distribution of a set of variables and their conditional dependencies [40, 54].

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.12}$$

The Bayesian network, shown in Figure 2.7, is comprised of 8 nodes and 13 arcs, which represent fissionable material operations (*op*), criticality controls (*ctrl*), and parameters that affect nuclear criticality. The Bayesian network, shown in Figure 2.7, is modeled using a combination of discrete and continuous probability distributions that have been converted into discrete probability tables. The arcs (Figure 2.7) represent conditional dependencies, which are calculated using 2.13.

$$p(x_B) = \prod_{b \in B} p(x_b | x_{parent(b)}) \tag{2.13}$$

where

$p(x_B) =$ joint probability distribution for B

$p(x_b|x_{parent(b)}) =$ probability distribution for b given the parents of b

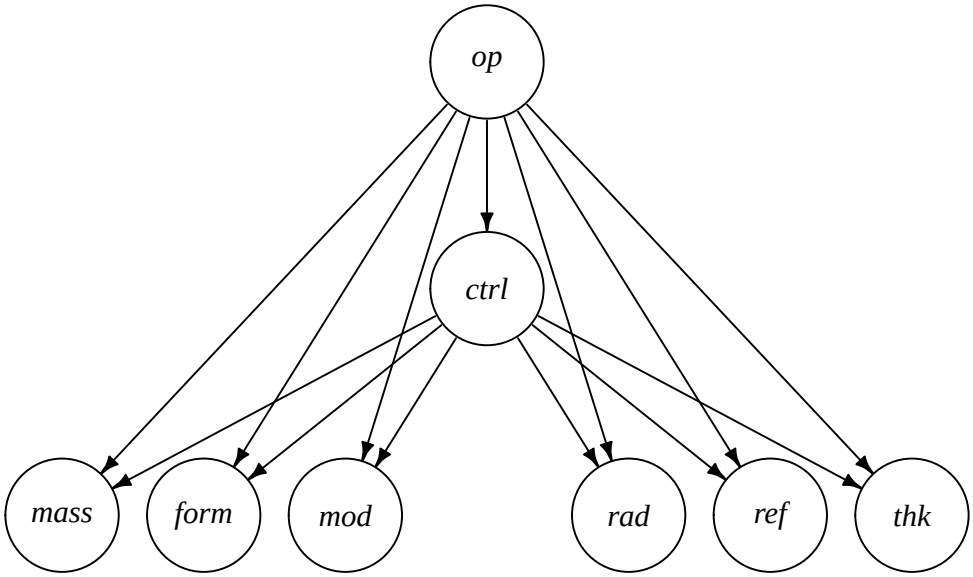

Figure 2.7: Bayesian network

$p(x_B)$ is non-negative and satisfies $\sum p(x_b|x^*_{parent(b)}) = 1$ for each parent $x^*_{parent(b)}$ of $x_{parent(b)}$. Thus, $p(x_b|x_{parent(b)})$ becomes the conditional probability distribution for x_b given $x_{parent(b)}$. The joint probability distributions of the Bayesian network (2.7) are calculated using 2.14.

$$p(x) = p(x_{op})p(x_{ctrl}|x_{op})\prod_i p(x_i|x_{op},x_{ctrl}) \tag{2.14}$$

where

$i =$ *mass, form, mod, rad, ref, thk* (Figure 2.7)

The joint probability distribution can also be explicitly expressed as a non-negative function, or *evidence potential* (2.15) [44]. The evidence potentional defines the structure of the *moral graph* [24], which is a type of undirected graph that is used by the *junction tree clustering algorithm* (1) to perform exact inference [63]. The moral graph of the Bayesian network

(Figure 2.7) is shown in Figure 2.8.

$$p(x) = \psi(x_{op}, x_{ctrl}) \prod_i \psi(x_i, x_{op}, x_{ctrl}) \tag{2.15}$$

where

$$\psi(x_{op}, x_{ctrl}) = \psi_{op \cup ctrl}(x_{op}, x_{ctrl}) = p(x_{op}|x_{ctrl}) = p(x_{op})p(x_{ctrl}|x_{op})$$

$$\psi(x_i, x_{op}, x_{ctrl}) = \psi_{i \cup op,ctrl}(x_i, x_{op}, x_{ctrl}) = p(x_i|x_{op}, x_{ctrl})$$

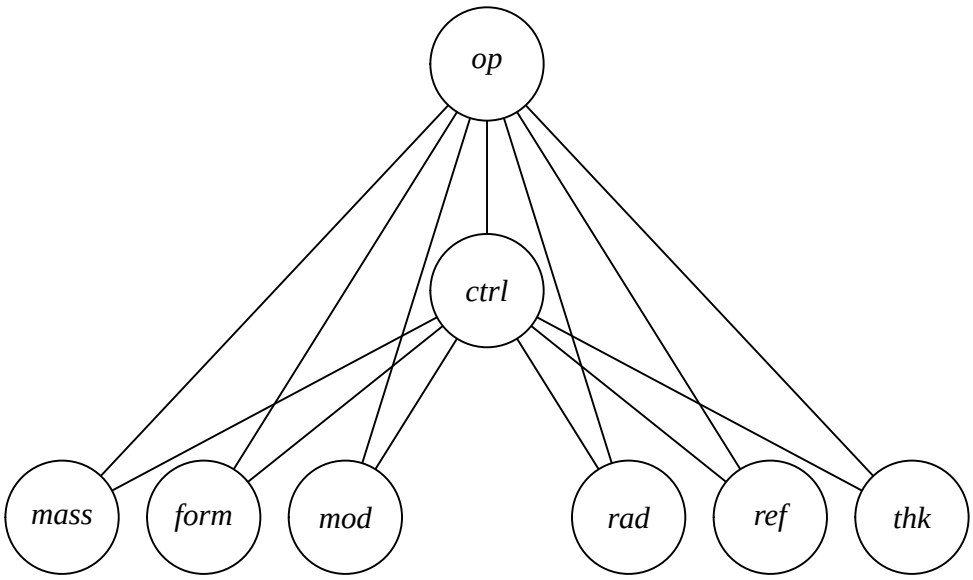

Figure 2.8: Moral graph

The Bayesian network is graphically decomposed using the *junction tree clustering algorithm* 1 [63] so that exact inference can be performed. The moral graph (Figure 2.8) is already triangulated [44], which means there are no cycles with more than three nodes, thus satisfying Algorithm 1. The next two steps in Algorithm 1 are to identify cliques (Table 2.1) and build a junction tree (Figure 2.9). The cliques (Table 2.1) were selected to minimize the number of arcs in the junction tree (Figure 2.9). By following Algorithm 1, the *clique potential representation* of $p(x)$ can be calculated using 2.16 [44].

Table 2.1: Cliques

i	cliques (C_i)	residuals (R_i)	separators (S_i)	parent cliques
1	*op*, *ctrl*, *mass*	*op*, *ctrl*, *mass*	$\varnothing$	
2	*op*, *ctrl*, *form*	*form*	*op*, *ctrl*	1
3	*op*, *ctrl*, *mod*	*mod*	*op*, *ctrl*	2
4	*op*, *ctrl*, *rad*	*rad*	*op*, *ctrl*	3
5	*op*, *ctrl*, *ref*	*ref*	*op*, *ctrl*	4
6	*op*, *ctrl*, *thk*	*thk*	*op*, *ctrl*	5

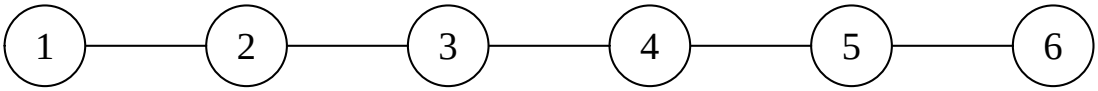

Figure 2.9: Junction tree

Moralize: Build the moral graph of the Bayesian network
Triangulate: Break every cycle spanning four or more nodes into subcycles of exactly three nodes by adding arcs to the moral graph, thus obtaining a *triangulated graph*.
Identify cliques: Identify cliques C_1, ..., C_k of the triangulated graph (i.e., subsets of nodes where each node is adjacent to all others in the subset).
Build junction tree: Build a tree in which each clique is a node, and adjacent cliques are connected by arcs. The tree must satisfy the *running intersection property*: if a node belongs to two cliques, C_i and C_j, it must also be included in all cliques in the unique path that connects C_i and C_j.
Parameterize: Use the parameters of the local distribution of $\mathcal{B}$ to calculate the parameter sets of the compound nodes of the junction tree.

Algorithm 1: Junction tree clustering algorithm [63]

$$p(x) = \prod_{i}^{6} \psi(C_i) \tag{2.16}$$

Starting with the last clique, C_6, where $C_6 = S_6 \cup R_6$, $S_6 \cap R_6 = \varnothing$, the factorization of 2.16 implies $R_6 \perp\!\!\!\perp (C_1 \cup ... \cup C_5)/S_6 | S_6$ [44]. Marginalizing over R_6 results in 2.17 [44].

$$p(C_1 \cup ... \cup C_5) = \prod_{i}^{5} \psi(C_i) \sum_{R_6} \psi(S_6, R_6) \tag{2.17}$$

Now, let $\psi(S_i) = \sum_{R_i} \psi(S_i, R_i)$. Combining this term with 2.17 results in 2.18 [44].

$$p(R_i|S_i) = \frac{\psi(S_i, R_i)}{\psi(S_i)} \tag{2.18}$$

Combining $\psi(S_i) = \sum_{R_i} \psi(S_i, R_i)$ with 2.17 and 2.18 results in 2.19 [44].

$$p(x) = p(C_1 \cup ... \cup C_5) p(R_6|S_6) = \left[\left(\prod_{i}^{5} \psi(C_i) \right) \psi(S_6) \right] \frac{\psi(C_6)}{\psi(S_6)} \tag{2.19}$$

The running intersection property ensures S_6 is contained in the neighbor of C_6. Combining 2.18 with 2.19 results in 2.20 [44].

$$p(x) = p(C_1) \prod_{i=2}^{6} p(R_i|S_i) \tag{2.20}$$

The resulting expression (2.20) is called a *set chain representation* [44]. The root potential now yields the joint marginal distribution of its nodes. Now, supposing C_1 is the parent of C_2, 2.20 can be rearranged as 2.21 [44].

$$p(x) = p(C_1) \frac{p(C_2)}{p(S_2)} \prod_{i=3}^{6} p(R_i|S_i) \tag{2.21}$$

When the clique C_2 has absorbed its message, $\psi(C_2) \leftarrow \psi(C_2) p(S_2)$, its potential is equal to the marginal distribution of its nodes. Propagating these messages until all leaves of the

junction tree are reached yields the *clique marginal representation* 2.22 [44].

$$p(x) = \frac{\prod_i^6 p(C_i)}{\prod_{j=2}^6 p(S_j)} \tag{2.22}$$

The marginal probabilities of each node in the junction tree can be calculated using (2.22) [44]. This allows a user to perform exact inference by providing and propagating evidence through 2.16-2.22, which leads to a clique marginal representation where the potentials become $\psi(C_i) = p(C_i|E^*)$ [44]. In this work, the *gRain* [34] software package is used to perform exact inference [35]. Example code and output are shown in Listing 2.1.

```
>
> library(gRain)
>
> grain.bn <- as.grain(bn) # Bayesian network conditional probability tables
>
> summary(grain.bn)
Independence network: Compiled: TRUE Propagated: FALSE
 Nodes : Named chr [1:8] 'op' 'ctrl' 'mass' 'form' 'mod' 'rad' 'ref' 'thk'
 - attr(*, 'names')= chr [1:8] 'op' 'ctrl' 'mass' 'form' ...
 Number of cliques:                  6
 Maximal clique size:                3
 Maximal state space in cliques: 168042
>
> grain.bn$rip
cliques
  1 : op ctrl mass
  2 : op ctrl form
  3 : op ctrl mod
  4 : op ctrl rad
  5 : op ctrl ref
  6 : op ctrl thk
separators
  1 :
  2 : op ctrl
  3 : op ctrl
  4 : op ctrl
  5 : op ctrl
  6 : op ctrl
parents
  1 : 0
  2 : 1
  3 : 2
  4 : 3
  5 : 4
  6 : 5
>
> finding <- setFinding(grain.bn, nodes = 'mass', states = '500')
>
> pFinding(finding)
[1] 2.611831e-08
>
```

Listing 2.1: Output from using the *gRain* [34] software package to calculate a marginal probability where mass = 500 grams

The *bnlearn* [62] software package is used to randomly sample Bayesian network parameters using Algorithm 2 [40, 63]. Example code and output are shown in Listing 2.2.

Let $X_1, ..., X_n$ be a topological ordering of X for Bayesian network $\mathcal{B}$ over X
for $i = 1, ..., n$
    $u_i \leftarrow x\langle Pa_{X_i} \rangle$ // assigned to Pa_{X_i} in $x_1, ..., x_{i-1}$
    sample x_i from $P(X_i|u_i)$
return $x_1, ..., x_n$

Algorithm 2: Bayesian network forward sampling algorithm [40]

```
>
> library(bnlearn)
> library(caret)
> library(parallel)
>
> # set variables
> sample.size <- 10
> cluster <- makeCluster((detectCores() / 4), type = 'SOCK')
>
> # sample conditional probability tables
> bn.data <- cpdist(
+     bn,
+     nodes = c('mass', 'form', 'mod', 'rad', 'ref', 'thk'),
+     evidence = TRUE,
+     batch = sample.size,
+     cluster = cluster,
+     n = sample.size) %>% na.omit()
>
> stopCluster(cluster)
>
> print(bn.data)
   mass  form  mod    rad   ref   thk
1    30 alpha none  5.715   ch2 1.905
2     7 alpha none  13.97 ss304 0.635
3    48 alpha none   6.35   ch2 0.635
4    80 alpha none  1.905   ch2 0.635
5    57 alpha none   6.35 ss304 1.905
6    61  puo2 none   2.54 ss304  1.27
7    46  puo2 none  8.255    du  2.54
8    22 alpha none  10.16   mgo 1.905
9    33 alpha none  0.635   ch2 0.635
10    9 alpha none 17.145   ch2  1.27
>
```

Listing 2.2: Output from using the *bnlearn* [62] software package to randomly sample Bayesian network parameters [63]

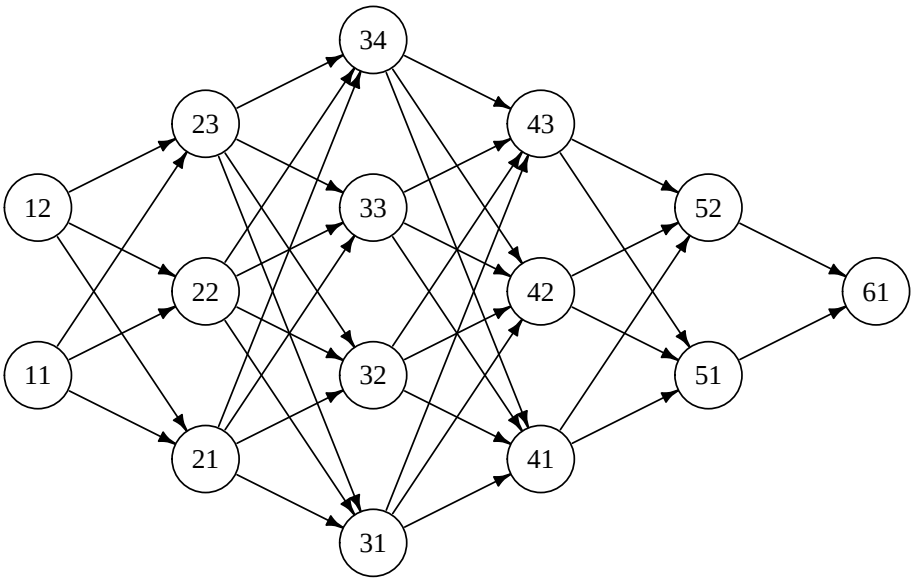

Figure 2.10: Neural network with four hidden layers

2.3 Neural Networks

A *neural network* is a connected system of nodes and directed arcs, similar to a Bayesian network. The differences between neural networks and Bayesian networks are that each node in a neural network consists of an activation function, a loss function, and an optimization algorithm, and each arc is assigned a weight based on its relative importance.

A *deep feedforward neural network* is a neural network with an input layer, two or more hidden layers, and an output layer (Figure 2.10). A deep feedforward neural network is comprised of many different functions, which approximate θ in the function $f(x;\theta)$ and form complex chains that map x to $f(x;\theta)$. Each chain is a function of the form $f_i(f_{i-1}(f_{i-2}(...(f_{i=1}(x;\theta)))))$ where the maximum value of i represents the depth of the network. The output of the first hidden layer of the neural network, shown in Figure 2.10, is

calculated using 2.23.

$$\begin{bmatrix} x_{21} \\ x_{22} \\ x_{23} \end{bmatrix} = g\left(\begin{bmatrix} W_{10\to21}x_{10} + W_{11\to21}x_{11} + W_{12\to21}x_{12} \\ W_{10\to22}x_{10} + W_{11\to22}x_{11} + W_{12\to22}x_{12} \\ W_{10\to23}x_{10} + W_{11\to23}x_{11} + W_{12\to23}x_{12} \end{bmatrix} + b_1 \right) \tag{2.23}$$

where

x_{ij} = output of neuron ij

$W_{ij\to(i+1)k}$ = weight from the output of neuron ij to the input of neuron $(i+1)k$

b_i = bias

$g(x)$ = activation function

Similarly, the output of each hidden layer, shown in Figure 2.10, is calculated using 2.24.

$$\begin{bmatrix} x_{31} \\ x_{32} \\ x_{33} \\ x_{34} \end{bmatrix} = g\left(\begin{bmatrix} W_{20\to31}x_{20} + W_{21\to31}x_{21} + W_{22\to31}x_{22} + W_{23\to31}x_{23} \\ W_{20\to32}x_{20} + W_{21\to32}x_{21} + W_{22\to32}x_{22} + W_{23\to32}x_{23} \\ W_{20\to33}x_{20} + W_{21\to33}x_{21} + W_{22\to33}x_{22} + W_{23\to33}x_{23} \\ W_{20\to34}x_{20} + W_{21\to34}x_{21} + W_{22\to34}x_{22} + W_{23\to34}x_{23} \end{bmatrix} + b_2 \right)$$

$$\begin{bmatrix} x_{41} \\ x_{42} \\ x_{43} \end{bmatrix} = g\left(\begin{bmatrix} W_{30\to41}x_{30} + W_{31\to41}x_{31} + W_{32\to41}x_{32} + W_{33\to41}x_{33} + W_{34\to41}x_{34} \\ W_{30\to42}x_{30} + W_{31\to42}x_{31} + W_{32\to42}x_{32} + W_{33\to42}x_{33} + W_{34\to42}x_{34} \\ W_{30\to43}x_{30} + W_{31\to43}x_{31} + W_{32\to43}x_{32} + W_{33\to43}x_{33} + W_{34\to43}x_{34} \end{bmatrix} + b_3 \right)$$

$$\begin{bmatrix} x_{51} \\ x_{52} \end{bmatrix} = g\left(\begin{bmatrix} W_{40\to51}x_{40} + W_{41\to51}x_{41} + W_{42\to51}x_{42} + W_{43\to51}x_{43} \\ W_{40\to52}x_{40} + W_{41\to52}x_{41} + W_{42\to52}x_{42} + W_{43\to52}x_{43} \end{bmatrix} + b_4 \right) \tag{2.24}$$

The output of the last layer, shown in Figure 2.10, is calculated using 2.25. Unlike 2.23 and 2.24, 2.25 uses a linear activation function, which is the same as having no activation function. This allows the neural network to generate output that is positive or negative

(Figure 2.11) [30].

$$\left[x_{61}\right] = \left[W_{50\to 61}x_{50} + W_{51\to 61}x_{51} + W_{52\to 61}x_{52}\right] + b_5 \tag{2.25}$$

Following 2.23-2.25 results in one forward pass of input through the neural network. The vector output from 2.25 is then passed to a loss function, which for regression problems, is typically the MSE, shown in 2.26 [19, 30].

$$\text{MSE} = \frac{1}{n}\sum_{j=1}^{n}(x_{observed_j} - x_{predicted_j})^2 \tag{2.26}$$

where

$n =$ total number of variables that are observed or predicted

$x =$ vector of observations or predictions

In this work, each neural network layer is densely (or fully) connected, and each hidden neuron uses a rectified linear unit activation function, which is a piecewise function that returns x if $x > 0$ and 0 if $x \leq 0$ (Figure 2.11).

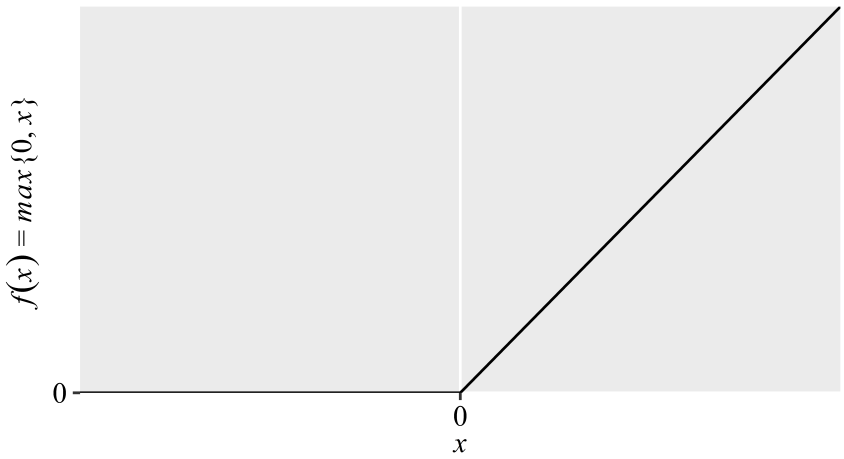

Figure 2.11: Rectified linear unit (ReLU) activation function

A simplified equation for a deep feedforward neural network with four hidden layers is shown in 2.27 [30].

$$f(x;W,b) = W_5^\top \prod_{i=1}^{4} \Big(max\{0, W_i^\top x_i + b_i\} \Big) + b_5 \tag{2.27}$$

where

$W_5^\top + b_5 =$ row vector mapping the last hidden layer to the output layer

$max\{0, W_i^\top x_i + b_i\} =$ row vector mapping adjacent input and hidden layers

2.26 is the general form of the MSE equation. Substituting $x_{predicted_i}$ for the right side of 2.27 yields 2.28.

$$\text{MSE} = \frac{1}{n} \sum_{j=1}^{n} \left(y_j - \left[W_5^\top \prod_{i=1}^{4} \Big(max\{0, W_i^\top x_i + b_i\} \Big) + b_5 \right]_j \right)^2 \tag{2.28}$$

The output from the MSE loss function (2.28) is passed to the optimization algorithm, which updates parameters (θ) by implementing the backpropagation algorithm [30]. The backpropagation algorithm is the backbone of a neural network and applies the chain rule to compute gradients (i.e., the vector of partial derivatives with respect to θ evaluated at time step i 3) and update parameters (θ) [30]. These parameters (θ) are also referred to as the weights (W) and biases (b) of the neural network in 2.23-2.27. In this work, the Adamax optimization algorithm (Algorithm 3) is used to update parameters (θ) [38]. Adamax is a variant of Adam (Algorithm 4) [38] based on the exponentially weighted infinity norm, which takes the place of the bias-corrected first and second moment estimates [38].

Require: α = step size
Require: $\beta_1, \beta_2 \in [0,1)$ = exponential decay rates for the moment estimates
Require: $f(x;\theta)$ = loss function with parameters θ
Require: θ_0 = initial parameter vector
$m_0 \leftarrow 0$ (initialize first moment vector)
$u_0 \leftarrow 0$ (initialize exponentially weighted infinity norm)
$i \leftarrow 0$ (initialize first iteration)
while θ_i not converged **do**
$i \leftarrow i+1$
$g_i \leftarrow \bigtriangledown_\theta f_i(\theta_{i-1})$ (get gradients from loss function at time step i)
$m_i \leftarrow \beta_1 \cdot m_{i-1} + (1-\beta_1) \cdot g_i$ (update biased first moment estimate)
$u_i \leftarrow max(\beta_2 \cdot u_{t-1}, |g_i|)$ (update exponentially weighted infinity norm)
$\theta_i \leftarrow \theta_{t-1} - (\alpha/(1-\beta_1^i)) \cdot m_i/u_i$ (update parameters)
return θ_i

Algorithm 3: Adamax optimization algorithm [38]

Require: α = step size
Require: $\beta_1, \beta_2 \in [0,1)$ = exponential decay rates for the moment estimates
Require: $f(x;\theta)$ = loss function with parameters θ
Require: θ_0 = initial parameter vector
$m_0 \leftarrow 0$ (initialize first moment vector)
$v_0 \leftarrow 0$ (initialize second moment vector)
$i \leftarrow 0$ (initialize first iteration)
while θ_i not converged **do**
$i \leftarrow i+1$
$g_i \leftarrow \bigtriangledown_\theta f_i(\theta_{i-1})$ (get gradients from loss function at time step i)
$m_i \leftarrow \beta_1 \cdot m_{i-1} + (1-\beta_1) \cdot g_i$ (update biased first moment estimate)
$v_i \leftarrow \beta_2 \cdot v_{i-1} + (1-\beta_2) \cdot g_i^2$ (update biased second moment estimate)
$\hat{m}_i \leftarrow m_i/(1-\beta_1^i)$ (calculate bias-corrected first moment estimate)
$\hat{v}_i \leftarrow v_i/(1-\beta_2^i)$ (calculate bias-corrected second moment estimate)
$\theta_i \leftarrow \theta_{i-1} - \alpha \cdot \hat{m}_i/(\sqrt{\hat{v}_i} + \varepsilon)$ (update parameters)
return θ_i

Algorithm 4: Adam optimization algorithm [38]

Chapter 3: Bayesian Network

The Bayesian network, shown in Figure 3.1, is comprised of 8 nodes and 13 arcs, which represent fissionable material operations (*op*), criticality controls (*ctrl*), and parameters that affect nuclear criticality [50].

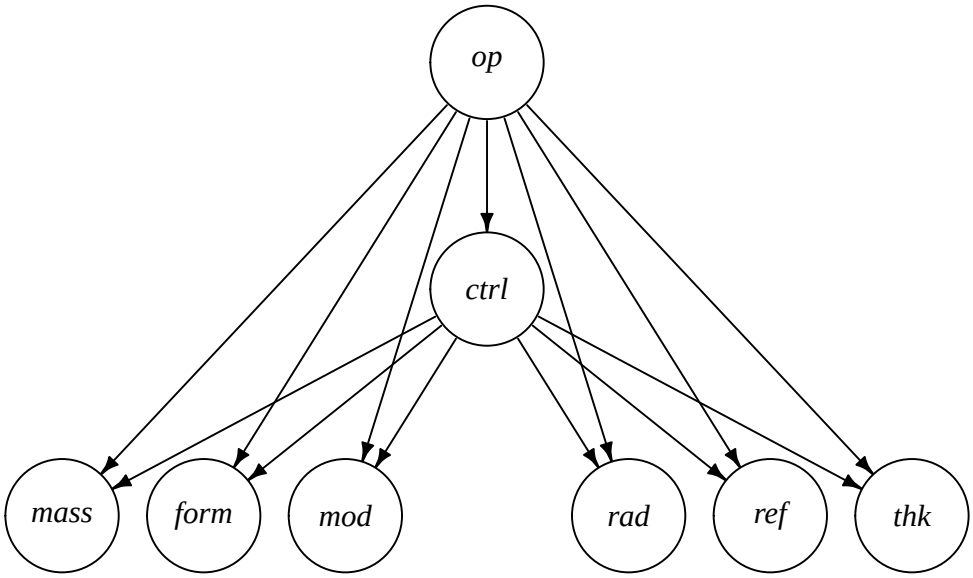

Figure 3.1: Bayesian network

The *bnlearn* [62] and *igraph* [22] software packages were used to build the Bayesian network and randomly sample parameters using the forward sampling algorithm (Algorithm 2). Fissionable material operations are divided into six categories (Table 3.1), based on safety plans and work control documents that describe each operation, as well as the criticality controls that apply to them. Criticality controls in Building 332 consist of standardized sets of controls that are analyzed and approved for individual fissionable material operations. Each set of controls has a single letter designation and consists of administrative limits on plutonium mass, form, moderation, and reflection (Table 3.3).

Most fissionable material operations have two to five sets of criticality controls. However, only one set of criticality controls can be active at each workstation. For example, a fissile

Table 3.1: Fissionable material operations in Building 332

operation	description
large sample	operations with $\geq$ 65 g of plutonium
machining	operations with a drill press, grinder, lathe, mill, saw, or other cutting equipment
metallurgy	operations with a casting furnace or other heating equipment
small sample	operations with $<$ 65 g of plutonium
solution	operations with liquids
waste	operations with radioactive waste

Table 3.2: Conditional probability table for fissionable material operations in Building 332

large sample	machining	metallurgy	small sample	solution	waste
0.3421	0.0789	0.1316	0.3158	0.1053	0.0263

Table 3.3: Criticality controls in Building 332

condition	mass	form	moderation	reflection
A	$\leq$ 65 g		no D_2O	
B	$\leq$ 220 g		H/X $\leq$ H_2O, $\leq$ 4 L	no Be/C/D_2O, $\leq$ 2 in
C	$\leq$ 1200 g		$\leq$ 2.5 L	none
D	$\leq$ 2500 g		$\leq$ 1 L	$\leq$ 2 L, $\leq$ 0.25 in
E	$\leq$ 2500 g		none	$\leq$ 0.25 in
M	$\leq$ 4000 g	$\leq$ 500 g fines	$\leq$ 1 in, $\leq$ 4 L	$\leq$ 0.3 in
P	$\leq$ 300 g		no liquids	$\leq$ 3.5 kg Be/C

material handler can use *Condition A* at one workstation, and then switch to *Condition E* if the operation requires them to move > 65 grams of plutonium into the workstation. To facilitate this move, the handler is required to ensure that the more restrictive moderator and reflector limits of *Condition E* are met prior to switching from *Condition A*. The conditional probability table for each set of criticality controls (*ctrl*), shown in Table 3.4, is derived from walkthrough forms, which nuclear engineers use to document active criticality controls at each workstation in the facility.

The parameters that affect nuclear criticality in Building 332 are shown in Table 3.5. These parameters link fissionable material operations (Table 3.1) and criticality controls (Table 3.4) to the bottom-tier Bayesian network parameters, shown in Figure 3.1. Spherical geometry and surrogate materials (Table 3.5) were selected to conservatively bound the physics of a process criticality accident. Spherical geometry is used because a sphere has the lowest surface area-to-volume ratio of any object, which reduces neutron leakage and therefore increases k_{eff} [39]. Water is used instead of organic and inorganic liquids because it provides superior moderation and reflection, which also increases k_{eff} [39]. Polyethylene is similarly used instead of rubber, filter paper, foam, thermocouple insulation, and plastic. Magnesium oxide (MgO) is included in Table 3.5 because it is commonly used in crucibles. Sepiolite is included because it is used to downblend plutonium oxide and absorb liquids in waste. Other materials, such as aluminum, beryllium, depleted uranium (^{238}U), graphite (C), and 304 stainless steel are included because they are used in experiments or are integral to gloveboxes and process equipment.

3.1 Fitting Probability Distributions

The probability tables for operations (*op*), controls (*ctrl*), *form*, *moderator*, and *reflector* were populated directly from unclassified computer records and waste parcel cards. The probability tables for *mass*, *radius*, and *reflector thickness*, however, are based on truncated probability distributions (3.1-3.4) that were fit to incomplete data using the maximum

Table 3.4: Conditional probability table for criticality controls in Building 332

condition	large sample	machining	metallurgy	small sample	solution	waste
A	0.5714	0.1538	0.6154	0.9375	1	0.5
B	0.0286	0	0.0769	0.0625	0	0
C	0.0286	0	0	0	0	0
D	0.1714	0	0	0	0	0
E	0.0857	0	0.3077	0	0	0
M	0.1143	0.8462	0	0	0	0
P	0	0	0	0	0	0.5

Table 3.5: Parameters that affect nuclear criticality in Building 332

parameter	description
mass	grams of plutonium (95% ^{239}Pu, 5% ^{240}Pu by weight)
form	plutonium in the form of alpha-phase metal or oxide (PuO_2)
moderator (*mod*)	MgO, CH_2, sepiolite, H_2O, or none
radius (*rad*)	radius of sphere (in or cm)
reflector (*ref*)	Al, Be, ^{238}U, C, Pb, MgO, CH_2, SS304, H_2O, or none
reflector thickness (*thk*)	reflector thickness (in or cm)
shape	sphere
volume (*vol*)	volume of sphere (cm^3)
concentration (*conc*)	concentration of plutonium in solution (g/cm^3)

likelihood estimation method [23]. Truncated gamma and normal distributions were selected because they are commonly used distributions that fit the data reasonably well (Table 3.6). Truncated log-normal and Weibull distributions were selected because they are heavy right-tailed distributions, which increases the probability of generating random samples that result in a criticality accident. The probability density functions of these distributions are shown in 3.1-3.4 [26].

$$\text{gamma distribution} \quad f(x;\alpha,\beta) = \frac{\beta^{\alpha} x^{\alpha-1} e^{-\beta x}}{\Gamma(\alpha)} \quad x,\alpha,\beta > 0 \tag{3.1}$$

$$\text{normal distribution} \quad f(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{(x-\mu)^2}{2\sigma^2}} \tag{3.2}$$

$$\text{log-normal distribution} \quad f(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma x} e^{\frac{(\ln x-\mu)^2}{2\sigma^2}} \tag{3.3}$$

$$\text{Weibull distribution} \quad f(x;\lambda,k) = \begin{cases} \frac{k}{\lambda}(\frac{x}{\lambda})^{k-1} e^{-(\frac{x}{\lambda})^k} & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{3.4}$$

The maximum likelihood estimators are derived from the likelihood function, shown in 3.5, where $X_1,...,X_n$ are independent and indentically distributed random variables from a population with a probability density function $f(x|\theta_1,...,\theta_k$.

$$L(\theta|x) = L(\theta_1,...,\theta_k|x_1,...,x_n) = \prod_{i=1}^{n} f(x_i|\theta_1,...,\theta_k) \tag{3.5}$$

Once the probability distributions were fit to the data, they were trucated by selecting a conservatively large upper limit for each variable, removing values beyond the limits, and then normalizing the remaining values so that each row in the conditional probability table sums to one. Kolmogorov-Smirnov [47], Cramér-von Mises, [20], and Anderson-Darling [9]

tests were also performed to determine the goodness of fit for each probability distribution. The general formulas for these test statistics are shown in 3.6-3.8 [9, 20, 47], where F_n is the empirical distribution function and F_0 is the fitted cumulative distribution function. The test statistics were calculated using the *fitdistrplus* [25] software package and compared to the critical values in R.B. D'Agostino and M.A. Stephens [23].

$$\text{Kolmogorov-Smirnov test statistic} \quad D = \sup_x |F_n(x) - F_0(x)| \tag{3.6}$$

$$\text{Cramér-von Mises test statistic} \quad W_n^2 = n \int_{-\infty}^{\infty} (F_n(x) - F_0(x))^2 dF_0(x) \tag{3.7}$$

$$\text{Anderson-Darling test statistic} \quad A_n^2 = n \int_{-\infty}^{\infty} \frac{(F_n(x) - F_0(x))^2}{F_0(x)(1 - F_0(x))} \tag{3.8}$$

An example set of truncated gamma distributions and quantile-quantile (Q-Q) plots for the 65-gram *Condition A* mass limit are shown in Figures 3.2 and 3.3. Histograms of Kolmogorov-Smirnov [47], Cramér-von Mises, [20], and Anderson-Darling [9] test statistics are shown in Figure 3.4 and Table 3.6. The critical values of these test statistics (Table 3.6) correspond to a significance level of $\alpha = 0.05$ with $n = 500$ samples [23].

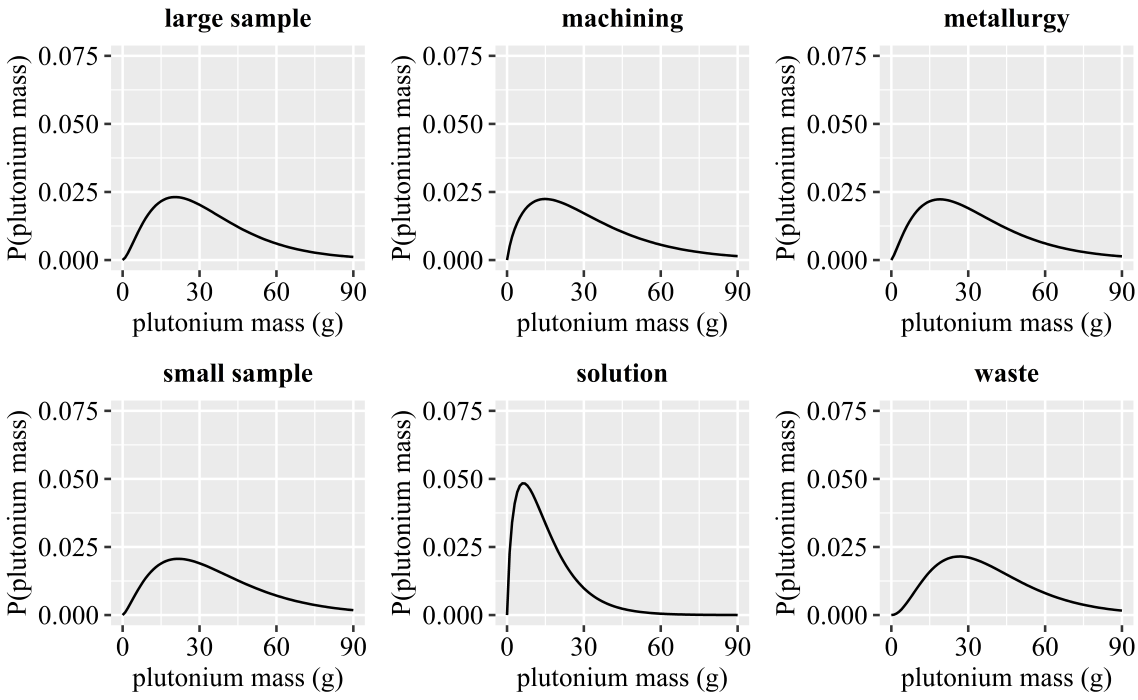

Figure 3.2: Truncated gamma distributions for 65-gram mass limit

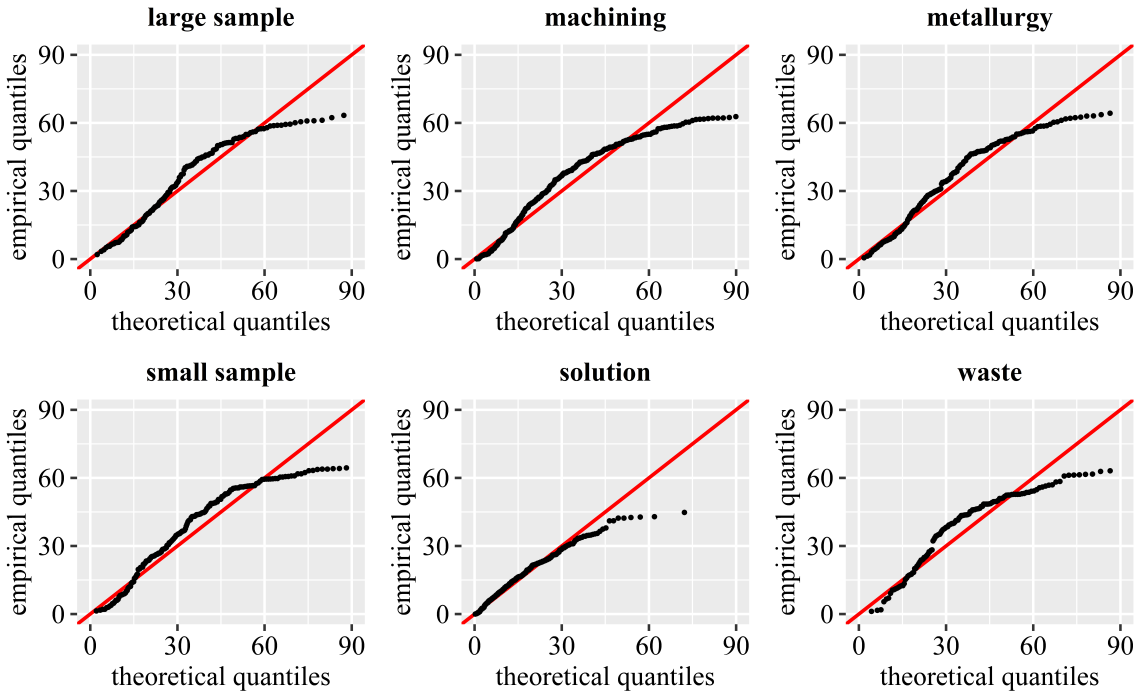

Figure 3.3: Q-Q plots of truncated gamma distributions for 65-gram mass limit

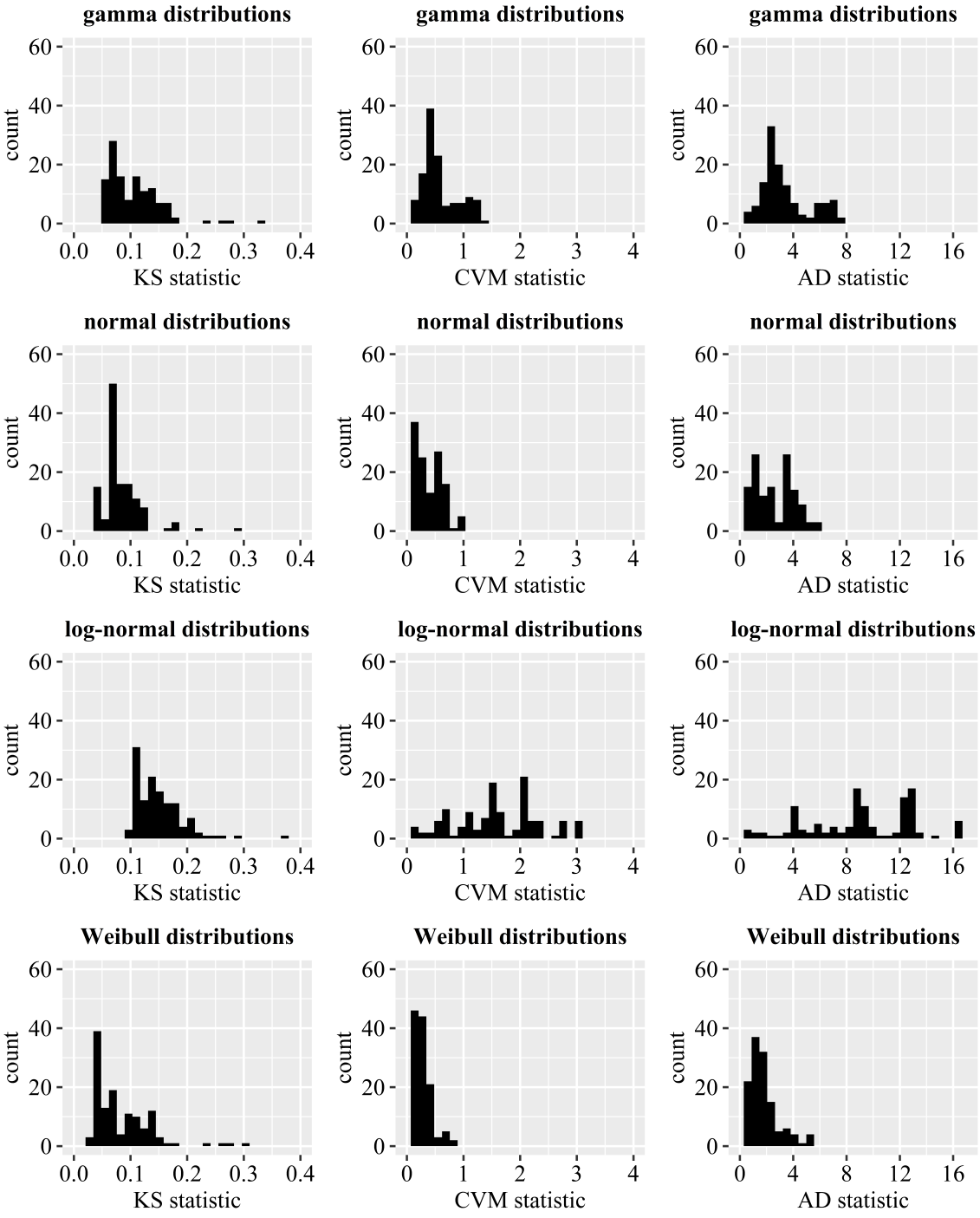

Figure 3.4: Histograms of test statistics

The results shown in Table 3.6 indicate that the test statistics for the truncated gamma, normal, and Weibull distributions fit the data slightly better than the truncated log-normal distributions—although this is by no means a definitive assessment of each fit. The purpose

Table 3.6: Test statistics

	KS test statistic		CVM test statistic		AD test statistic	
distribution	mean	range	mean	range	mean	range
gamma	0.105	0.052-0.305	0.626	0.056-1.339	3.704	0.333-7.909
normal	0.083	0.039-0.278	0.411	0.038-0.883	2.810	0.301-5.787
log-normal	0.148	0.089-0.313	1.663	0.041-2.796	9.870	0.271-16.348
Weibull	0.081	0.029-0.307	0.305	0.020-0.802	2.126	0.158-5.581
critical value	0.061		0.220		0.751-0.795	

of calculating these test statistics is to document the goodness of fit, which is commonly used in conjunction with graphical checks of Q-Q plots (Figure 3.3) to validate risk models in the nuclear industry.

3.2 Generating Random Samples

The Bayesian network forward sampling algorithm (2) is used to generate random samples, which are passed to the neural network metamodel.

Chapter 4: Neural Network Metamodel

The neural network metamodel is comprised of 10 neural neural networks that were built using the *Keras* [18] and *TensorFlow* [7] software packages. The main benefit of using a neural network metamodel is that it can generate predictions in less time than it takes MCNP to perform the same number of calculations. This is especially beneficial if millions (or even billions) of predictions are needed to estimate process criticality accident risk and reduce the variance (v) of the risk estimate to an acceptable (low) value (4.1).

$$v = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{4.1}$$

where

v = variance of the risk estimate

n = number of samples

x_i = risk estimate (e.g., 1e-07 accidents/year)

$\bar{x}$ = mean risk estimate

Each of the 10 neural networks has the same structure, which consists of one input layer, six hidden layers, and one output layer. The input layer has 28 neurons that correspond to the Bayesian network parameters, shown in Table 3.5, and the output layer has one neuron that corresponds to k_{eff}. The six hidden layers have the following number of neurons in each layer: 8192-256-256-256-256-16. The number of hidden layers and the number of neurons in each hidden layer were determined using a manual training process, described in the next sections.

Table 4.1: Training parameters based on 1,542,792 MCNP output files

parameter	description
mass	0-4 kg of Pu in 25 g increments
form	Pu (19.86 g/cm^3) or PuO_2 (11.5 g/cm^3)
moderator (*mod*)	MgO, CH_2, sepiolite, H_2O, or none
radius (*rad*)	0-18 inches in 1-inch increments
reflector (*ref*)	Al, Be, ^{238}U, C, Pb, MgO, CH_2, SS304, H_2O, or none
reflector thickness (*thk*)	0-6 inches in 1-inch increments
shape	sphere
volume (*vol*)	volume of sphere (cm^3)
concentration (*conc*)	concentration of Pu in solution (g/cm^3)

4.1 Preparing Data

Each neural network was trained using a 64-16-20 data split, based on the parameters shown in Table 4.1. This was done by setting aside 20% of the 1,542,792 output files for testing, and splitting the other 80% into two groups of 64% for training and 16% for cross-validation. This is also sometimes called an 80-20 data split because 64% is 80% of 80% of the 1,542,792 output files, and 16% is 20% of 80%.

A much smaller set of output files (Table 4.2) was initially used because each neural network took several hours to train, and it wasn't feasible to spend hundreds of hours training a neural network metamodel. Later on, it was determined that the accuracy of neural networks could be improved by training for fewer epochs (i.e., fewer passes of training data) on a larger dataset. The increments and upper limits of the discrete parameters, shown in Tables 4.1 and 4.2, were selected based on the rationale discussed previously, as well as the total amount of fissionable material that is allowed in Building 332, a Security Category III nuclear facility (6). The parameters, shown in Tables 4.1 and 4.2, and the *Grid* (**??**)

Table 4.2: Training parameters based on 202,786 MCNP output files

parameter	description
mass	0-2 kg of Pu in 50 g increments
form	Pu (19.86 g/cm^3) or PuO_2 (11.5 g/cm^3)
moderator (*mod*)	MgO, CH_2, sepiolite, H_2O, or none
radius (*rad*)	0-18 inches in 1-inch increments
reflector (*ref*)	Al, Be, ^{238}U, C, Pb, MgO, CH_2, SS304, H_2O, or none
reflector thickness (*thk*)	0-3 inches in 1-inch increments
shape	sphere
volume (*vol*)	volume of sphere (cm^3)
concentration (*conc*)	concentration of Pu in solution (g/cm^3)

and *Build* (**??**) functions were used to build and run the MCNP input decks on Quartz, a supercomputer at Lawrence Livermore National Laboratory (Figure 4.1). Once the MCNP input decks were run, data from the output files was extracted, processed, and written to comma-separated values (CSV) and RData files, using the *Tabulate* (**??**) and *Split* (**??**) functions.

The training data was scaled by calculating the column-wise mean and standard deviation of each discrete variable, subtracting each mean from its corresponding column, and then dividing each column by its corresponding standard deviation. These last two steps were repeated for the test data using the means and standard deviations from the training data. The training and test data were then one-hot-encoded using the *dummyVars* function in the *caret* [41] software package, which converts categorical variables into multi-column dummy variables, where membership within each sub-category is 0 or 1. The purpose of scaling and one-hot-encoding data was to improve the initial parameter (θ) updates of the neural network [19], and minimize bias towards individual training parameters (Table 4.2).

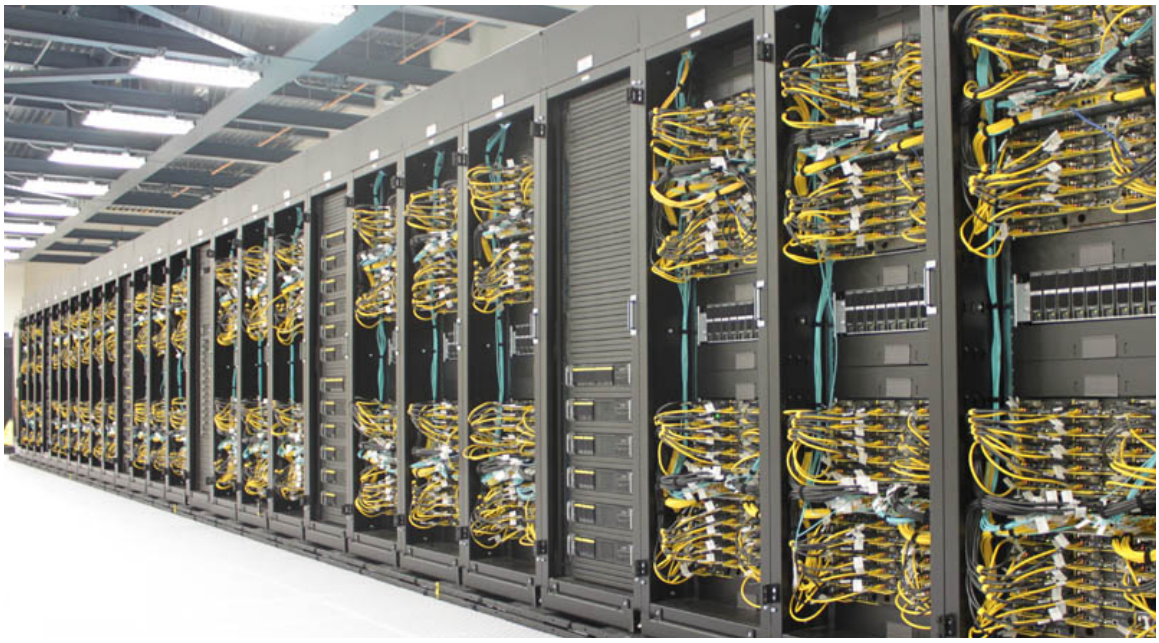

Figure 4.1: Quartz supercomputer at Lawrence Livermore National Laboratory

4.2 Training Neural Networks

Each neural network was initially trained for 1,500 epochs using a "mini-batch" training approach [30], which consists of splitting the training data into smaller batches of 8,192, computing gradients using MSE (2.26), and updating parameters (θ) after each batch. Then, each neural network was trained for an additional 150 epochs, and the model was saved as a separate HDF5 file after each epoch. Lastly, each neural network was evaluated by comparing the sum of mean absolute errors (4.2) on training and cross-validation data for the last 150 epochs (Figures 4.2 and 4.3). The neural network with the lowest sum of mean absolute errors was selected as the baseline for the neural network metamodel. To correct for possible bias towards cross-validation data (i.e., the smaller dataset), the weighted averages were also compared (Figures 4.4 and 4.5). Although the sums and weighted averages differ slightly, the selected neural network has the lowest sum and weighted average of mean absolute errors on training and cross-validation data, so there is no change in outcome

(Figures 4.2-4.5).

$$\text{mean absolute error (MAE)} = \frac{1}{n}\sum_{i=1}^{n}|x_{observed_i} - x_{predicted_i}| \tag{4.2}$$

4.2.1 Applying a Sum of Squared Errors Loss Function

The MSE loss function (2.26) was initially used to train each neural network. The reason it was used is because nearly all journal articles, preprints After reading more about loss functions in *The Theory of Point Estimation* [45] and other books [12, 26, 42], and then failing to find any relevant journal articles or preprints on why MSE is Later on, the sum of squared errors loss function (4.3) was tested, and it was found to outperform the more commonly used MSE loss function [19, 30]. This section discusses why the sum of squared errors loss function (4.3) is better, and provides empirical evidence and code to support this claim.

$$\text{sum of squared errors (SSE)} = \sum_{i=1}^{n}(x_{observed_i} - x_{predicted_i})^2 \tag{4.3}$$

When a neural network is trained in batches, the number of samples in the last batch is calculated using n mod b, where n is the number of training samples and b is batch size. Typically, this calculation results in a smaller last batch, except in cases where the number of training samples is evenly divisible by batch size. A smaller last batch isn't a problem by itself, but when the MSE loss function (2.26) is used, the computed gradients are divided by the number of samples in each batch. This causes the samples in the last batch to be weighted more heavily than all other samples, which causes parameter (θ) updates to become more sensitive to the computed gradients of the last batch.

The effect the last batch has on training stability and overall performance is highly dependent on the optimization algorithm that is used. A portion of the Adamax optimization algorithm (Algorithm 3) is reproduced in Algorithm 5, as it is implemented in the neural network metamodel (**??**, **??**).

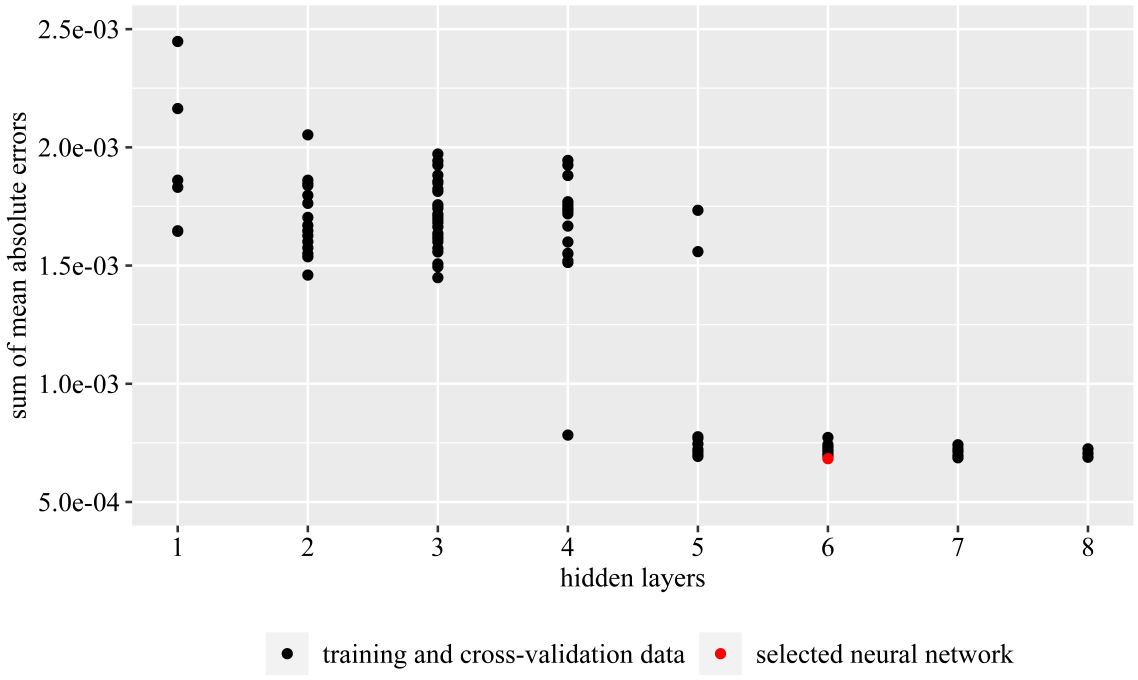

Figure 4.2: Sum of mean absolute errors on training and cross-validation data

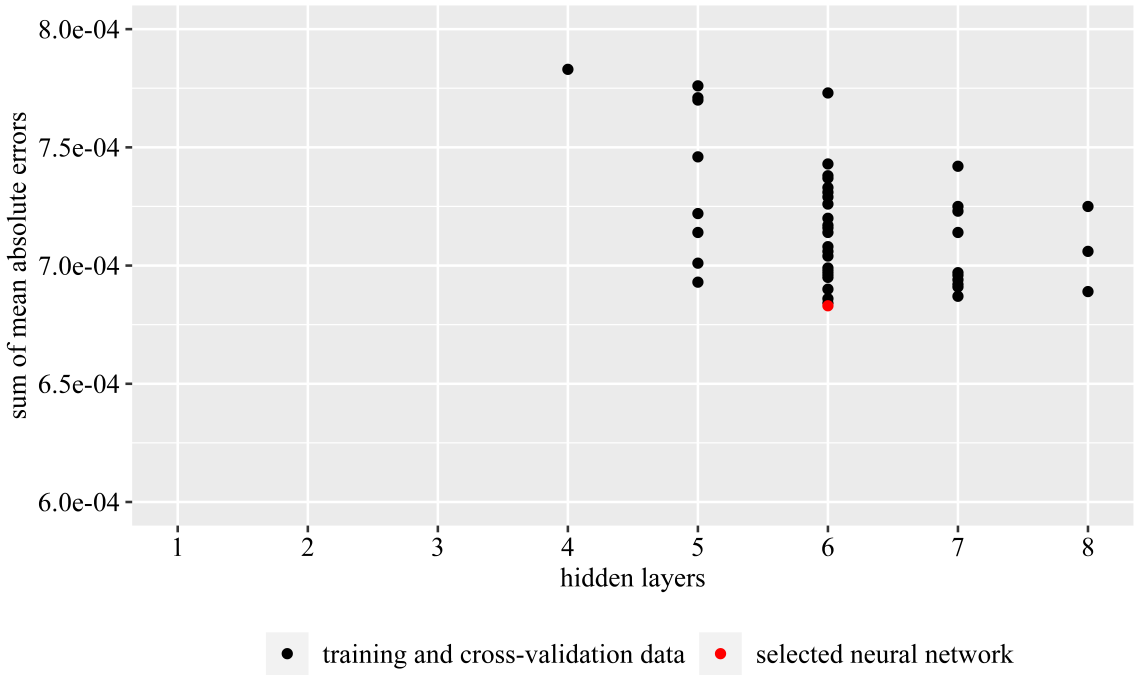

Figure 4.3: Sum of mean absolute errors on training and cross-validation data

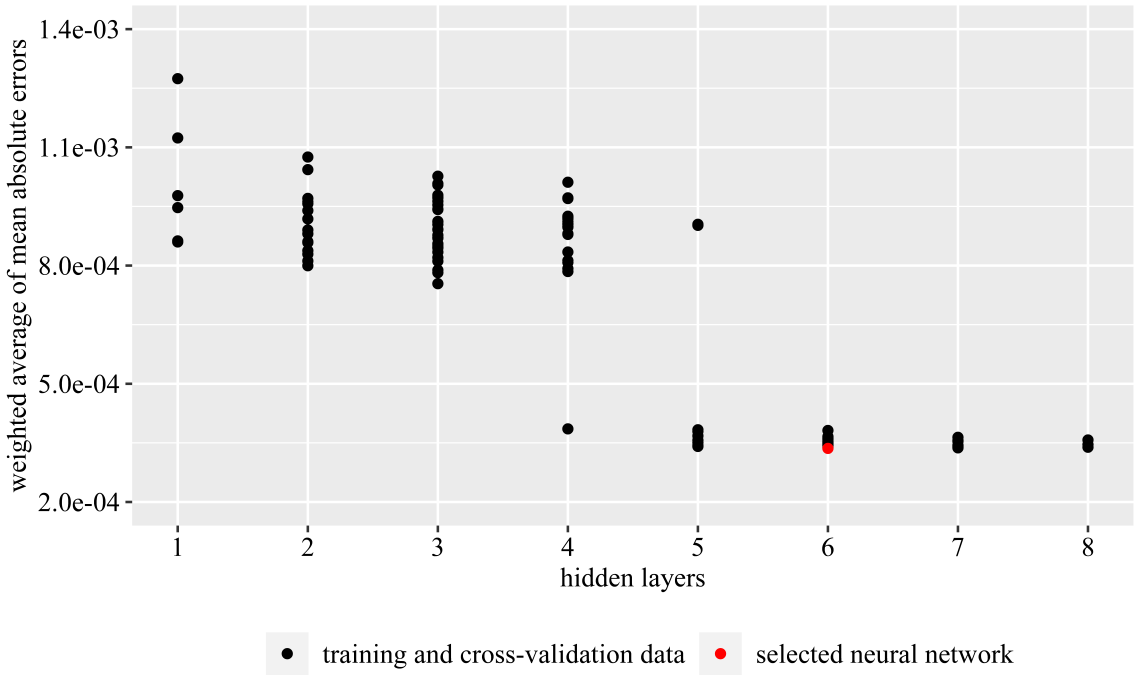

Figure 4.4: Weighted average of mean absolute errors on training and cross-validation data

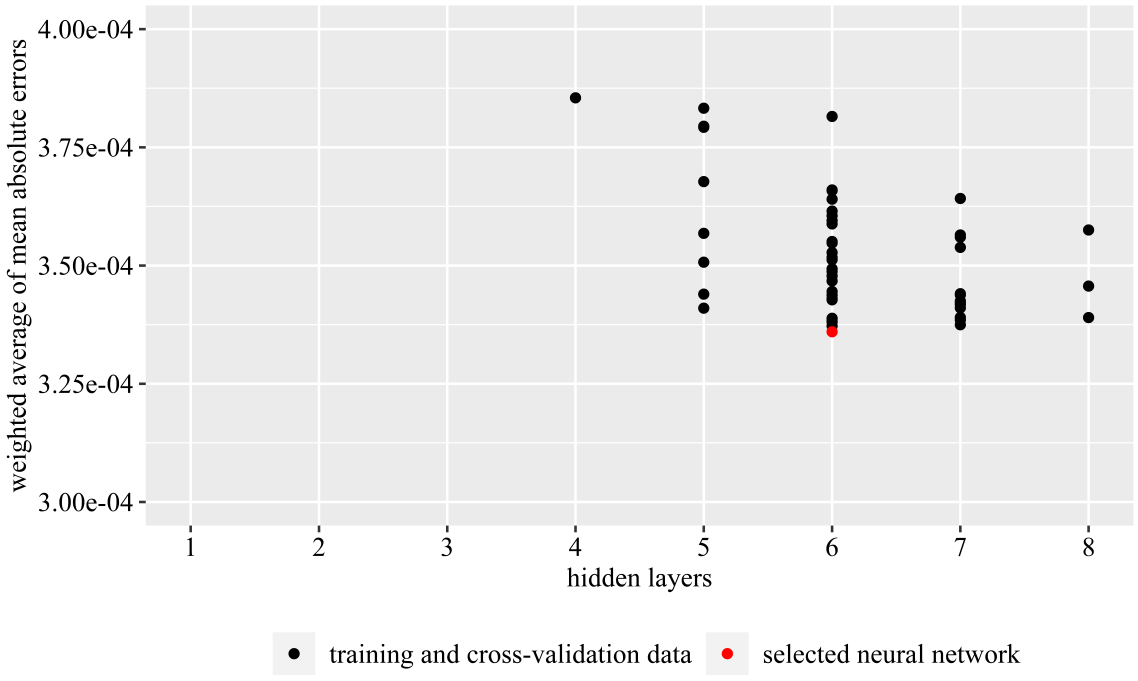

Figure 4.5: Weighted average of mean absolute errors on training and cross-validation data

while θ_i not converged **do**
$i \leftarrow i + 1$
$g_i \leftarrow \bigtriangledown_\theta f_i(\theta_{i-1})$ (get gradients from loss function at time step i)
$m_i \leftarrow 0.9 \cdot m_{i-1} + (1 - 0.9) \cdot g_i$ (update biased first moment estimate)
$u_i \leftarrow max(0.999 \cdot u_{t-1}, |g_i|)$ (update exponentially weighted infinity norm)
$\theta_i \leftarrow \theta_{t-1} - (\alpha / (1 - 0.9^i)) \cdot m_i / u_i$ (update parameters)
return θ_i

Algorithm 5: Adamax optimization algorithm **while** loop [38]

β_1 and β_2 are set to their default values of 0.9 and 0.999, respectively [38]. Under certain circumstances, it might make sense to subsitute each loss function into Algorithm 5 and apply the backpropagation algorithm [30]. However, since MSE and SSE are nearly the same, it is much easier to just examine the effect that the leading $\frac{1}{n}$ term (2.26) has on the parameter (θ) updates. The following code was written to perform pseudo parameter (θ) updates under a relatively simple set of assumptions.

The selected neural network was tested using several different optimization algorithms (Figure 4.6) that are available in *TensorFlow* [7] and other deep learning software packages. Adamax [38] outperformed all other optimization algorithms, as indicated by having the lowest sum of mean absolute errors on training and cross-validation data (Figure 4.6). The Adamax optimization algorithm was also tested by varying the learning rate (α), which is a scaling factor that controls how much the parameters (θ) are updated during training (Figure 4.7). The default Adamax learning rate is set to 0.002 [7, 19, 38].

When the Adamax learning rate is set to 0.00075, the training error lowers to 1.9e-04, and the cross-validation error remains at 3.4e-04 (Figure 4.7). As a result, the learning rate was set to 0.00075 to allow for more gradual parameter (θ) updates. 20 neural networks were trained using this learning rate, and the results are plotted and shown in Figures 4.8, 4.9, and **??**-**??**.

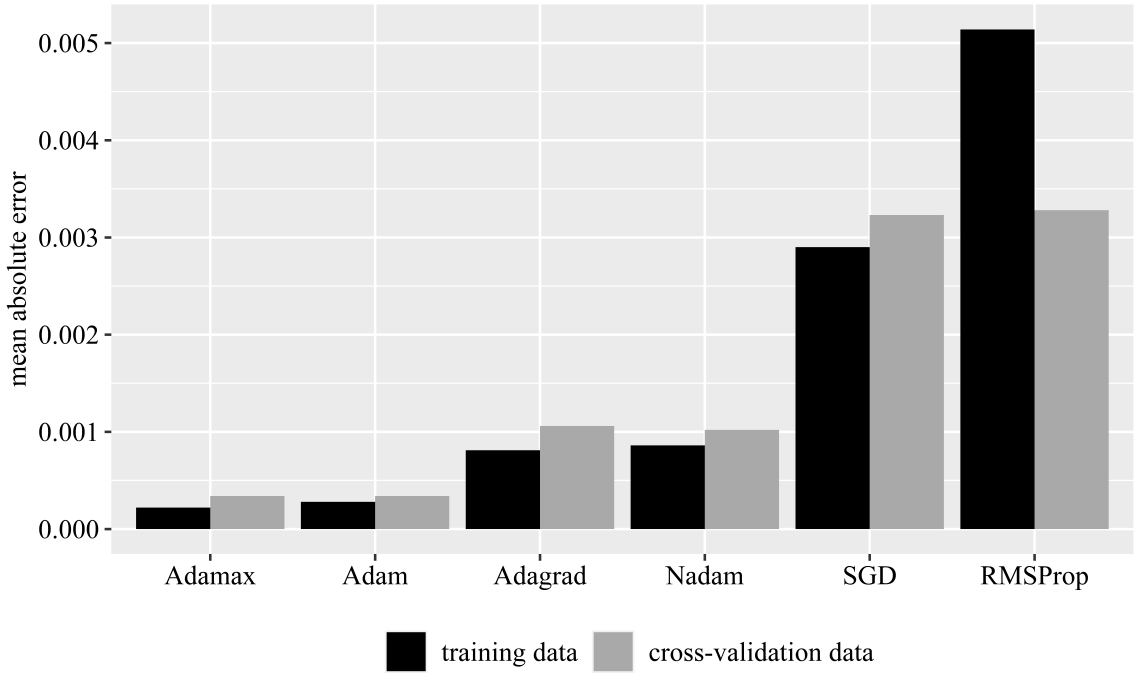

Figure 4.6: Training the selected neural network using different optimization algorithms

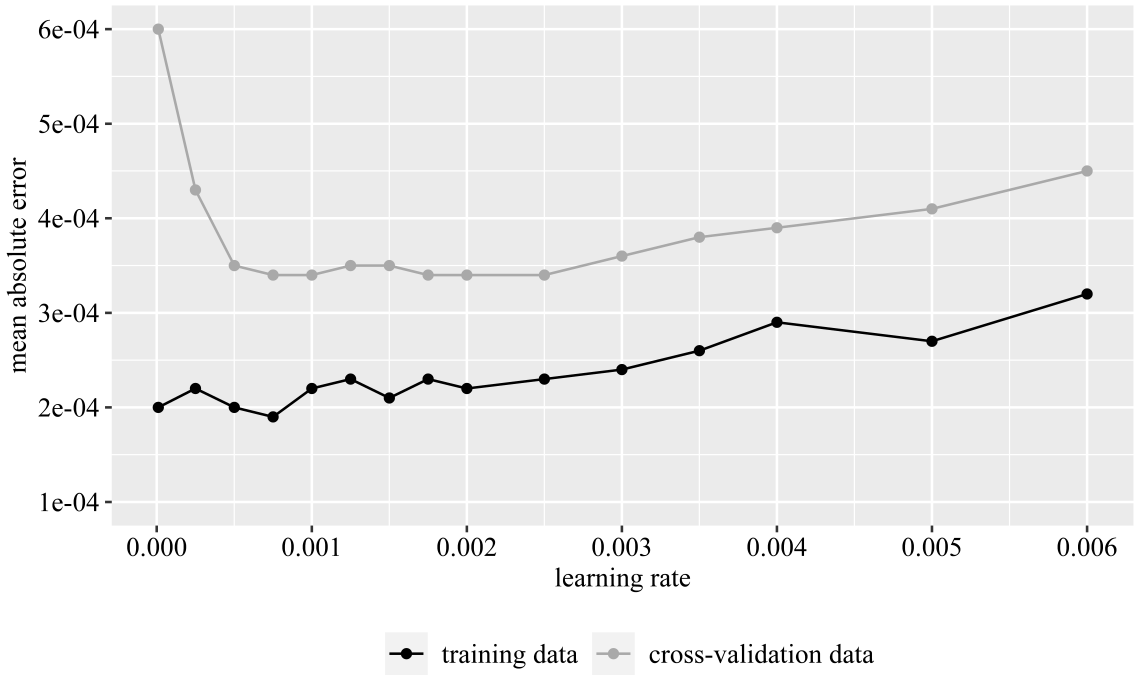

Figure 4.7: Training the selected neural network using different learning rates

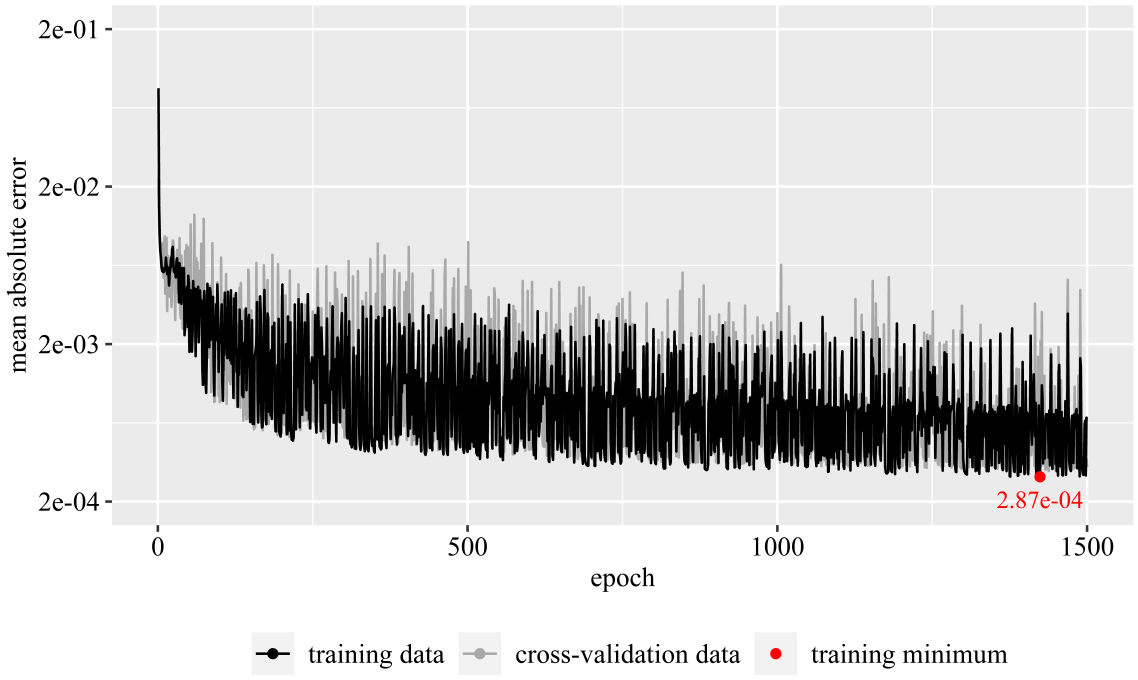

Figure 4.8: Training the selected neural network for 1,500 epochs

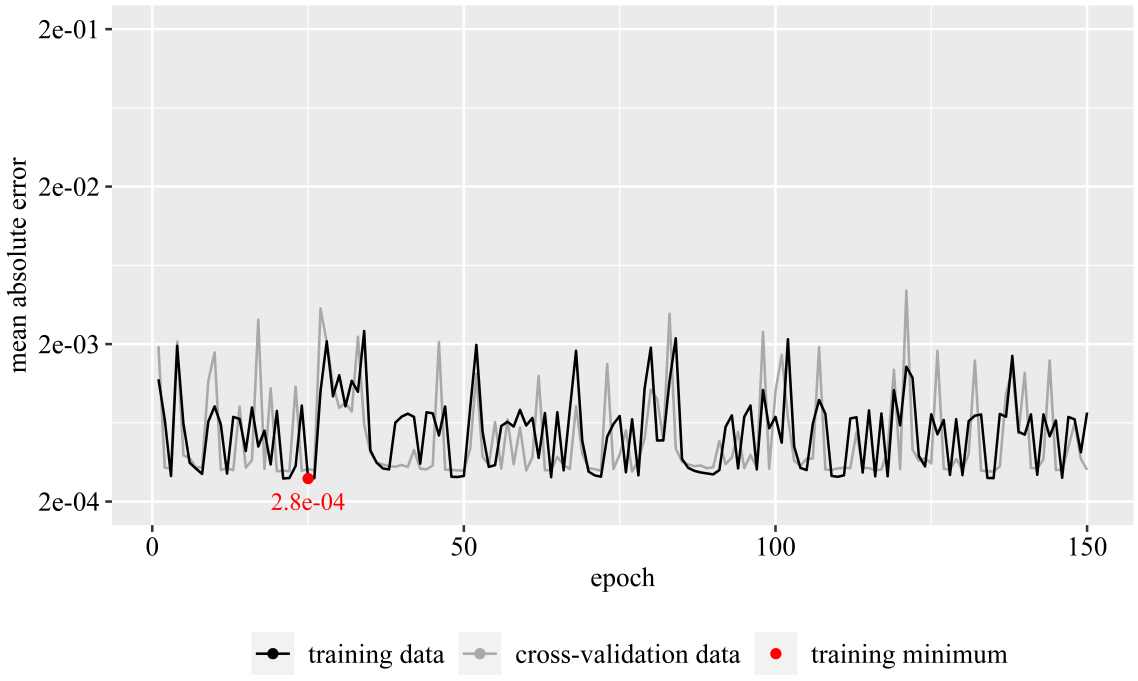

Figure 4.9: Training the selected neural network for an additional 150 epochs

4.3 Model Averaging

The purpose of using more than one neural network is to reduce generalization error, which is the expected error on new input [41]. This was done by training 50 neural networks separately and then averaging the results of predicted keff values for the test data, which consists of 26,593 output files the neural networks have never seen. The idea behind this technique, known as model averaging, is that each neural network in the ensemble metamodel will not make the same errors on test data [41]. The expected squared error of the ensemble metamodel, taken from Goodfellow, et al., [41] is shown in Eq. 10 for a set of n regression models with errors

If the errors are perfectly correlated, then c = v, and the expected squared error of the ensemble metamodel reduces to v [41]. Conversely, if the errors are perfectly uncorrelated, then c = 0, and the expected squared error reduces to v/n [41]. The ensemble metamodel test errors are plotted and shown in Fig. 14.

Chapter 5: Estimating Process Criticality Accident Risk

5.1 Results

As expected, the results shown in Fig. 14 indicate that the test errors are partially correlated, as indicated by a decrease in test error as the number of neural networks in the ensemble metamodel increases. These results also indicate that setting the learning rate to a value that corresponds to a lower training error allows the ensemble metamodel to asymptotically approach this lower value (Fig. 14). Based on these results, it was determined that 10-20 neural networks would likely be enough to minimize the test error on the larger test data set (Table 6) without significantly extending the time it takes to train the ensemble metamodel. An ensemble metamodel, comprised of 10 neural networks, was re-trained on 1,584,973 output files. To accommodate the larger data set, batch size was increased to 32,768 output files. The ensemble metamodel test errors are plotted and shown in Fig. 15, for a test data set that consists of 316,993 output files.

The test errors shown in Fig. 15 are slightly higher than previous test errors (Fig. 14), but low enough to forego further training and optimization. These test errors were re-plotted to show the variation in individual output files (Fig. 16).

As shown in Fig. 16, the test errors become slightly more diffuse, but with fewer outliers, as keff increases. The keff standard deviations of all 1,584,973 output files were plotted (Fig. 17) to summarize the keff standard deviations and show that test errors (Fig. 16) follow a similar trend.

The average ensemble metamodel test error is 2.74e-04 (Fig. 15), which is very close to the average standard deviation of 2.97e-04 (Fig. 17). Even though these results are comparable, there are hundreds of outliers, including several that are 5-10 times larger than 2.74e-04. Based on the results shown in Fig. 15 and Fig. 16, it is reasonable to conclude that the ensemble metamodel predicts keff to within

Now that the ensemble metamodel has been validated against MCNP6.2 [15] and ENDF/B-VII.1 nuclear data [17], it can be combined with the Bayesian network and used to estimate process criticality accident risk. A preliminary estimate was performed by fitting gamma distributions to the data and running 1 million simulations (Fig. 18).

The results shown in Fig. 18 indicate that the hybrid risk model fails to predict a process criticality accident at a frequency However, since there are outliers, an additional 10 to 100 million simulations were run (Fig. 19, Fig. 20).

The results shown in Fig. 19 and Fig. 20 indicate that the risk of a process criticality accident in Building 332 is estimated to be 1e-07 accidents per year, based on the number of points above the upper subcritical limit. This estimate has corresponding critical masses of approximately 600-730 grams of plutonium (95% 239Pu, 5% 240Pu by weight), which are very similar to 3 of the 7 spherical plutonium critical masses associated with historical process criticality accidents [1]. Additional estimates were performed by fitting normal, log-normal, and Weibull distributions to the data and running 100 million simulations for each probability distribution (Fig. 21).

The results shown in Fig. 21 correspond to process criticality accident risk estimates of 5.9e-05 and 1e-07 accidents per year for log-normal and gamma distributions, and < 1e-08 accidents per year for normal and Weibull distributions, respectively. These estimates are significantly lower than the previous process criticality accident risk estimate of 3.4e-05 accidents per year, which was calculated using fault tree analysis [12]. This difference is primarily due to the removal of most of the fissile material from Building 332 [47, 48], as well as the reduction from Category I/II to Category III operations, which limits the quantities and types of plutonium that are allowed in Building 332 [49].

Another notable difference was the wide variation in the results (Fig. 21), which are driven by probability distributions—a consistent weak point in nearly every risk model. The obvious benefit of the hybrid risk model is that these probability distributions can be updated and compared to others, which haven't been tested. Another benefit is that the hybrid risk

model is based on the actual physics of a process criticality accident—as opposed to fault tree analysis, which relies on estimates of parameters that could result in a process criticality accident [12].

5.2 Uncertainty Analysis

Chapter 6: Conclusions

The process criticality accident risk for Building 332 was estimated to be 5.9e-05 and 1e-07 accidents per year for log-normal and gamma distributions, respectively. Additional estimates of process criticality accident risk were < 1e-08 accidents per year for normal and Weibull distributions. These estimates were significantly lower than the previous process criticality accident risk estimate of 3.4e-05 accidents per year [12], although the previous estimate was based on fault tree analysis that was performed prior to de-inventory [47, 48]. Overall, the hybrid risk model worked well and generated results that are consistent with critical benchmark experiments [18] and historical process criticality accident data [1].

The hybrid risk model presented in this paper uses a Bayesian network and an ensemble metamodel to generate conditional probability queries, predict keff values, and formulate estimates of process criticality accident risk. Aside from the risk-based application described in this paper, the hybrid risk model can also be used to develop criticality controls, guide the placement of criticality accident alarm system detectors, and predict the number of safety violations that will occur each year in the facility.

The process of training and optimizing the ensemble metamodel was performed manually, due to the large number of output files that were used, as well as the large number of options that are available to optimize neural network architectures. Although there are several algorithms that could have been used to automate this process, the 1-2 hours it took to train each neural network using the smaller data set proved to be too much to allow these algorithms to run continuously. In comparison, it took 8-9 hours to train each neural network using the larger data set, which translated into four days of runtime to train the ensemble metamodel.

Bibliography

[1] Final Site-wide Environmental Impact Statement for Continued Operation of Lawrence Livermore National Laboratory and Supplemental Stockpile Stewardship and Management Environmental Impact Statement, DOE/EIS-0348. Technical report, U.S. Department of Energy, National Nuclear Security Administration, 2005.

[2] Development of Probabilistic Risk Assessments for Nuclear Safety Applications, DOE-STD-1628-2013. Technical report, U.S. Department of Energy, 2016.

[3] *MCNP User's Manual, Code Version 6.2*. LA-UR-17-29981. Los Alamos National Laboratory, 2017.

[4] National Security Strategy of the United States of America. Technical report, Office of the President of the United States, 2017.

[5] Nuclear Posture Review. Technical report, U.S. Department of Defense, 2018.

[6] International nuclear and radiological event scale (ines), 2020.

[7] M. Abadi. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv: 1603.04467v2*, 2016.

[8] A.E. Ames, N. Mattucci, S. MacDonald, G. Szonyi, and D.M. Hawkins. Quality loss functions for optimization across multiple response surfaces. *Journal of Quality Technology*, 2018.

[9] T.W. Anderson and D.A. Darling. A test of goodness of fit. *Journal of the American Statistical Association*, 1954.

[10] S. Au and E. Patelli. Rare event simulation in finite-infinite dimensional space. *Reliability Engineering and System Safety*, 2016.

[11] K.P. Bennett and A. Demiriz. Semi-supervised support vector machines. 1999.

[12] D.P. Bertsekas and J.J. Tsitsiklis. *Introduction to Probability, 2nd Edition*. Athena Scientific, 2008.

[13] G. Blatman and B. Sudret. Adaptive sparse polynomial chaos expansion based on least angle regression. *Journal of Computational Physics*, 2011.

[14] A. Bobbio, L. Portinale, M. Minichino, and E. Ciancamerla. Improving the analysis of dependable systems by mapping fault trees into Bayesian networks. *Reliability Engineering and System Safety*, 2001.

[15] F. Cadini and A. Gioletta. A Bayesian Monte Carlo-based algorithm for the estimation of small failure probabilities. *Reliability Engineering and System Safety*, 2016.

[16] F. Cadini, A. Gioletta, and E. Zio. Improved metamodel-based importance sampling for the performance assessment of radioactive waste repositories. *Reliability Engineering and System Safety*, 2015.

[17] M.B. Chadwick. ENDF/B-VII.1 nuclear data for science and technology: Cross sections, covariances, fission product yields, and decay data. *Nuclear Data Sheets*, 2011.

[18] F. Chollet. Keras, 2020.

[19] F. Chollet and J.J. Allaire. *Deep Learning with R*. Manning Publications Co., 2018.

[20] H. Cramér. On the composition of elementary errors. *Scandinavian Actuarial Journal*, 1928.

[21] T. Crestaux, O. Le Maître, and J.M. Martinez. Polynomial chaos expansion for sensitivity analysis. *Reliability Engineering and System Safety*, 2009.

[22] G. Csardi and T. Nepusz. The igraph software package for complex network research. *InterJournal*, 2006.

[23] R.B. D'Agostino and M.A. Stephens. *Goodness-of-Fit Techniques*. Marcel Dekker, 1986.

[24] A. Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2010.

[25] M.L. Delignette-Muller and C. Dutang. fitdistrplus: An R Package for Fitting Distributions. *Journal of Statistical Software*, 2015.

[26] J.L. Devore. *Probability and Statistics for Engineering and the Sciences, 9th Edition*. Cengage, 2016.

[27] V. Dubourg, B. Sudret, and F. Deheeger. Metamodel-based importance sampling for structural reliability analysis. *Reliability Engineering and System Safety*, 2013.

[28] K. Durga Rao, V. Gopika, V.V.S. Sanyasi Rao, H.S. Kushwaha, A.K. Verma, and A. Srividya. Dynamic fault tree analysis using Monte Carlo simulation in probabilistic safety assessment. *Reliability Engineering and System Safety*, 2008.

[29] M. Čepin and B. Mavko. A dynamic fault tree. *Reliability Engineering and System Safety*, 2002.

[30] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. Massachusetts Institute of Technology, 2016.

[31] O. Hahn and F.W. Strassman. Über die entstehung von radiumisotopen aus urandurch bestrahlen mit schnellen und verlangsamten neu-tronen (about the formation of radium isotopes from uranium by irradiation with fast and slow neutrons). *The Science of Nature*, 1938.

[32] H.O. Hartley. Smalled composite designs for quadratic response functions. *Biometrics*, 1959.

[33] G.E. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 2006.

[34] S. Højsgaard. Graphical Independence Networks with the gRain package for R. *Journal of Statistical Software*, 2012.

[35] S. Højsgaard, D. Edwards, and S.L. Lauritzen. *Graphical Models with R*. Springer-Verlag, 2012.

[36] C. Kermisch and P.E. Labeau. Communicating about nuclear events: Some suggestions to improve ines. *Reliability Engineering and System Safety*, 2013.

[37] N. Khakzad, F. Khan, and P. Amyotte. Safety analysis in process facilities: Comparison of fault tree and Bayesian network approaches. *Reliability Engineering and System Safety*, 2011.

[38] D.P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. 2011.

[39] R.A. Knief. *Nuclear Criticality Safety*. American Nuclear Society, 2000.

[40] D. Koller and N. Friedman. *Probabilistic Graphical Models*. Massachusetts Institute of Technology, 2009.

[41] M. Kuhn. Building Predictive Models in R Using the caret Package. *Journal of Statistical Software*, 2008.

[42] M. Kuhn and K. Johnson. *Applied Predictive Modeling*. Springer, 2013.

[43] J.F. Lamarsh. *Introduction to Nuclear Engineering*. Addison-Wesley Publishing Co., 1975.

[44] S.L. Lauritzen and Spiegelhalter D. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society*, 1988.

[45] E.L. Lehmann and G. Casella. *Theory of Point Estimation, 2nd Edition*. Springer, 1998.

[46] A. Liaw and M. Wiener. Classification and regression by randomforest. *Northwestern University*, 2002.

[47] F.J. Massey. The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American Statistical Association*, 1951.

[48] R.J. Mattson. An assessment of criticality safety at the Rocky Flats Plant. *DOE/EH/789081. U.S. Department of Energy*, 1989.

[49] J.N. McKamy. Lessons learned from the Rocky Flats Building 771 near miss criticality accident. Technical report, U.S. Department of Energy, National Nuclear Security Administration, 2014.

[50] T.P. McLaughlin. *A Review of Criticality Accidents*. LA-13638. Los Alamos National Laboratory, 2000.

[51] L. Meitner and O.R. Frisch. Disintegration of uranium by neutrons: A new type of nuclear reaction. *Nature*, 1939.

[52] S. Oladyshkin and W. Nowak. Data-driven uncertainty quantification using the arbitrary polynomial chaos expansion. *Reliability Engineering and System Safety*, 2012.

[53] M. Papadrakakis and N.D. Lagaros. Reliability-based structural optimization using neural networks and monte carlo simulation. *Computer Methods in Applied Mechanics and Engineering*, 2002.

[54] J. Pearl. *Probabilistic Reasoning in Intelligent Systems, 2nd Edition*. Morgan Kaufamann Publishers, Inc., 1988.

[55] J. Quiñonero Candela and C.E. Rasmussen. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 2005.

[56] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 2019.

[57] K.A. Reay and J.D. Andrews. A fault tree analysis strategy using binary decision diagrams. *Reliability Engineering and System Safety*, 2002.

[58] R. Remenyte-Prescott and J.D. Andrew. An enhanced component connection method for conversion of fault trees to binary decision diagrams. *Reliability Engineering and System Safety*, 2008.

[59] N.H. Roberts. *Fault Tree Handbook*. NUREG-0492. U.S. Nuclear Regulatory Commission, 1981.

[60] E. Ruijters, D. Reijsbergen, P.T. deBoer, and M. Stoelinga. Rare event simulation for dynamic fault trees. *Reliability Engineering and System Safety*, 2019.

[61] B. Schölkopf and A.J. Smola. *Learning with Kernels: Support Vector Machines, Optimization, and Beyond*. MIT Press, 2001.

[62] M. Scutari. Learning Bayesian Networks with the bnlearn R Package. *Journal of Statistical Software*, 2010.

[63] M. Scutari and J. Denis. *Bayesian Networks with Examples in R*. Chapman & Hall, 2014.

[64] A.J. Smola and P. Bartlett. Sparse greedy gaussian process regression. 2001.

[65] J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific, 2002.

[66] I. Tien and A. Der Kiureghian. Algorithms for Bayesian network modeling and reliability assessment of infrastructure systems. *Reliability Engineering and System Safety*, 2016.

[67] T.J. Urbatsch, R.A. Forster, R.E. Prael, and R.J. Beckman. Estimation and interpretation of k_{eff} confidence intervals in MCNP. *Nuclear Technology*, 1995.

[68] H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016.

[69] R. Zhang, Y. Liu, and H. Sun. Physics-informed multi-LSTM networks for metamodeling of nonlinear structures. *Computer Methods in Applied Mechanics and Engineering*, 2020.

Table C. Graded Safeguards Table

	Attractiveness Level	Pu/U-233 Category (kg)				Contained U-235/Separated Np-237/Separated Am-241 and Am-243 Category (kg)				All E Materials Category IV
		I	II	III	IV[1]	I	II	III	IV[1]	
WEAPONS Assembled weapons and test devices	A	All	N/A	N/A	N/A	All	N/A	N/A	N/A	N/A
PURE PRODUCTS Pits, major components, button ingots, recastable metal, directly convertible materials	B	≥ 2	≥ 0.4 < 2	≥ 0.2 < 0.4	< 0.2	≥ 5	≥ 1 < 5	≥ 0.4 < 1	< 0.4	N/A
HIGH-GRADE MATERIALS Carbides, oxides, nitrates, solutions (≥ 25g/L) etc.; fuel elements and assemblies; alloys and mixtures; UF_4 or UF_6 (≥ 50% enriched)	C	≥ 6	≥ 2 < 6	≥ 0.4 < 2	< 0.4	≥ 20	≥ 6 < 20	≥ 2 < 6	< 2	N/A
LOW-GRADE MATERIALS Solutions (1 to 25 g/L), process residues requiring extensive reprocessing; Pu-238 (except waste); UF_4 or UF_6 (≥ 20% < 50% enriched)	D	N/A	≥ 16	≥ 3 < 16	< 3	N/A	≥50	≥ 8 < 50	< 8	N/A
ALL OTHER MATERIALS Highly irradiated[3] forms, solutions (< 1g/L), compounds; uranium containing < 20% U-235 or < 10% U-233[2](any form, any quantity)	E	N/A	N/A	N/A	Reportable Quantities	N/A	N/A	N/A	Reportable Quantities	Reportable Quantities

[1]The lower limit for Category IV is equal to reportable quantities in this Order.

[2]The total quantity of U-233 = (Contained U-233 + Contained U-235). The category is determined by using the Pu/U-233 side of this table.

[3]In this Order "highly irradiated" is defined in Attachment 4 (Definitions).