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

May 16, 2021

Dissertation directed by

Shahram Sarkani
Professor of Engineering Management and Systems Engineering

Thomas A. Mazzuchi
Professor of Engineering Management and Systems Engineering and of Decision Sciences

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 May 16, 2021. 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 of Engineering Management and Systems Engineering, Dissertation Co-Director

Thomas A. Mazzuchi, Professor of Engineering Management and Systems Engineering and of Decision Sciences, Dissertation Co-Director

Amir Etemadi, Associate Professor of Engineering and Applied Science, Committee Member

Thomas Holzer, Professional Lecturer of Engineering Management and Systems Engineering, Committee Member

Joseph P. Blackford, Professional Lecturer of Engineering Management and Systems Engineering, Committee Member

Dedication

For my family.

Acknowledgments

I would first and foremost like to thank my wife, Katrina, for supporting me throughout my undergraduate and graduate education, and for paving the way so that I could eventually become the second Dr. Zywiec in our family. 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 Dr. Elizabeth Heckmaier, Dr. Anthony Nelson, and Shauntay Coleman, for providing technical feedback and encouragement, 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, Dr. Debdas Biswas, Dr. Forrest Brown, Dr. Shang-Chih "Philip" Chou, Dr. Song Huang, Dr. Soon Sam Kim, Dr. Jerry McKamy, Dr. Thomas McLaughlin, Catherine Percher, Dr. John Scorby, and Paul Yap-Chiongco, for teaching me everything I know about nuclear criticality safety.

Abstract

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

Since the discovery of fission [61] and subsequent first criticality of Chicago Pile-1 [57], more than 60 criticality accidents have occurred 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* [86]. This dissertation focuses on the development of a methodology that uses a coupled Bayesian network and neural network metamodel to estimate process criticality accident risk. This methodology is essentially a generalized software-based framework that was written in the R programming language and applied to fissile material operations in the Plutonium Facility (Building 332) at Lawrence Livermore National Laboratory. The most significant benefit of using Building 332 as a case study was that it enabled a comprehensive review of the entire process of building a coupled Bayesian network and neural network metamodel and estimating process criticality accident risk. This review also enabled iterative improvements to be made to the methodology as it was being developed. Overall, the coupled Bayesian network and neural network metamodel worked extremely well and has several advantages over existing methodologies.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

Since the discovery of fission [61] and subsequent first criticality of Chicago Pile-1 [57], more than 60 criticality accidents have occurred throughout the world. A *criticality accident* is defined as "the release of energy as a result of accidentally producing a self-sustaining or divergent fission chain reaction" [86]. Criticality 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* [86].

In an effort to enable prompt communication of the safety significance of nuclear and radiological events to a wider audience, the International Nuclear and Radiological Event Scale (INES) was introduced in 1990 by the International Atomic Energy Agency (IAEA) [24]. INES goes from level 0 to level 7 and is intended to be logarithmic, with each level representing an accident approximately ten times as severe as the next level down [24]. The Chernobyl disaster, for example, is considered 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" [24]. On the other end of the scale is an INES level 0 event, which is defined as a "deviation with no off-site consequences" [24]. 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" [24, 71].

The primary danger posed by a criticality accident is the sudden burst of neutron and gamma ray radiation that occurs, which can be lethal to those within 10-20 feet [70, 74]. At

distances beyond 20 feet, the burst of radiation is less intense, and its effects are survivable—assuming prompt evacuation occurs [70, 74, 86]. Other hazards associated with a criticality accident include secondary sources of radiation, such as activated materials and airborne fission products [70, 74, 86].

Nuclear criticality safety is a discipline within the field of nuclear engineering that is dedicated to the prevention of criticality accidents [74]. In the 1940s, nuclear physicists performed experiments with fissile material, and these experiments were used to calculate limits and safety margins for fissile material operations. Over time, these hand calculations were replaced by increasingly complex radiation transport codes, which are used to calculate the *effective neutron multiplication factor* (k_{eff}) of systems containing fissile material. In the simplest terms, if $k_{eff} \geq 1$, the system will produce a self-sustaining or divergent fission chain reaction (i.e., a criticality accident). If $k_{eff} < 1$, the system is subcritical—and presumably safe. Today, radiation transport codes are used almost exclusively to calculate k_{eff} to ensure that a criticality accident will not occur. The ten parameters that affect nuclear criticality are shown in Table 1.1 [74].

Table 1.1: Parameters that affect nuclear criticality

parameter	description
mass	mass of fissile material
absorption	material that absorbs neutrons
geometry/shape	specific geometric arrangement of fissile and non-fissile materials
interaction	neutron interaction between adjacent units of fissile material
concentration/density	concentration or density of fissile material
moderation	material that slows down (or thermalizes) neutrons
enrichment	increased weight fraction of fissile isotope in a particular type of fissile material (e.g., ^{235}U in enriched uranium)
reflection	material that reflects neutrons back into the system
volume	volume of the system
temperature	temperature of the system

1.1 Problem Statement

Process criticality accidents are difficult to predict not only because they are rare [86], but also because fissile material operations are complex and difficult to analyze [74, 83]. To illustrate this complexity, two sets of k_{eff} contours are plotted and shown in Figure 1.1 and Figure 1.2. The first plot represents a sphere of alpha-phase plutonium ($\rho = 19.86$ g/cm^3, 95% ^{239}Pu, 5% ^{240}Pu by weight), homogeneously dispersed in a sphere of water, with one inch of water reflection (Figure 1.1). The second plot represents a sphere of alpha-phase plutonium, homogeneously dispersed in a sphere of magnesium oxide, with one inch of water reflection (Figure 1.2).

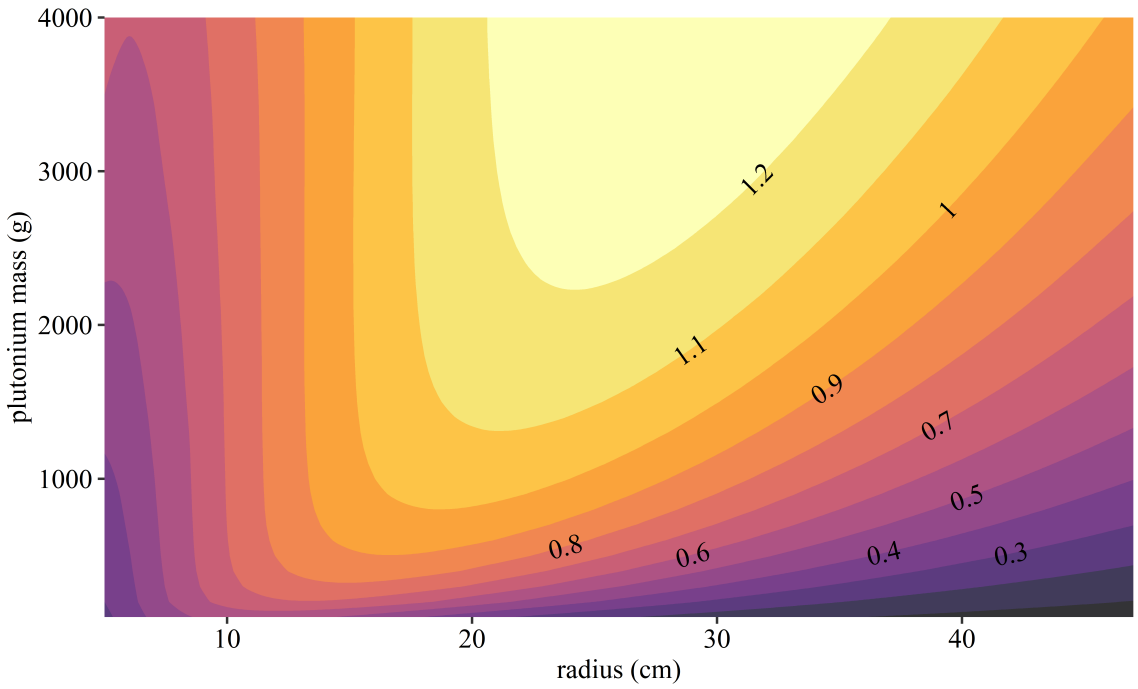

Figure 1.1: k_{eff} contour plot of 0-4 kilograms of alpha-phase plutonium, homogeneously dispersed in a sphere of water, with one inch of water reflection

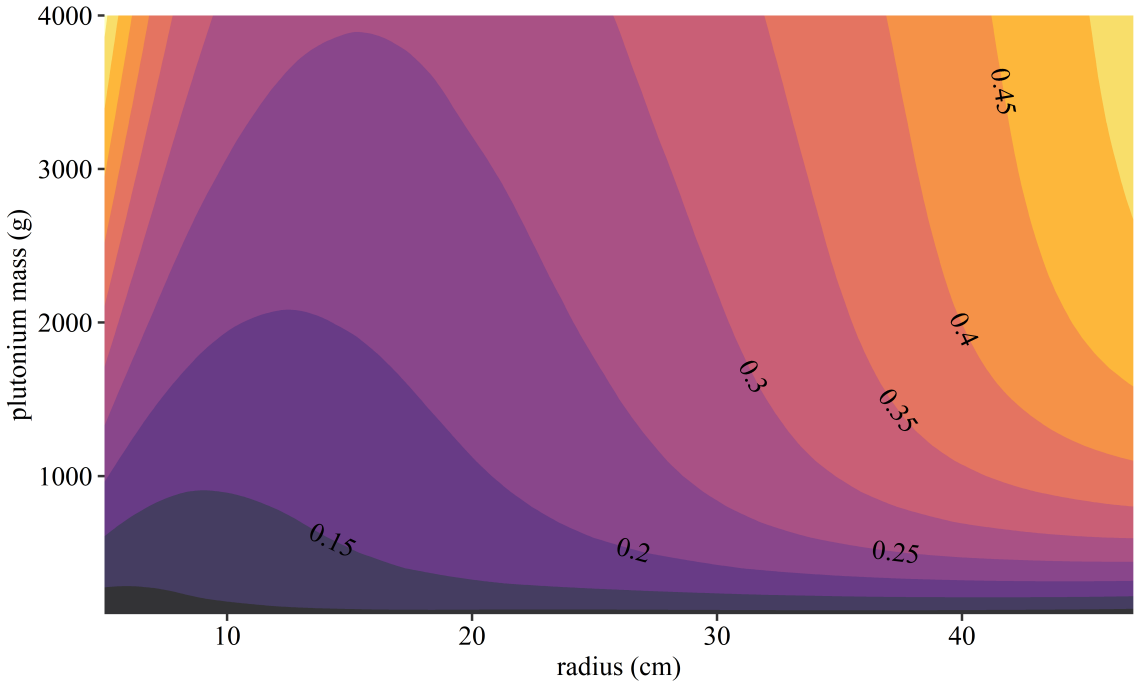

Figure 1.2: k_{eff} contour plot of 0-4 kilograms of alpha-phase plutonium, homogeneously dispersed in a sphere of magnesium oxide, with one inch of water reflection

Despite the similarities between these systems, the contours of both plots are much different. If Figure 1.1 and Figure 1.2 are assumed to be a simplified representation of fissile material operations in a nuclear facility, then it is reasonable to also assume that the majority of operations will fall somewhere in the lower left-hand corner of either plot, since most operations occur at k_{eff} values $\ll 1$ [74]. These assumptions lead to two problems:

1. The unique contours of these plots make it difficult to estimate process criticality accident risk ($P(k_{eff} \geq 1)$). This difficulty is compounded by the fact that fissile material operations often include changes in moderation, reflection, and other parameters that affect nuclear criticality, which increases the dimensionality of the system, which covers all fissile material operations.

2. A radiation transport code, such as MCNP [19], is needed to calculate k_{eff}. However, a typical MCNP calculation takes several seconds or minutes to run, which introduces a computational bottleneck if millions (or even billions) of calculations are needed to fully characterize a complex, high-dimensional system.

The methodology described in this dissertation addresses the first problem by fitting probability distributions to facility data and using a Bayesian network to sample parameters that affect nuclear criticality. The second problem is addressed by using a neural network metamodel to predict k_{eff} values for samples that the Bayesian network generates, based on calculations that were performed using MCNP [19].

1.2 Research Questions

During the early stages of this work, several questions were raised about neural networks, Bayesian networks, and process criticality accident risk:

1. Can neural networks be trained to predict k_{eff} to high precision?
2. Can a coupled Bayesian network and neural network metamodel be used to estimate process criticality accident risk?
3. Is a methodology based on this type of model reasonably efficient and robust?
4. How does this methodology compare to industry-standard fault tree analysis?

These questions were addressed by applying the methodology described in this dissertation to fissile material operations in the Plutonium Facility (Building 332) at Lawrence Livermore National Laboratory. The results of this case study were then compared to fault tree analysis that was performed for Building 332 [125].

1.3 Research Objectives

The questions from the previous section were reframed as research objectives to facilitate the development of the methodology and its application to the selected case study. These research objectives are provided below.

1. Develop a process for training neural networks to predict k_{eff} to high precision
2. Develop a methodology that uses a coupled Bayesian network and neural network metamodel to estimate process criticality accident risk

3. Establish trade-offs between neural network performance and training time

4. Compare process criticality accident risk estimates to other sources of data, such as critical mass studies, historical process criticality accidents, and fault tree analysis

These objectives were achieved by building a software-based framework in R, applying it to fissile material operations in a nuclear facility, and analyzing the results. Building 332 was selected for this application because it is a small facility, and the risk of a process criticality accident was expected to be low, which was thought to present a greater challenge, in terms of sampling [125]. It was also important to ensure that the methodology followed industry best practices. In support of this objective, the following references were used to guide the development of this methodology:

[6] Recommendations for Analyzing Accidents Under the National Environmental Policy Act. U.S. Department of Energy, 2002.

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

[14] ANSI/ANS-8.1-2014, Nuclear Criticality Safety in Operations with Fissionable Materials Outside Reactors. American Nuclear Society, 2014.

[15] DOE-STD-3009-2014, Preparation of Nonreactor Nuclear Facility Documented Safety Analysis. U.S. Department of Energy, 2014.

[17] DOE-STD-1104-2016, Review and Approval of Nuclear Facility Safety Basis and Safety Design Basis Documents. U.S. Department of Energy, 2016.

The first reference provides background information about how and why U.S. Department of Energy (DOE) sites estimate process criticality accident risk [6]. The other references include one national consensus standard [14] and three DOE standards [12, 15, 17], which are required to be implemented in all nuclear and radiological facilities that are owned and operated under contract by DOE.

1.4 Dissertation Organization

This dissertation is divided into five chapters:

Chapter 1 introduces the problem statement and provides a brief overview of research questions and objectives.

Chapter 2 provides a mathematical background in nuclear criticality, nuclear cross sections, and Monte Carlo radiation transport, as well as an overview of historical process criticality accidents and critical experiments. Chapter 2 also provides information about Bayesian networks and neural networks, which form the basis of the methodology described in this dissertation.

Chapter 3 describes the methodology that was used to build a coupled Bayesian network and neural network metamodel and estimate process criticality accident risk.

Chapter 4 discusses the results of applying the methodology to fissile material operations in Building 332.

Chapter 5 discusses the conclusions of the case study, the contributions this dissertation makes to the broader field of risk analysis, and opportunities for future work.

Chapter 2: Literature Review

This chapter provides a mathematical background in nuclear criticality, nuclear cross sections, and Monte Carlo radiation transport, as well as an overview of historical process criticality accidents and critical experiments. This chapter also provides information about Bayesian networks and neural networks, which form the basis of the methodology described in this dissertation.

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 releasing neutrons and gamma rays (Figure 2.1) [55]. This phenomenon was discovered on December 17, 1938 by Otto Hahn and Fritz Strassmann [61] and later explained in an article written by Lise Meitner and Otto Frisch, which was published in *Nature* [87].

Nuclear criticality is a self-sustaining fission chain reaction that occurs when fissile material is assembled into a critical mass. 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 (2.1) [74]. A typical neutron generation (i.e., neutron lifetime) lasts approximately 1e-04 to 1e-07 seconds [55]. A simplified form of the equation to calculate k_{eff} is shown in 2.1 [74].

$$k_{eff} = \frac{\text{neutrons produced from fission in one generation}}{\text{neutrons produced from fission in the previous generation}} \tag{2.1}$$

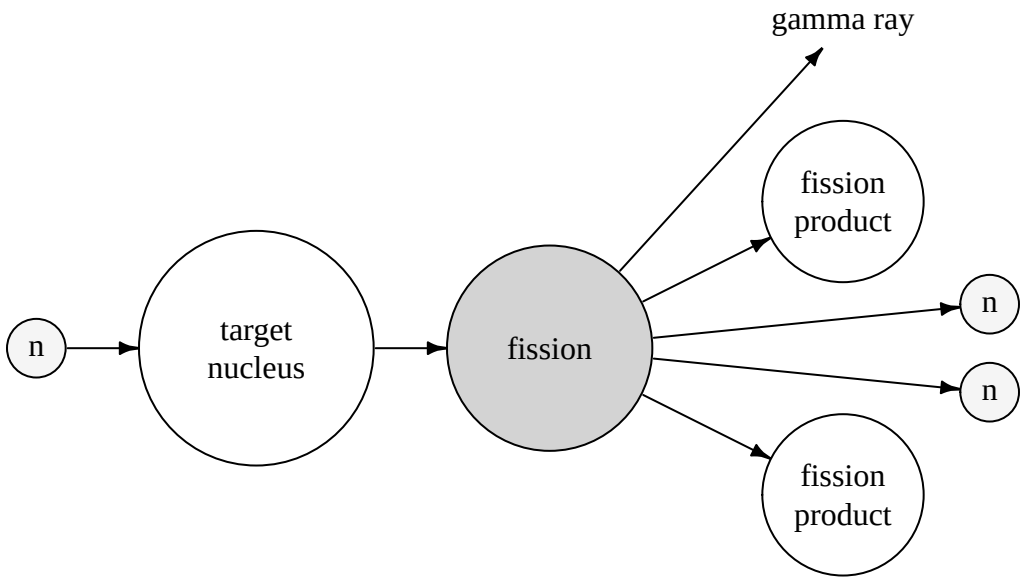

Figure 2.1: Fission diagram

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 critical ($k_{eff} = 1$); if it is greater, the system is supercritical ($k_{eff} > 1$); if it is less, the system is subcritical ($k_{eff} < 1$) [74].

2.2 Nuclear Cross Sections

A *nuclear cross section* describes the probability that a reaction will occur between an incident particle and a target nucleus (Figure 2.2). The concept of a nuclear cross section is quantified in terms of "characteristic area" where a larger area means a higher probability of interaction. The standard unit for measuring a nuclear cross section (σ) is the *barn*, which is equal to 1e-24 cm^2 [55].

Nuclear cross sections depend on the type and energy of the incident particle, the size of the target nucleus, the type of reaction that occurs (e.g., fission, scattering, absorption), and,

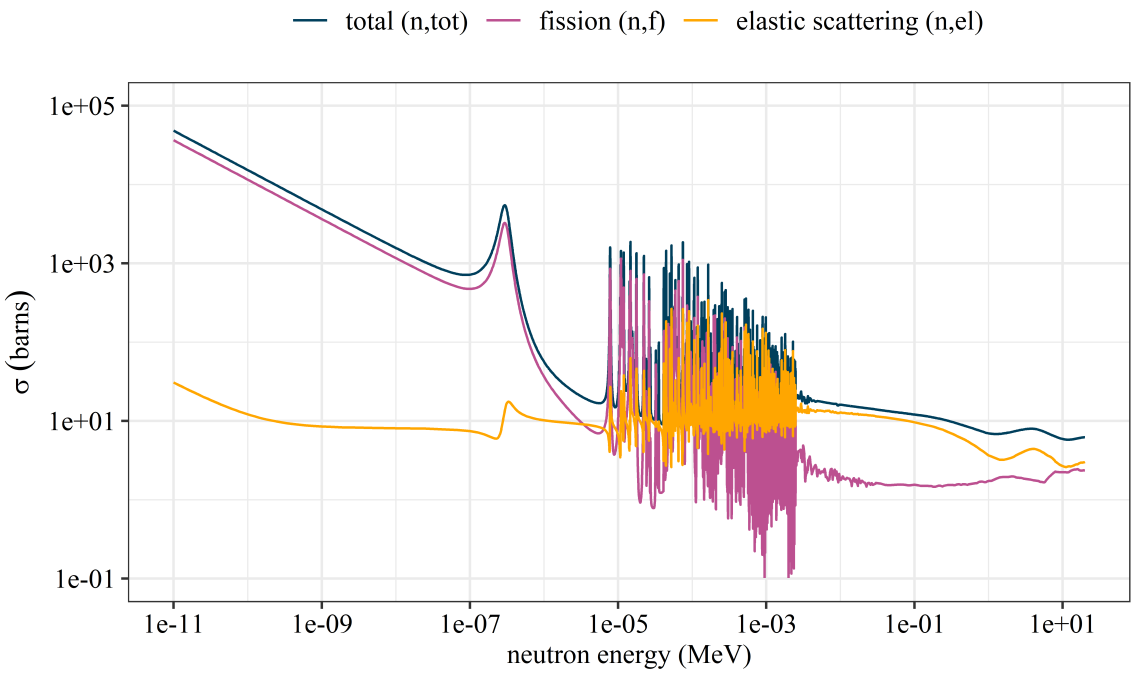

Figure 2.2: ^{239}Pu cross sections at 293.6 K

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 *microscopic cross sections* to distinguish them from *macroscopic cross sections*, which are weighted by the number density (N) of the target material (2.2) [55].

$$\Sigma_t(E) = N\sigma_t(E) \tag{2.2}$$

where:

$\Sigma_t(E)$ = macroscopic total cross section (cm^{-1})

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

$\sigma_t(E)$ = 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 [55]. 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 $p(\Omega \cdot \Omega', E \rightarrow E')d\Omega' dE'$ [55].

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 [42, 55]. The main takeaway from this section is that nuclear cross sections are synonymous with the probability of interaction between incident neutrons and target nuclei, which is an important concept in nuclear criticality.

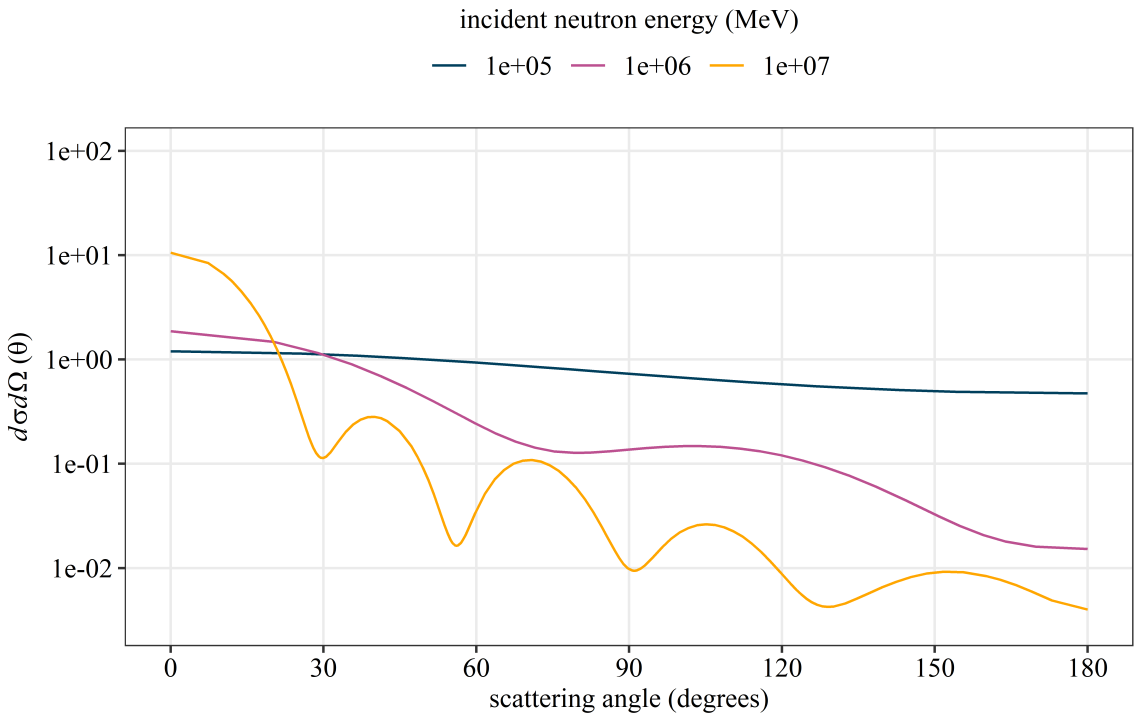

Figure 2.3: ^{239}Pu elastic scattering angular distributions at 293.6 K

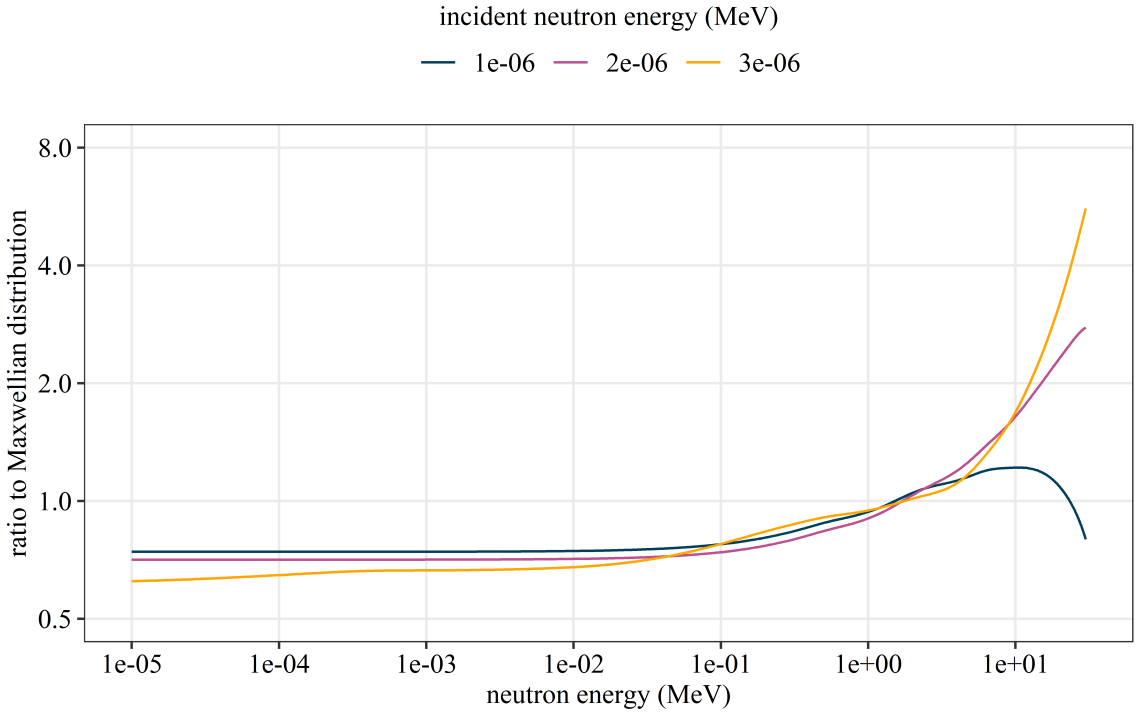

Figure 2.4: ^{239}Pu (n,f) energy distributions at 293.6 K

2.3 Monte Carlo Radiation Transport

MCNP is a general-purpose, continuous-energy Monte Carlo radiation transport code [19] that was used throughout this dissertation. MCNP works by converting an input deck (i.e., text file) into a computer model of a system, which consists of a specific geometric arrangement of fissile and non-fissile 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).

An example of an MCNP input deck is shown in Figure 2.5. This input deck represents 500 grams of alpha-phase plutonium (ρ = 19.86 g/cm^3, 95% ^{239}Pu, 5% ^{240}Pu by weight), homogeneously dispersed in a 13.97-cm radius sphere of water, with one inch of water reflection. The computer model of this input deck is shown in Figure 2.6. One inch of water reflection was added to all of the models used in this dissertation because it represents the equivalent reflection provided by human hands [98], which were assumed to be present under all normal and accident conditions.

```
500 alpha h2o 13.97 none sph $ <mass> <form> <mod> <rad> <ref> <shape>
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                        $ thermal scattering treatment for H2O
m2    1001.80c +2 $ H-1
      8016.80c +1 $ O-16
mt2  lwtr.20t    $ thermal scattering treatment for H2O
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

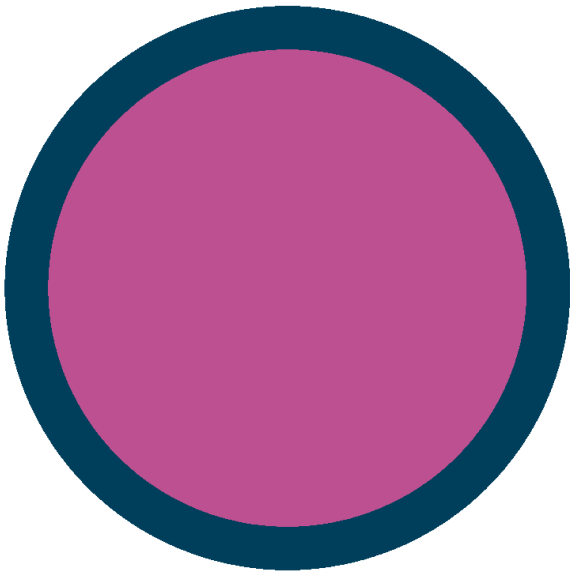

Figure 2.6: MCNP model of 500 grams of alpha-phase plutonium, homogeneously dispersed in a 13.97-cm radius sphere of water (magenta), with one inch of water reflection (dark blue)

Each MCNP input deck is divided into six sets of "cards" or "blocks" [19]:

title card: message block that typically contains information about the MCNP input deck

surface cards: block that specifies one or more surfaces (e.g., "so" is a sphere centered at the origin)

cell cards: block that specifies surface-based cell geometry and assigns materials and densities to cells

material cards: block that specifies material isotopes, weight fractions, and S(α,β) thermal scattering treatments

kcode data card: block that specifies the source that is used to calculate k_{eff}

ksrc data card: block that specifies the coordinates of the source point(s)

In this dissertation, each MCNP input deck was built using the *Grid* (Appendix A.1.1) and *Build* (Appendix A.1.1.1) functions. Each material card consists of one or more isotopes with assigned weight fractions and Z-A identifiers (or ZAIDs) [19]. A ZAID is a two-digit atomic number (Z), followed by a two- to three-digit mass number (A) and an alphanumeric suffix [19]. The ".80c" and ".20t" suffixes after each ZAID correspond to nuclear cross sections and S(α,β) thermal scattering treatments that are based on the ENDF/B-VII.1 nuclear data library [42]. Unless otherwise specified, MCNP defaults to using a "free gas" model for each material card, which assumes all isotopes are unbound. S(α,β) thermal scattering treatments automatically adjust nuclear cross sections below a certain energy

threshold to account for the effects that molecular binding and crystal lattices have on thermal (or low energy) scattering events [42].

The kcode data card in each MCNP input deck was set to simulate random walks for 500 cycles of 10,000 neutrons each [19]. After the first cycle of random walks, the mean number of neutrons produced from fission was calculated, and locations where fissions occur were selected as the new source points for the next cycle. The kcode data card was set to only use cycles 50 through 500 to ensure that the source points were distributed throughout the system, prior to calculating k_{eff} (Figure 2.5). There were seven starting source points specified in each ksrc data card: 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 two-thirds the radius of the inner sphere (Figure 2.5).

There are three estimators that MCNP uses to calculate k_{eff}: collision, absorption, and track length [19]. The collision estimator is calculated using 2.3 [118].

$$k_{eff}^{c} = \frac{1}{N} \sum_{i} W_i \frac{\sum_k f_k \bar{\nu}_k \sigma_{f_k}}{\sum_k f_k \sigma_{t_k}} \tag{2.3}$$

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

$\bar{\nu}_k$ = mean 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 2.4 [118].

$$k_{eff}^{a} = \frac{1}{N} \sum_{i} W_i \bar{\nu}_k \frac{\sigma_{f_k}}{\sigma_{c_k} + \sigma_{f_k}} \tag{2.4}$$

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 2.5 [118].

$$k_{eff}^{t} = \frac{1}{N} \sum_{i} W_i \rho d \sum_{k} f_k \bar{\nu}_k \sigma_{f_k} \tag{2.5}$$

where:

k_{eff}^{t} = track length estimator

i is summed over all neutron trajectories

ρ = atomic density of the cell

d = neutron trajectory track length

MCNP takes the mean of all k_{eff} values for each estimator, for all active cycles, and uses it to calculate the maximum likelihood estimate of k_{eff} (2.6) [19, 118].

$$k^x_{eff} = \frac{1}{C}\sum_{50}^{C} k^x_{eff,c} \tag{2.6}$$

where:

C = number of active cycles

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

Assuming the $k^x_{eff,\,c}$ values for each estimator are independent, the sample standard deviation of the mean estimate of k_{eff} can be obtained by 2.7 [118].

$$\sigma_{\bar{k}^x_{eff}} = \frac{1}{\sqrt{C}}\sigma_{k^x_{eff}} = \left[\frac{1}{C(C-1)}\sum_{c=50}^{C}(k^x_{eff,\,c} - \bar{k}^x_{eff})^2\right]^{1/2} \tag{2.7}$$

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 (2.8) [118].

$$\Sigma = \begin{bmatrix} \sigma^2_{k^c_{eff}} & \sigma^2_{k^{ca}_{eff}} & \sigma^2_{k^{ct}_{eff}} \\ \sigma^2_{k^{ac}_{eff}} & \sigma^2_{k^a_{eff}} & \sigma^2_{k^{at}_{eff}} \\ \sigma^2_{k^{tc}_{eff}} & \sigma^2_{k^{ta}_{eff}} & \sigma^2_{k^t_{eff}} \end{bmatrix} \tag{2.8}$$

A square root solution of a 3×3 identity matrix is multiplied by Σ (2.8) and its transpose to obtain a 3×3 matrix, which further reduces to a 2×2 matrix with $i \times j$ (Σ_{zij}) submatrices (2.9) [118].

$$\begin{aligned} \Sigma_z &= A\Sigma A^T \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} \sigma^2_{k^c_{eff}} & \sigma^2_{k^{ca}_{eff}} & \sigma^2_{k^{ct}_{eff}} \\ \sigma^2_{k^{ac}_{eff}} & \sigma^2_{k^a_{eff}} & \sigma^2_{k^{at}_{eff}} \\ \sigma^2_{k^{tc}_{eff}} & \sigma^2_{k^{ta}_{eff}} & \sigma^2_{k^t_{eff}} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \\ &= \left[\begin{array}{c|cc} \sigma^2_c & \sigma^2_c - \sigma^2_{ca} & \sigma^2_c - \sigma^2_{ct} \\ \hline \sigma^2_c - \sigma^2_{ca} & \sigma^2_c + \sigma^2_a - 2\sigma^2_{ca} & \sigma^2_c + \sigma^2_{at} - \sigma^2_{ca} - \sigma^2_{ct} \\ \sigma^2_c - \sigma^2_{ct} & \sigma^2_c + \sigma^2_{at} - \sigma^2_{ca} - \sigma^2_{ct} & \sigma^2_c + \sigma^2_t - 2\sigma^2_{ct} \end{array}\right] \\ &= \left[\begin{array}{c|c} \Sigma_{z11} & \Sigma_{z12} \\ \hline \Sigma_{z21} & \Sigma_{z22} \end{array}\right] \end{aligned} \tag{2.9}$$

Σ_{z12} and Σ_{z22} are converted to sum of squares matrices with $(n-1)$ degrees of freedom (2.10) [118].

$$S_{zij} = (n-1)\Sigma_{zij} \tag{2.10}$$

A (2.9) is also used to calculate $\bar{d}$ (2.11), which is used in conjunction with the collision estimator (2.3) and sum of squares matrices (2.10) to calculate the maximum likelihood estimate of k_{eff} (2.12) [118].

$$A \begin{bmatrix} \bar{k}^c_{eff} \\ \bar{k}^a_{eff} \\ \bar{k}^t_{eff} \end{bmatrix} = \begin{bmatrix} \bar{k}^c_{eff} \\ \bar{k}^c_{eff} - \bar{k}^a_{eff} \\ \bar{k}^c_{eff} - \bar{k}^t_{eff} \end{bmatrix} = \begin{bmatrix} \bar{k}^c_{eff} \\ \bar{d} \end{bmatrix} \tag{2.11}$$

where:

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

$$k_{eff} = \bar{k}^c_{eff} - S_{z12} S^{-1}_{z22} \bar{d} \tag{2.12}$$

2.12 is the collision estimator (2.3) modified by the variance-weighted residual between the collision estimator and the other two estimators (2.4, 2.5). The k_{eff} value from 2.12 is reported as the "final result" in the MCNP output file [19].

2.4 Historical Process Criticality Accidents

This section provides an overview of historical process criticality accidents and their consequences. All of the accidents discussed in this section were taken from *A Review of Criticality Accidents* [86], which is a report that summarizes 60 historical criticality accidents. This report contains the most comprehensive and complete account of historical criticality accidents and is highly cited throughout this dissertation. The 22 historical process criticality accidents from this report are summarized below [86].

- 21 accidents occurred with fissile material in solutions or slurries. 14 of these were also "multi-pulse" accidents.
- 1 accident occurred with metal ingots.
- 9 workers died as a result of receiving lethal doses of radiation.
- 3 workers had limbs amputated.

The first process criticality accident occurred in 1953 at the Mayak Production Association in Ozyorsk, Russia and involved plutonium nitrate solution, which was momentarily driven supercritical ($k_{eff} > 1$) when approximately 842 grams of plutonium was transferred into a drum with a 500-gram mass limit [86]. During the accident, some of the solution got caught in a vacuum trap, which caused the plutonium mass in the drum to drop below the estimated critical mass of 824 grams and become subcritical ($k_{eff} < 1$) [86]. One worker received an estimated dose of 100 rad and survived [86]. A second worker, who was closer to the drum, received an estimated dose of 1,000 rad [86]. He suffered severe radiation sickness and amputation of both legs, but lived for another 35 years after the accident [86].

The consequences of process criticality accidents are typically quantified by total fission yield (i.e., the total number of fission events that occur) and radiation doses to nearby workers. The total fission yield of a process criticality accident is dependent on the parameters that affect nuclear criticality (Table 3.5), the internal source term (i.e., how many neutrons are available to start a fission chain reaction), and the amount of excess reactivity in the system, which is given in units of dollars ($) and cents (¢). One dollar ($) is the interval between prompt criticality (prompt $k_{eff} = 1$) and delayed criticality (steady-state $k_{eff} = 1$), and the reactivity equivalent of one one-hundredth of a dollar is one cent (¢) [79].

Prompt criticality is achieved with prompt neutrons alone (i.e., neutrons released directly from fission) [79]. Delayed criticality is achieved with both prompt and delayed neutrons (i.e., neutrons released from the subsequent decay of fission fragments) [79]. 21 out of 22 historical process criticality accidents initially achieved prompt supercriticality ($k_{eff} > 1$) [86]. The one notable exception occurred at the Idaho Chemical Processing Plant in 1978 when a scrubbing tank slowly filled with uranyl nitrate and water and is believed to have briefly achieved delayed criticality before a prompt supercritical pulse occurred [86].

2.4.1 Total Fission Yield Estimates

The total fission yield estimates for all 22 historical process criticality accidents are shown in Table 2.1 [86].

Table 2.1: Total fission yield estimates for historical process criticality accidents

number [1]	site	year	total fission yield
1	Mayak Production Association	1953	2.0e+17
2	Mayak Production Association	1957	1.0e+17
3	Mayak Production Association	1958	2.0e+17
4	Oak Ridge Y-12 Plant [2]	1958	1.3e+18
5	Los Alamos Scientific Laboratory	1958	1.5e+17
6	Idaho Chemical Processing Plant [2]	1959	4.0e+19
7	Mayak Production Association [2]	1960	2.5e+17
8	Idaho Chemical Processing Plant [2]	1961	6.0e+17
9	Siberian Chemical Combine [2]	1961	1.2e+16
10	Hanford Works	1962	8.0e+17
11	Mayak Production Association [2]	1962	2.0e+17
12	Siberian Chemical Combine [2]	1962	7.9e+17
13	Siberian Chemical Combine [2]	1963	1.6e+16
14	United Nuclear Fuels Recovery Plant [2]	1964	1.3e+17
15	Electrostal Machine Building Plant [2]	1965	8.0e+15
16	Mayak Production Association [2]	1965	5.5e+17
17	Mayak Production Association [2]	1968	1.3e+17
18	Windscale Works	1970	1.0e+15
19	Idaho Chemical Processing Plant	1978	2.7e+18
20	Siberian Chemical Combine	1978	3.0e+15
21	Novosibirsk Chemical Concentration Plant [2]	1997	5.5e+15
22	JCO Fuel Fabrication Plant [2]	1999	2.5e+18

[1] numbers were taken from *A Review of Criticality Accidents* [86]

[2] multi-pulse accident [86]

The total fission yield estimates shown in Table 2.1 vary widely and provide limited information about the 22 historical process criticality accidents [86]. The 1978 accident at the Idaho Chemical Processing Plant set the record for highest total fission yield, which was estimated to be 4e+19 fissions [86]. This accident took place in a 15,400-liter tank containing approximately 800 liters of uranyl nitrate and water [86]. The large volume of solution allowed more fissions to occur, and it is believed that internal waves were created in the tank, which caused the system to achieve supercriticality multiple times over the course of 15-20 minutes [86]. The system was driven permanently subcritical when approximately 400 liters of water boiled out of the tank [86].

The mean total fission yield for all 22 historical process criticality accidents is 2.3e+18 and the median total fission yield is 2e+17 (Table 2.1). However, if the 1978 accident at the Idaho Chemical Processing Plant is removed, the mean total fission yield lowers to 5.1e+17. The total fission yield estimates shown in Table 2.1 are difficult to reconstruct because the parameters that affect nuclear criticality (Table 3.5), the internal source term, and the amount of excess reactivity in the system were not being tracked. If they had been, these accidents would not have occurred.

2.4.2 Radiation Dose Estimates

The radiation dose estimates for all 22 historical process criticality accidents are shown in Table 2.2 [86]. These doses are given in grays (Gy). One gray is defined as "the absorption of one joule of radiation energy per kilogram of matter" (1 Gy = 1 J/kg) [70]. For context, the lethal doses associated with whole-body exposure to ionizing radiation, taken from T.E. Johnson [70], are shown in Table 2.3.

Table 2.2: Radiation dose estimates for historical process criticality accidents

number [1]	site	year	dose (Gy)
1	Mayak Production Association	1953	1-10
2	Mayak Production Association	1957	3-30
3	Mayak Production Association	1958	6-60
4	Oak Ridge Y-12 Plant [2]	1958	0.29-4.61
5	Los Alamos Scientific Laboratory	1958	0.53-120
6	Idaho Chemical Processing Plant [2]	1959	0.32-0.5
7	Mayak Production Association [2]	1960	0.0024-0.02
8	Idaho Chemical Processing Plant [2]	1961	none
9	Siberian Chemical Combine [2]	1961	2
10	Hanford Works	1962	0.19-1.1
11	Mayak Production Association [2]	1962	none
12	Siberian Chemical Combine [2]	1962	0.12
13	Siberian Chemical Combine [2]	1963	0.05
14	United Nuclear Fuels Recovery Plant [2]	1964	100
15	Electrostal Machine Building Plant [2]	1965	0.034
16	Mayak Production Association [2]	1965	0.001-0.0027
17	Mayak Production Association [2]	1968	0.0015-0.016
18	Windscale Works	1970	0.01-0.02
19	Idaho Chemical Processing Plant	1978	none
20	Siberian Chemical Combine	1978	0.05-20
21	Novosibirsk Chemical Concentration Plant [2]	1997	0.004
22	JCO Fuel Fabrication Plant [2]	1999	1-20

[1] numbers were taken from *A Review of Criticality Accidents* [86]

[2] multi-pulse accident [86]

Table 2.3: Lethal doses associated with whole-body exposure to ionizing radiation

dose (Gy)	time to death (days)
3-5	30-60
5-15	10-20
> 15	1-5

The total fission yield and radiation dose estimates shown in Table 2.1 and Table 2.2 are plotted and shown in Figure 2.7. The 1978 accident at the Idaho Chemical Processing Plant was removed from Figure 2.7 because it made the plot unreadable.

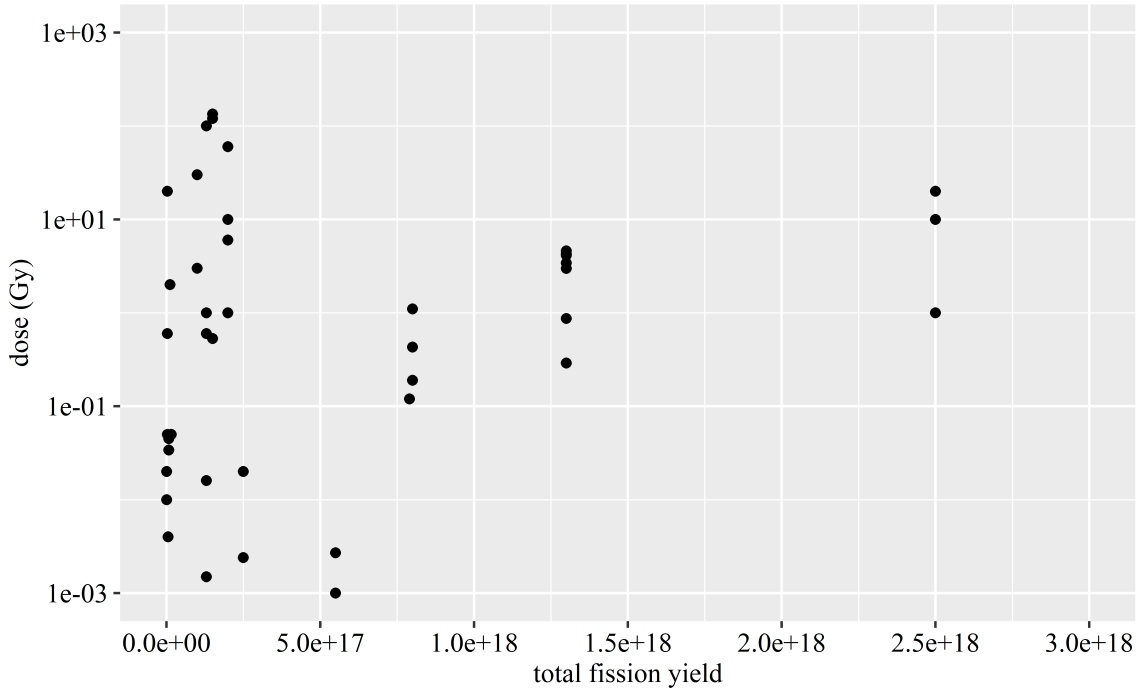

Figure 2.7: Total fission yield and radiation dose estimates for historical process criticality accidents

The results shown in Figure 2.7 are consistent with the following observations:

- The sharp spike in radiation doses at total fission yields $<$ 5e+17 is due to workers

who were standing within arm's reach of the accident when it occurred. The closer the worker was to the accident, the higher the radiation dose [86].

- There is a slight positive correlation between total fission yield and radiation dose, which is observable at total fission yields $>$ 1e+17 [86]. However, this correlation is overshadowed by other dominant factors, such as time, distance, and shielding [70].

2.5 Critical Experiments

Between the late 1960s and early 2000s, dozens of critical experiments were performed using homogeneous fissile solutions to study the radiation profiles of process criticality accidents [39]. Most of these experiments were performed at the Commissariat à l'énergie atomique et aux énergies alternatives (CEA) Valduc facility in Salives, France (CRAC and SILENE) [32], the Nuclear Safety Research Center in Tokai-mura, Japan (TRACY) [88, 89], the Pajarito Site in New Mexico (SHEBA) [39, 94, 119], and the Hanford Site in Washington [76, 95].

A report written by F. Barbry [32] summarizes the results of the CRAC and SILENE experiments and describes a maximum radiation dose of 5.8e+02 Gy at one meter from the source, which generated 1e+18 fissions in a 15-cm radius cylinder containing 80 g/L of uranyl nitrate in water. A similar report written by M. Murazaki et al. [88] summarizes the results of the TRACY experiments, which are shown in Table 2.4 and Table 2.5. The TRACY experiments were performed using a 77.2-cm radius cylinder containing approximately 375-380 g/L of uranyl nitrate in water, with and without 50 cm of water reflection [88]. Nuclear accident dosimeters were used to measure radiation doses at 2-4 meters from the

source [88]. To ensure that the results from M. Murazaki et al. [88] were consistent with other critical experiments, the integrated power (MJ) column from Table 1 of M. Murazaki et al. [88] was converted into total fission yield by multiplying each value by 6.242e+18 MJ/MeV and then dividing by 200 MeV, which is the assumed total energy released from each fission event [88].

Table 2.4: TRACY experiments (without reflection)

	dose (Gy)			
run number [1]	2 m	3 m	4 m	total fission yield
R287	12.31 ± 0.70	5.81 ± 0.32	3.46 ± 0.19	5.93e+17
R288	14.15 ± 0.81	7.06 ± 0.41		6.80e+17
R289	15.46 ± 0.89	7.85 ± 0.44	4.69 ± 0.27	7.71e+17
R290	16.29 ± 0.93	8.39 ± 0.49	5.39 ± 0.31	8.08e+17
R317	7.48 ± 0.43	4.08 ± 0.24	2.14 ± 0.12	3.53e+17
R318	7.82 ± 0.45	4.29 ± 0.25	2.31 ± 0.14	3.53e+17
R319	16.28 ± 0.95	9.18 ± 0.51	5.32 ± 0.30	8.43e+17
R321	14.99 ± 0.86	8.40 ± 0.49	4.85 ± 0.27	7.65e+17
R322	14.56 ± 0.84	8.27 ± 0.47	4.75 ± 0.27	7.68e+17
R324	6.96 ± 0.40	3.95 ± 0.22	2.10 ± 0.12	3.34e+17

[1] run numbers were taken from M. Murazaki et al. [88]

The results shown in Table 2.4 and Table 2.5 indicate that there is some correlation between total fission yield and radiation dose, similar to the results shown in Figure 2.7. However, these results also indicate a much stronger correlation between radiation dose and distance, which explains the differences that are shown in Figure 2.7. A report written by C.C. Cappiello et al. [39] summarizes the results of the SHEBA experiments, which were

Table 2.5: TRACY experiments (with 50 cm of water reflection)

	dose (Gy)			
run number [1]	2 m	3 m	4 m	total fission yield
R291	0.08 ± 0.01	0.06 ± 0.00	0.04 ± 0.00	1.75e+17
R293	0.10 ± 0.01	0.07 ± 0.00	0.05 ± 0.00	1.82e+17
R296	0.20 ± 0.01	0.15 ± 0.01	0.09 ± 0.01	3.53e+17
R325	0.08 ± 0.00	0.05 ± 0.00	0.03 ± 0.00	1.59e+17
R326	0.20 ± 0.01	0.14 ± 0.01	0.08 ± 0.00	3.56e+17
R327	0.37 ± 0.02	0.25 ± 0.01	0.15 ± 0.01	6.59e+17
R328	0.42 ± 0.02	0.29 ± 0.01	0.18 ± 0.01	7.33e+17

[1] run numbers were taken from M. Murazaki et al. [88]

performed using uranyl fluoride in water. A separate report written by D.G. Vasilik et al. [119] summarizes the results of four different SHEBA experiments, which used nuclear accident dosimeters to measure radiation doses at 3-4.6 meters from the source. The results of these experiments are shown in Table 2.6 [119].

Table 2.6: SHEBA experiments

number [1]	dose (Gy)	distance (m)	reflection	total fission yield
1	3.3	3	12 cm acrylic	1.6e+17
2	4.1	3	none	1.5e+17
3	8.1	3	20 cm concrete	2.9e+17
4	3.3	4.6	15 cm steel	3.0e+17

[1] numbers were taken from D.G. Vasilik et al. [119]

The results shown in Tables 2.4-2.6 indicate that reflection has a diametric effect on

total fission yield and radiation dose; if a reflector is placed near a critical or supercritical system, it tends to drive the total fission yield higher by reflecting neutrons back into the system—giving them a "second chance" to cause fission [74]. However, reflection also slows down (or thermalizes) neutrons, which reduces the radiation dose to nearby workers (Tables 2.4-2.6).

In another accident, workers involved in the 1958 accident at Y-12 promptly evacuated the facility after the criticality accident alarm system sounded, due to a single-pulse accident that took place in a 55-gallon drum that was being filled with uranyl nitrate and water [86]. Shortly after the accident occurred, solution exited the drum through a vent line, which temporarily drove the system subcritical [86]. However, the drum continued to fill with uranyl nitrate and water, which caused the system to pulse 2-3 more times until the solution was diluted with enough water to drive it permanently subcritical [86]. Despite the high total fission yield (1.3e+18 fissions), the subsequent alarm and prompt evacuation reduced the radiation dose to nearby workers and likely saved their lives [70, 86]. The radiation dose to the public is unknown, although there were reported readings of increased beta-gamma activity due to short-lived airborne fission products [1].

The first pulse fission yield and radiation dose estimates for 13 uranium solution accidents are plotted and shown in Figure 2.8, along with the results of the TRACY and SHEBA experiments [86, 88, 119]. The first pulse fission yield estimates are plotted in Figure 2.8 instead of the total fission yield estimates because they are more comparable to the TRACY and SHEBA experiments, which were all single-pulse experiments [86, 88, 119]. The 1978 accident at the Idaho Chemical Processing Plant was removed from Figure 2.8 because it made the plot unreadable. The Hanford Site experiments were not included due to lack of

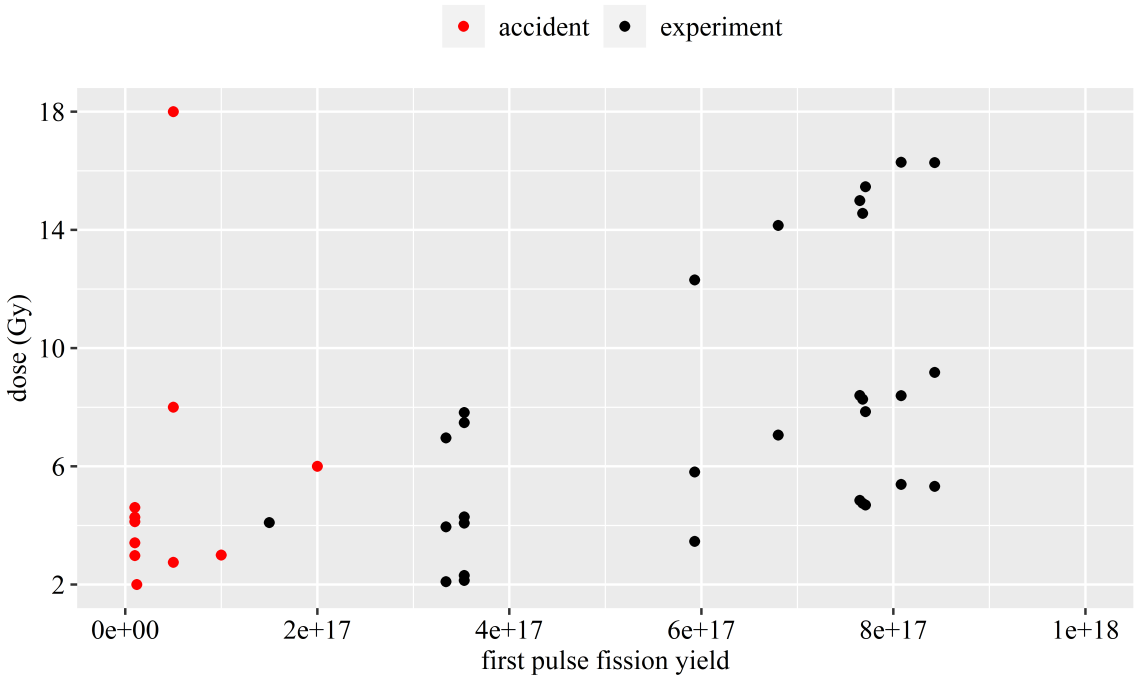

Figure 2.8: First pulse fission yield and radiation dose estimates for 13 uranium solution accidents and the TRACY and SHEBA experiments

data [76, 95].

The results shown in Figure 2.8 indicate that there is little to no overlap between total fission yield and radiation dose estimates for historical process criticality accidents and critical experiments. Since 2004, the Nuclear Safety Research Center in Tokai-mura, Japan is the only facility in the world that is capable of performing critical experiments with uranium solution [88, 89]. Critical experiment facilities at CEA Valduc in Salives, France [32], the Pajarito Site in New Mexico [39, 94, 119], and the Hanford Site in Washington [76, 95] have all been shut down and decommissioned. The lack of overlap between historical process criticality accidents and critical experiments and the inability to perform critical experiments with plutonium solution is a major gap in this area of research.

2.6 The Nuclear Criticality Slide Rule

To overcome the lack of critical experiments, researchers developed a set of graphs, known as the *nuclear criticality slide rule*, to give emergency responders "order of magnitude" estimates of total fission yield and radiation doses under various accident conditions [56, 68]. The two main references for the nuclear criticality slide rule were written by M. Duluc et al. [56] for plutonium accidents and C.M Hopper and B.L. Broadhead [68] for uranium accidents. These references provide comprehensive graphs and calculations that use standardized factors [11, 21] to convert neutron and gamma ray fluence calculations into radiation doses [56, 68]. These fluence-to-dose conversion factors are convenient because they take into account varying distances and reflector thicknesses, which can be scaled based on the total fission yield [56]. The fluence-to-dose conversion factors were taken from the following references:

[11] ANSI/HPS N13.3-2013, Dosimetry for Criticality Accidents. American National Standards Institute, Health Physics Society, 2013.

[21] ANSI/ANS-6.1.1-2020, Photon and Neutron Fluence-to-Dose Conversion Coefficients. American Nuclear Society, 2020.

M. Duluc et al. [56] also describes a two-step process for using MCNP to calculate the neutron and gamma ray fluence:

1. Perform a kcode calculation to simulate the energy and spatial distribution of neutrons.

2. Perform a separate sdef calculation to simulate neutron and gamma ray doses (with neutron multiplication turned off) [19].

Even though this is a straightforward process, sdef calculations take much longer to setup and run than kcode calculations. In this dissertation, each kcode calculation took approximately 45-60 seconds to run because all particles were terminated after they left the system boundary, which was specified as the void outside the one inch of water reflection (Figure 2.5). However, if a separate set of sdef calculations were performed, each MCNP input deck would need to be modified to expand the system boundary and simulate particle interactions further away from the source. Modeling Building 332 and performing these calculations would require significantly more computational power and time, which were beyond the scope of this dissertation. The main takeaway from this section is that the nuclear criticality slide rule exists, and it provides the capability to analyze the consequences of process criticality accidents.

2.7 Process Criticality Accident Risk Estimates

The following process criticality accident risk and total fission yield estimates were taken from DOE environmental impact statements:

Table 2.7: Process criticality accident risk estimates

site	risk (accidents/year)	total fission yield
Lawrence Livermore National Laboratory [7]	3.2e-05	1.0e+18
Los Alamos National Laboratory [4]	$<$ 1.0e-04	5.0e+17
Nevada National Security Site [13]	$\leq$ 1.0e-02	unknown
Pantex Plant [3]	$<$ 1.0e-06	unknown
Rocky Flats Plant [2]	8.8e-03	2.2e+20
Sandia National Laboratories [5]	$<$ 1.0e-07	unknown
Savannah River Site [22]	1.0e-02	5.0e+17
Y-12 National Security Complex [9]	$\leq$ 1.0e-02	3.28e+18

The radiation doses to nearby workers are not shown in Table 2.7 because they vary widely and appear to have no consistent distances or technical basis. The process criticality accident risk estimates shown in Table 2.7 were all calculated using fault tree analysis (FTA). FTA 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 [104].

One problem with using FTA to estimate process criticality accident risk is that there are an infinite number of failure paths, which makes it impossible to calculate the top-level failure probability [109]. Another problem, discussed in K.A. Reay and J.D. Andrews

[100] and R. Remenyte-Prescott and J.D. Andrews [102], is that fault trees can easily become large and incoherent. In E. Ruijters et al. [105], this is described as a "state-space explosion problem". The risk estimates shown in Table 2.7 circumvented these problems by only analyzing a subset of "worst case" or "bounding case" accident scenarios with unverifiable higher-than-normal probabilities [83, 125]. Although this approach may result in a conservative risk estimate, it completely ignores all other accident scenarios. There are at least two reports that challenge this approach, which were written by R.J. Mattson [83] and J.N. McKamy [85]. The following paragraphs were taken from R.J. Mattson's report on the Nuclear Criticality Safety Program at Rocky Flats [83]:

> "The Safety Analysis Report (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."

The SAR that is cited in R.J. Mattson's report [83] was written in 1987, and it estimated the risk of a criticality accident at Rocky Flats to be 3e-04 accidents per year [83], which is slightly lower than the value shown in Table 2.7. Despite these low risk estimates, several "near miss" criticality accidents occurred at Rocky Flats between 1987 and 1994 [83, 85]. The contributing factors from one such incident, which involved the transfer of plutonium solution from tanks to bottles, are paraphrased below.

Contributing factors to the 1993 near miss criticality accident at Rocky Flats [85]

- The presence of fissile solutions and uncharacterized holdup of fissile material increased the risk of a criticality accident.
- Fissile material operations were reliant on administrative controls to prevent a criticality accident.
- Line management tolerated recurrent, low-level operational failures.
- Line management perceived the risk of a criticality accident to be trivial.
- Line management was focused on other issues (e.g., performance incentives, radiological contamination, environmental remediation).
- The absence of independent safety oversight led to a lack of review of operational failures.

These contributing factors also draw parallels with incidents that have occurred at other DOE sites, including Los Alamos National Laboratory [110, 121, 122] and the Y-12 National Security Complex [28, 60, 62, 63, 84]. One such incident at Y-12 involved the unintended accumulation of 2-3 kilograms of uranium in a sand separator, which led to the discovery of several additional kilograms of accumulated uranium throughout the facility, all of which violated existing criticality controls [84]. These incidents are important to consider because they often take place over a long period of time, which challenges the assumption that DOE sites operate safely, except for isolated incidents that are due to worker error [125].

Each DOE site is required to maintain a site-wide environmental impact statement (EIS) [10], which summarizes analyses that are performed to estimate the likelihood and consequences (i.e., risk) of various accidents, including criticality accidents. The following excerpts were taken from 40 CFR § 1502 [10], which are the governing rules and regulations for environmental impact statements:

> **§ 1502.3 Statutory requirements for statements**
>
> As required by section 102(2)(C) of NEPA, environmental impact statements are to be included in every Federal agency recommendation or report on proposals for legislation and other major Federal actions significantly affecting the quality of the human environment.

> **§ 1502.23 Methodology and scientific accuracy**
>
> Agencies shall ensure the professional integrity, including scientific integrity, of the discussions and analyses in the environmental documents. Agencies shall make use of reliable existing data and resources. Agencies may make use of

> any reliable data sources, such as remotely gathered information or statistical models. They shall identify any methodologies used and shall make explicit reference to the scientific and other sources relied upon for conclusions in the statement. Agencies may place discussion of methodology in an appendix. Agencies are not required to undertake new scientific and technical research to inform their analyses. Nothing in this section is intended to prohibit agencies from compliance with the requirements of other statutes pertaining to scientific and technical research.

40 CFR § 1502 [10] allows for a sort of broken logic to be applied to environmental impact statements [6], in that it requires an environmental impact statement to be prepared before an application is accepted—without requiring DOE to compare "reliable existing data and resources" to actual operations. 10 CFR § 830 Subpart B [8] and DOE-STD-3009 [15] attempt to close this break by requiring each DOE site to prepare a documented safety analysis for each facility, compare "current site characteristics with existing environmental analyses and impact statements", and "state that no significant discrepancies exist, or indicate a need to revise and update existing environmental documentation". Despite this requirement, none of the process criticality accident risk estimates included in original EIS submissions have ever been updated, unless there was a significant modification to the facility (Table 2.7). Although the exact reasons for this are unknown, one possible explanation is that FTA is difficult, time-consuming, and expensive to prepare [104]. Another possible explanation is that FTA relies on unverifiable assumptions of "optimal criticality conditions" [83, 125], which decouple FTA from the underlying parameters that affect nuclear criticality, thereby

preventing analysts and engineers from being able to compare FTA to actual operations.

The methodology described in this dissertation is also time-consuming to prepare, due to the large number of MCNP calculations that are needed. However, once these calculations are performed, they enable facility data and risk estimates to be updated easily and cheaply. This is similar to the approach described in S. Rebello et al. [101], which uses a coupled Bayesian network and hidden Markov model to estimate optimal safety thresholds for complex industrial processes.

2.8 Bayesian Networks

The complexity of fissile material operations led to the selection of a Bayesian network as the first-half of the methodology described in this dissertation. The term *Bayesian network* was coined by Judea Pearl in 1985 [96] and later expanded upon in his book, *Probabilistic Reasoning in Intelligent Systems* [97]. The following definition was taken from the original article [96]:

> "Bayesian networks are directed acyclic graphs in which the nodes represent propositions (or variables), the arcs signify the existence of direct causal dependencies between the linked propositions, and the strengths of these dependencies are quantified by conditional probabilities."

The methodology described in this dissertation is based on F. Cadini and A. Gioletta [37, 38], which use Bayesian networks and metamodels to estimate small failure probabilities. The Bayesian network shown in Figure 2.9 is comprised of 8 nodes and 13 arcs, which represent fissile material operations (*op*), criticality controls (*ctrl*), and parameters that affect

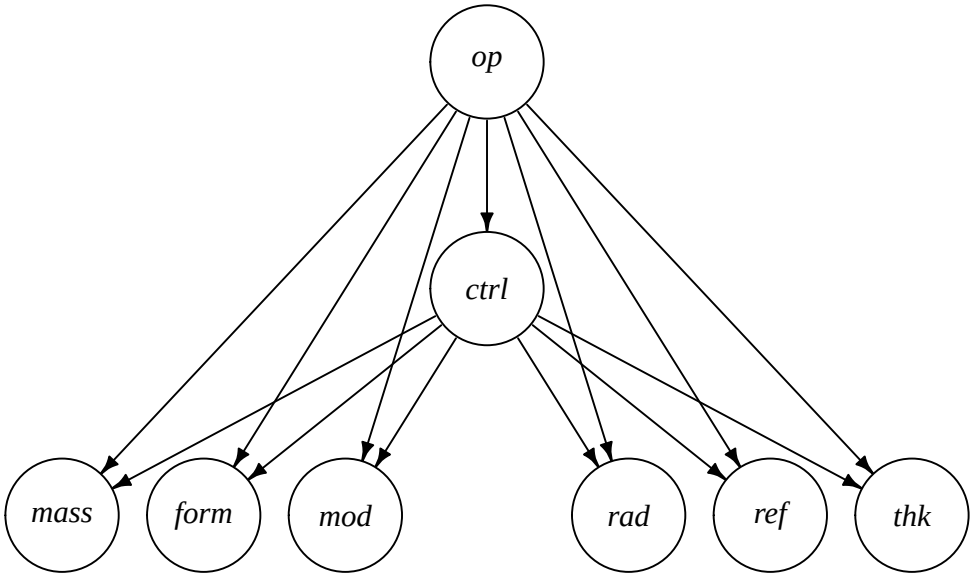

Figure 2.9: Bayesian network

nuclear criticality. This Bayesian network (Figure 2.9) is modeled using a combination of discrete and continuous probability distributions that were converted into discrete probability tables.

The arcs in a Bayesian network 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

$p(x_B)$ 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 (2.9) 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 2.9)

The joint probability distribution can also be explicitly expressed as a non-negative function, or *evidence potential* (2.15) [80]. The evidence potential defines the structure of the *moral graph* [50], which is a type of undirected graph that is used by the *junction tree clustering algorithm* (Algorithm 1) to perform exact inference [108]. The moral graph of the Bayesian network (Figure 2.9) is shown in Figure 2.10.

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

where:

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

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

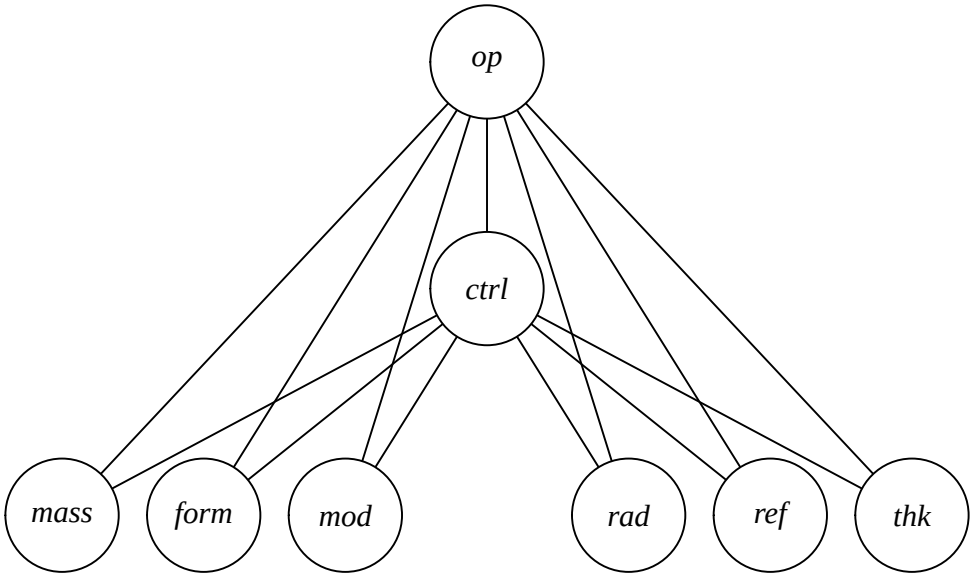

Figure 2.10: Moral graph

The Bayesian network is graphically decomposed using the *junction tree clustering algorithm* 1 [108] so that exact inference can be performed. The moral graph (Figure 2.10) is already triangulated [80], which means there are no cycles with more than three nodes, thus satisfying Algorithm 1. Visually, the only difference between the moral graph and the Bayesian network is the lack of arrows in the moral graph. The next two steps in Algorithm 1 are to identify cliques (Table 2.8) and build a junction tree (Figure 2.11). The cliques (Table 2.8) were selected to minimize the number of arcs in the junction tree (Figure 2.11). By following Algorithm 1, the *clique potential representation* of $p(x)$ can be calculated using 2.16 [80].

Moralize: Build the moral graph of the Bayesian network ($\mathcal{B}$).
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

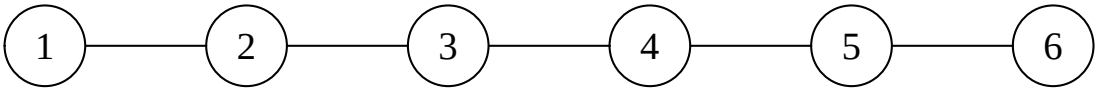

Figure 2.11: Junction tree

Table 2.8: 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

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

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 $R_6 \perp\!\!\!\perp (C_1 \cup ... \cup C_5)/S_6 | S_6$ [80]. Marginalizing over R_6 results in 2.17 [80].

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

Now, let $\psi(S_i) = \sum_{R_i} \psi(S_i, R_i)$. Combining this term with 2.17 results in 2.18 [80].

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

Combining $\psi(S_i) = \sum_{R_i} \psi(S_i, R_i)$ with 2.17 and 2.18 results in 2.19 [80].

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

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 [80].

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

The resulting expression (2.20) is called a *set chain representation* [80]. 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 [80].

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

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) [80].

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

The marginal probabilities of each node in the junction tree can be calculated using 2.22 [80]. This allows the Bayesian network to perform exact inference by propagating evidence through 2.16-2.22, which leads to a clique marginal representation in which the potentials become $\psi(C_i) = p(C_i|E^*)$ [80]. The *gRain* [66] software package was used to perform exact inference [67]. Example code and output are shown in Figure 2.12. The *cpdist* function in the *bnlearn* [107] software package was used to generate samples using the forward sampling algorithm (Algorithm 2) [75, 108]. Example code and output are shown in Figure 2.13.

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: Forward sampling algorithm

```
>
> 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
>
> setFinding(grain.bn, nodes = 'mass', states = '500') %>% pFinding()
[1] 2.611831e-08
>
```

Figure 2.12: Output from using the *gRain* software package to calculate the probability that plutonium mass = 500 g

```
>
> 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
>
```

Figure 2.13: Output from using the *bnlearn* software package to forward sample Bayesian network parameters

There are several benefits of using a Bayesian network to estimate process criticality accident risk:

- Bayesian networks are capable of modeling multiple causes of failure [36, 117].
- Bayesian networks are capable of performing forward (or predictive) and backwards (or diagnostic) analysis with uncertainties [36].
- Bayesian networks provide information about the occurrence and non-occurrence of events [72]. N. Khakzad et al. [72] discusses the benefit of using a Bayesian network to determine "the most probable configuration" of an accident, given its occurrence.
- Bayesian networks are capable of updating probabilities and propagating evidence throughout the model [36, 72, 96, 97, 108].

In addition to the benefits described above, the mechanisms leading to top-level failure do not need to be well understood prior to building the Bayesian network [72, 108]. The only things that are needed are an understanding of the parameters that affect nuclear criticality (Table 1.1) and the ability to model each one as a set of discrete or continuous probability tables [96, 97, 108].

One drawback of using a Bayesian network is that its ability to predict latent variables (e.g., k_{eff}) is constrained by the size and complexity of the model [75, 96, 97, 117]. Existing methods, such as FORM and SORM, use a Taylor series expansion to approximate the “most probable failure point” [72], while others use Monte Carlo variance reduction techniques to select samples that are in the "vicinity of failure" [37]. 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 [37].

2.8.1 Rare Event Modeling

Rare or *extreme events* are events that occur with low frequency and encompass natural phenomena (e.g., earthquakes, tsunamis, floods) and anthropogenic hazards (e.g., industrial accidents, stock market crashes, terrorism), as well as phenomena for which natural and anthropogenic factors interact in complex ways [25]. Rare events are related to the *law of truly large numbers*, which states that "when enormous numbers of events and people and their interactions cumulate over time, almost any outrageous event is bound to occur" [53]. The "outrageous event" in this dissertation is a process criticality accident, and the "enormous numbers of events" are the samples generated by the Bayesian network.

Fortunately, certain assumptions can be made about process criticality accidents, which limit the number of samples that are needed to estimate process criticality accident risk. In *Anomalies of Nuclear Criticality* [45], E.D. Clayton discusses several "small mass" concepts for systems containing ^{239}Pu or ^{235}U. These concepts are based on calculations performed by K.R. Yates [123] and R.S. Olson and M.A. Robkin [92], which sought to determine if there is a theoretical *minimum critical mass* for systems containing ^{239}Pu or ^{235}U.

An article written by W.J. Zywiec and A.J. Nelson [124] corrects several errors in K.R. Yates [123] and R.S. Olson and M.A. Robkin [92] and describes a theoretical minimum critical mass of approximately 88 grams of ^{239}Pu. 88 grams of ^{239}Pu is a useful starting point in this dissertation because it can be used to ignore or throw away samples with plutonium masses $<$ 88 grams. The ability to throw away samples provides a significant computational

advantage because it can be used to formulate conditional probability queries [108], which reduce the time it takes to predict k_{eff} values and estimate process criticality accident risk. The technical basis and implementation of these conditional probability queries are discussed further in Chapter 3.

2.9 Metamodels

The metamodel described in this dissertation is based on articles written by F. Cadini [38], V. Dubourg et al. [54], and M. Papadrakakis and N.D. Lagaros [93]. These articles describe how metamodels can be used to eliminate or reduce the computational burden of performing direct, simulation-based analysis of physical systems. The purpose of using a metamodel in this dissertation was to leverage approximately 1.5 million MCNP calculations to predict k_{eff} values for a bounded, but much more refined, granular domain.

There are several different types of metamodels that can be used to predict k_{eff}, including Gaussian process regression models [54, 99, 111], neural networks [59, 65], polynomial chaos expansions [35, 91, 112], quadratic response surfaces [29, 64], random forests [78, 81], and support vector machines [34, 106, 113]. Neural networks were selected for this application because they are optimized to perform vector- and matrix-based calculations on a graphics processing unit (GPU) much faster than on a central processing unit (CPU) [27, 43].

The *caret* [77] software package was initially used to test several different types of metamodels, including Gaussian process regression models [54, 99, 111], neural networks [59, 65], random forests [78, 81], and support vector machines [34, 106, 113]. However,

caret [40] was not designed to run on a GPU, which limited its capabilities and made it much less attractive than other software packages, such as *Keras* [43] and *TensorFlow* [27]. The *caret* [77] software package was also used to create dummy variables, which are discussed further in Chapter 3.

The *Keras* [43] and *TensorFlow* [27] software packages were used to build and train neural networks. *Keras* is a high-level application programming interface (API), which runs on top of *TensorFlow* and manages interactions between Python, R, and the *TensorFlow* platform [43]. The core data structures of *Keras* are layers and models [43]. An example of a sequential model with six hidden layers and one output layer is shown in Figure 2.14.

```
>
> model <- keras_model_sequential() %>%
+ layer_dense(units = layers[1], activation = 'relu', input_shape =
  ↪ dim(dataset$training.df)[2]) %>%
+ layer_dense(units = layers[2], activation = 'relu') %>%
+ layer_dense(units = layers[3], activation = 'relu') %>%
+ layer_dense(units = layers[4], activation = 'relu') %>%
+ layer_dense(units = layers[5], activation = 'relu') %>%
+ layer_dense(units = layers[6], activation = 'relu') %>%
+ layer_dense(units = 1, activation = 'linear')
>
```

Figure 2.14: *Keras* sequential model

TensorFlow is an end-to-end, open-source machine learning platform, which combines four key abilities [27]:

- Efficiently executing low-level tensor operations on CPUs, GPUs, and tensor processing units (TPUs)
- Computing the gradient of arbitrary differentiable expressions
- Scaling computation to many devices

- Exporting programs to external servers, browsers, mobile, and embedded devices

The *Keras* [43] and *TensorFlow* [27] software packages were selected because they are well documented and compatible with R, a programming language that has several graphical and statistical software packages that were used throughout this dissertation. These software packages include *igraph* [48], *fitdistrplus* [51], *gRain* [66], *bnlearn* [107], and *ggplot2* [120].

2.9.1 Neural Networks

A *neural network* is a connected system of nodes and directed arcs, similar to a Bayesian network. The key differences between Bayesian networks and neural networks are that each node (or *neuron*) 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.

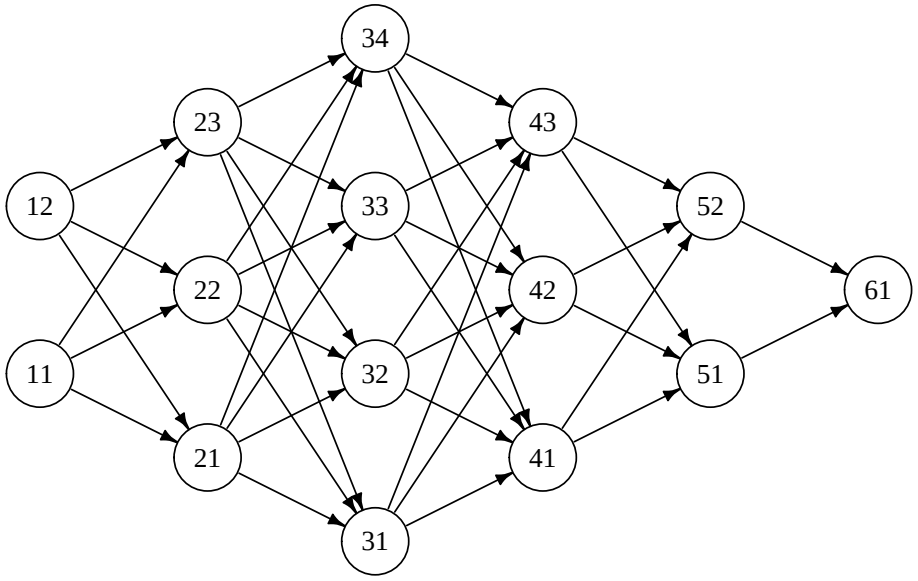

Figure 2.15: Neural network with four hidden layers

A *deep feedforward neural network* is a neural network with one or more input layers,

two or more hidden layers, and one or more output layers (Figure 2.15). A deep feedforward neural network is comprised of many different functions, which approximate parameters (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 2.15, is calculated using 2.23.

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

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 hidden layers 2-4 (Figure 2.15) are calculated using 2.24.

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

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

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

In each layer, there is a hidden bias neuron that takes no input and provides some default output when all inputs equal zero [59]. The output of the last layer, shown in Figure 2.15, 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 [59].

$$
\begin{bmatrix} x_{61} \end{bmatrix} = \begin{bmatrix} W_{50\to61}x_{50} + W_{51\to61}x_{51} + W_{52\to61}x_{52} \end{bmatrix} + b_5 \tag{2.25}
$$

Propagating input through 2.23-2.25 results in one forward pass of data through the neural network. The vector output from 2.25 is then passed to a loss function, which, for regression

problems, is typically the mean squared error (MSE) (2.26) [44, 59].

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

where:

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

$x =$ vector of observations or predictions

In this dissertation, each neural network layer was densely (or fully) connected, and each hidden neuron used a rectified linear unit activation function, which is a piecewise function that returns x if $x > 0$ and 0 if $x \leq 0$ (Figure 2.16).

Figure 2.16: Rectified linear unit (ReLU) activation function

A simplified equation for a deep feedforward neural network with four hidden layers is shown in 2.27 [59].

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

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 loss function. Substituting $x_{predicted_i}$ into the right side of 2.27 yields 2.28.

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

Output from the MSE loss function (2.28) is passed to the optimization algorithm, which updates parameters (θ) by implementing the backpropagation algorithm. The backpropagation algorithm is the backbone of a neural network and applies the chain rule to compute the gradients (i.e., the vector of partial derivatives with respect to θ evaluated at time step i) and update parameters (θ) [59]. These parameters (θ) are also referred to as the weights (W) and biases (b) of the neural network (2.23-2.27).

The Adamax optimization algorithm (Algorithm 3) was used to update parameters (θ) throughout this dissertation [73]. Adamax is a variant of the Adam optimization algorithm [73] based on the *exponentially weighted infinity norm* (u), which takes the place of the bias-corrected first and second moment estimates.

Require: $\alpha =$ 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: $\theta_0 =$ 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)
    $u_i \leftarrow max(\beta_2 \cdot u_{i-1}, |g_i|)$ (update exponentially weighted infinity norm)
    $\theta_i \leftarrow \theta_{i-1} - (\alpha/(1-\beta_1^i)) \cdot m_i/u_i$ (update parameters)
return θ_i

Algorithm 3: Adamax optimization algorithm [73]

Chapter 3: Methodology

This chapter discusses the methodology that was used to build a coupled Bayesian network and neural network metamodel and estimate process criticality accident risk.

3.1 Bayesian Network

This section describes how the Bayesian network was built—with a focus on fitting probability distributions to facility data and analyzing the goodness of fit. The Bayesian network shown in Figure 3.1 was built to model fissile material operations (*op*), criticality controls (*ctrl*), and the parameters that affect nuclear criticality in Building 332—although it can be readily applied to any nuclear or radiological facility. The structure of the Bayesian network was based on each fissile material operation (*op*) having one to five sets of conditionally dependent criticality controls (*ctrl*) and six conditionally dependent parameters that affect nuclear criticality (*mass*, *form*, *mod*, *ref*, *thk*, *shape*). The *bnlearn* [107] and *igraph* [48] software packages were used to build and visualize the Bayesian network shown in Figure 3.1.

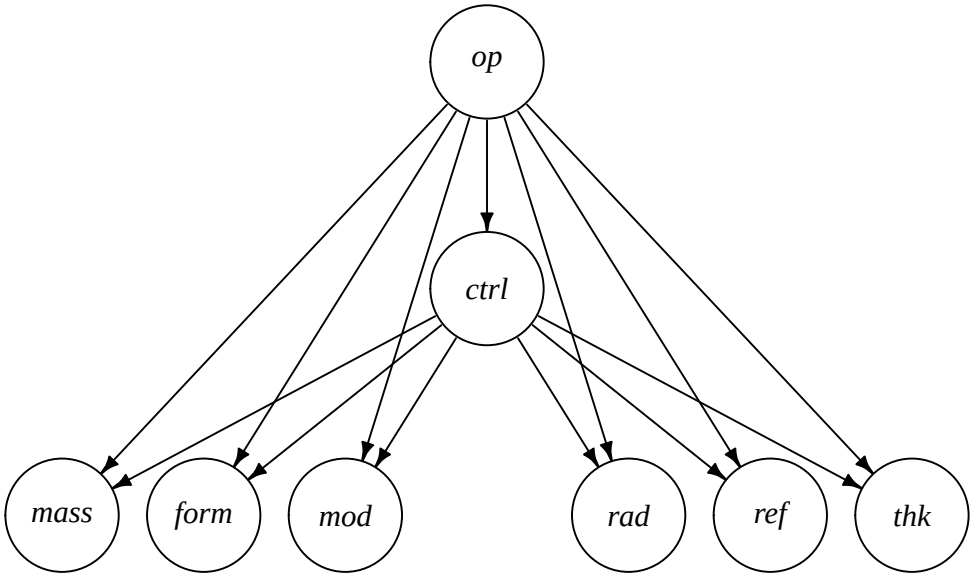

Figure 3.1: Bayesian network

Fissile material operations (*op*) are divided into six categories based on work control documents that describe individual operations—and the controls that apply to them (Table 3.1). Work control documents are stored in an online database at Lawrence Livermore National Laboratory, and each one is written and administered by a responsible manager, a work planner, and a team of multi-disciplinary safety professionals [23]. Criticality controls in Building 332 are developed and approved by nuclear criticality safety engineers for each fissile material operation and work control document. Each set of controls in Building 332 has a single letter designation (e.g., *Condition A*, *Condition B*, *Condition C*) and consists of administrative limits on mass, form, moderation, and reflection (Table 3.3).

Most fissile material operations have two to five sets of criticality controls. However, only one set of criticality controls can be active at each workstation (i.e., a glovebox, table, or other area with clearly marked boundaries). For example, a worker can use *Condition A* at one workstation and then switch to *Condition E* if the operation requires them to move > 65

Table 3.1: Fissile material operations in Building 332

operation	description
large sample	operations with ≥ 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: Probability table for fissile 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	≤ 65 g		no D_2O	
B	≤ 220 g		H/X $\leq H_2O$, ≤ 4 L	no Be/C/D_2O, ≤ 2 in
C	≤ 1200 g		≤ 2.5 L	none
D	≤ 2500 g		≤ 1 L	≤ 2 L, ≤ 0.25 in
E	≤ 2500 g		none	≤ 0.25 in
M	≤ 4000 g	≤ 500 g fines	≤ 1 in, ≤ 4 L	≤ 0.3 in
P	≤ 300 g		no liquids	≤ 3.5 kg Be/C

grams of plutonium into the workstation. To facilitate this move, the worker ensures 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 (Table 3.4) is derived from logbook entries and other paper records, which nuclear criticality safety engineers use to document active criticality controls at each workstation.

Table 3.4: 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

The parameters that affect nuclear criticality in Building 332 are shown in Table 3.5. These parameters are conditionally dependent on fissile material operations (Table 3.1) and criticality controls (Table 3.4). Spherical geometry and surrogate materials were selected to conservatively bound the physics of process criticality accidents (Table 3.5). Spherical geometry was used because a sphere has the lowest surface area-to-volume ratio of any object, which reduces neutron leakage and increases k_{eff} [74]. High-density polyethylene ($\rho = 0.965$ g/cm^3) was used instead of liquids and other hydrogenous materials because it provides superior moderation and reflection, which also increases k_{eff} [45, 74].

Magnesium oxide was included in Table 3.5 because it is commonly found in crucibles.

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)

Sepiolite was included in Table 3.5 because it is used to absorb liquids in waste and down-blend plutonium oxide (i.e., render it useless for weapons manufacture). Other materials, such as aluminum, beryllium, depleted uranium (^{238}U), graphite (C), SAE 304 stainless steel (SS304), and water were included because they are either used in experiments or are integral to gloveboxes and process equipment.

3.1.1 Fitting Probability Distributions to Facility Data

The probability tables for operations (*op*), controls (*ctrl*), *form*, moderation (*mod*), and reflection (*ref*) were populated directly from computer and paper records, which contain information on fissile and non-fissile materials that are involved in operations. The probability tables for *mass*, radius (*rad*), and reflector thickness (*thk*), however, were based on truncated probability distributions (3.1-3.4) that were fit to facility data using the maximum likelihood estimation method [49]. Gamma and normal distributions were selected because they are

commonly used distributions in the field of risk analysis, and they fit the data reasonably well (Table 3.6). Log-normal and Weibull distributions were selected because they are heavy right-tailed distributions, which increases the probability of generating samples that result in a criticality accident. The probability density functions of these distributions are shown in 3.1-3.4 [52].

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

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

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

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

The maximum likelihood estimators are derived from the likelihood function (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 facility data, they were trucated by selecting

a conservatively large upper limit for each variable, removing values beyond each limit, and then normalizing the remaining values so that each row in the conditional probability table sums to one. Kolmogorov-Smirnov [82], Cramér-von Mises, [47], and Anderson-Darling [30] tests were also performed to determine the goodness of fit for each truncated probability distribution. The general formulas for these test statistics are shown in 3.6-3.8 [30, 47, 82] 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* [51] software package and compared to critical values that were taken from R.B. D'Agostino and M.A. Stephens [49].

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

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

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

An example set of truncated gamma and log-normal distributions and quantile-quantile (Q-Q) plots for the 65-gram *Condition A* mass limit are shown in Figures 3.2-3.5. Histograms of Kolmogorov-Smirnov [82], Cramér-von Mises, [47], and Anderson-Darling [30] test statistics are shown in Figure 3.6 and Table 3.6. The critical values of these test statistics correspond to a significance level of $\alpha = 0.05$ with $n = 500$ samples [49].

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

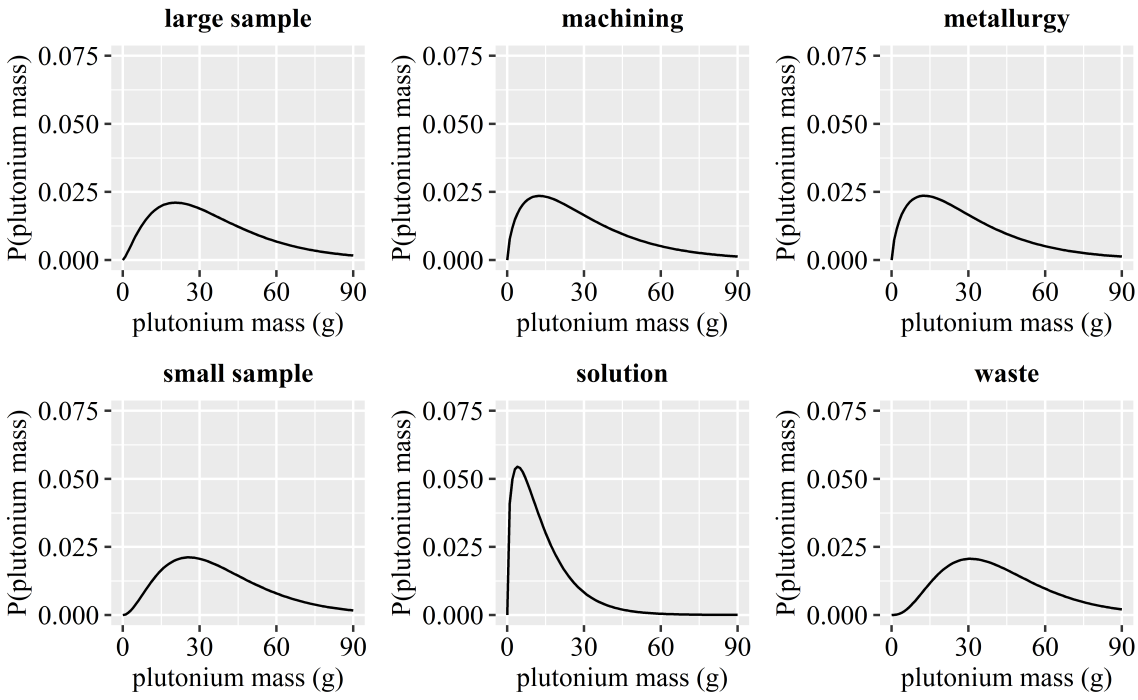

Figure 3.2: Truncated gamma distributions for the 65-g mass limit

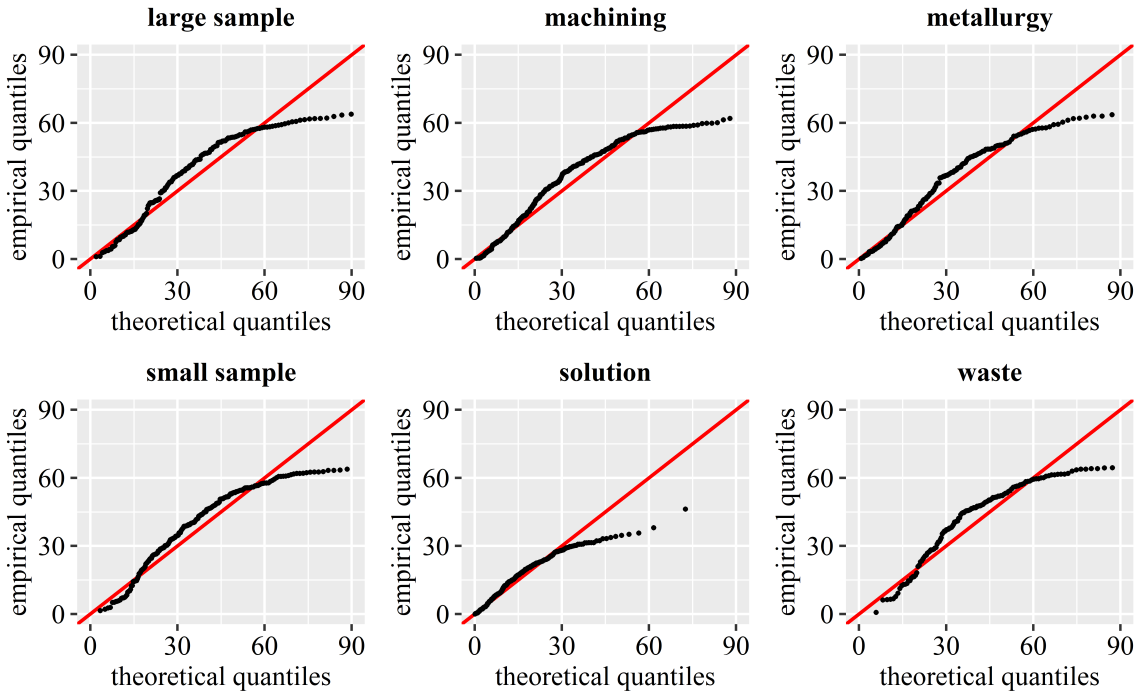

Figure 3.3: Q-Q plots of truncated gamma distributions for the 65-g mass limit

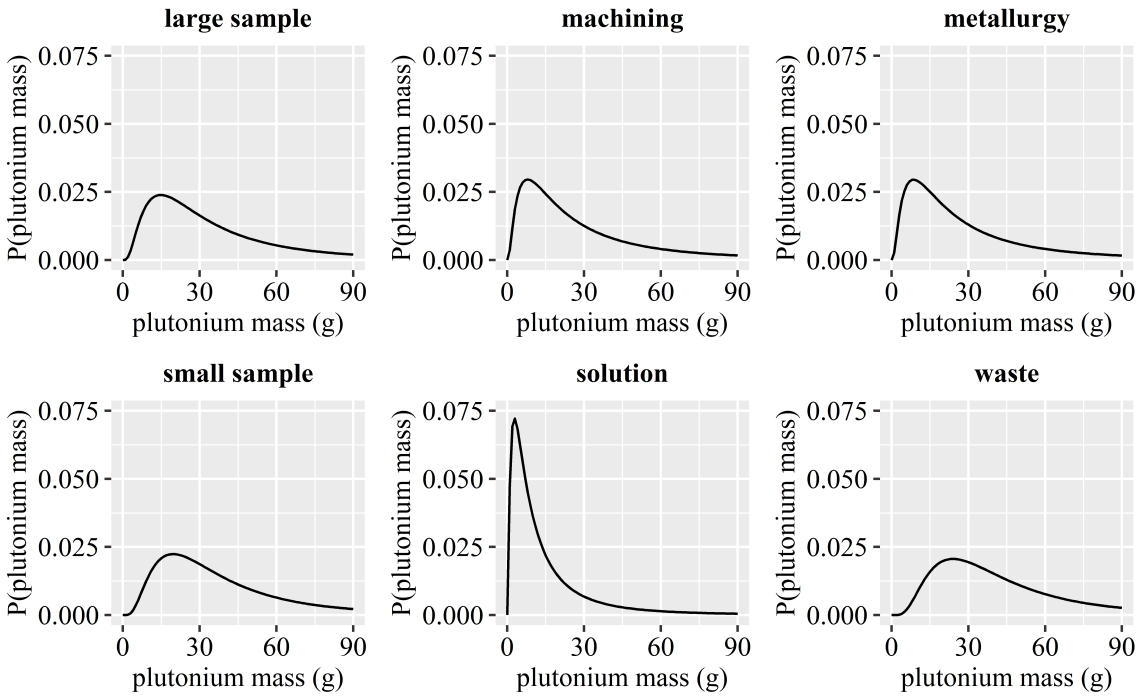

Figure 3.4: Truncated log-normal distributions for the 65-g mass limit

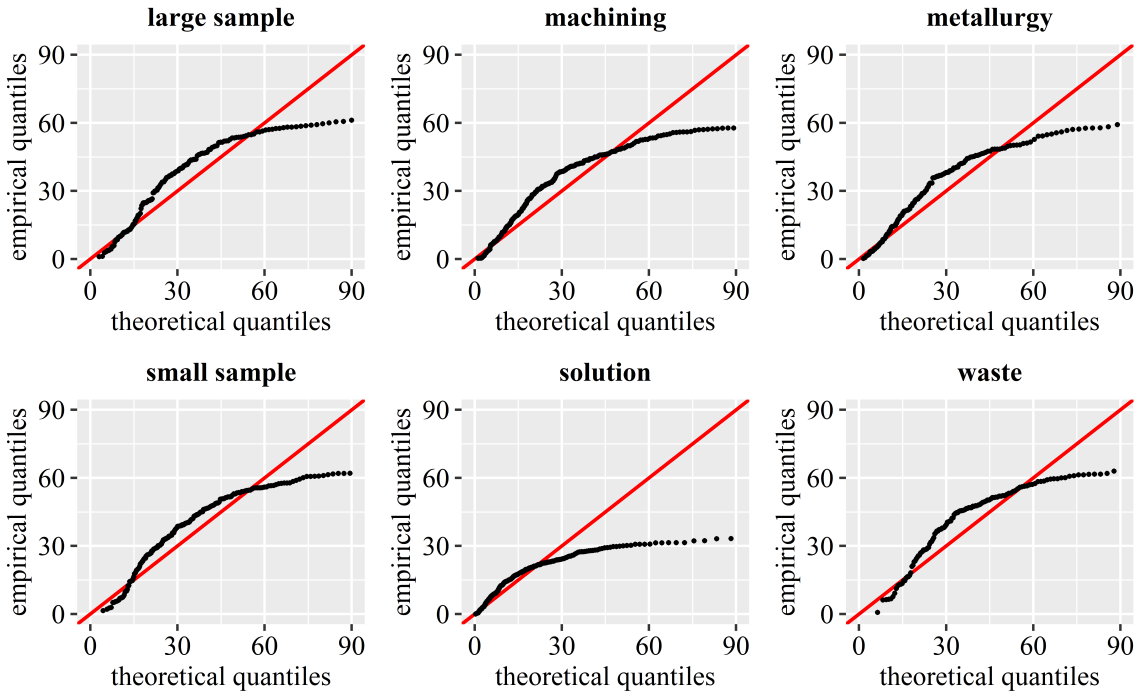

Figure 3.5: Q-Q plots of truncated log-normal distributions for the 65-g mass limit

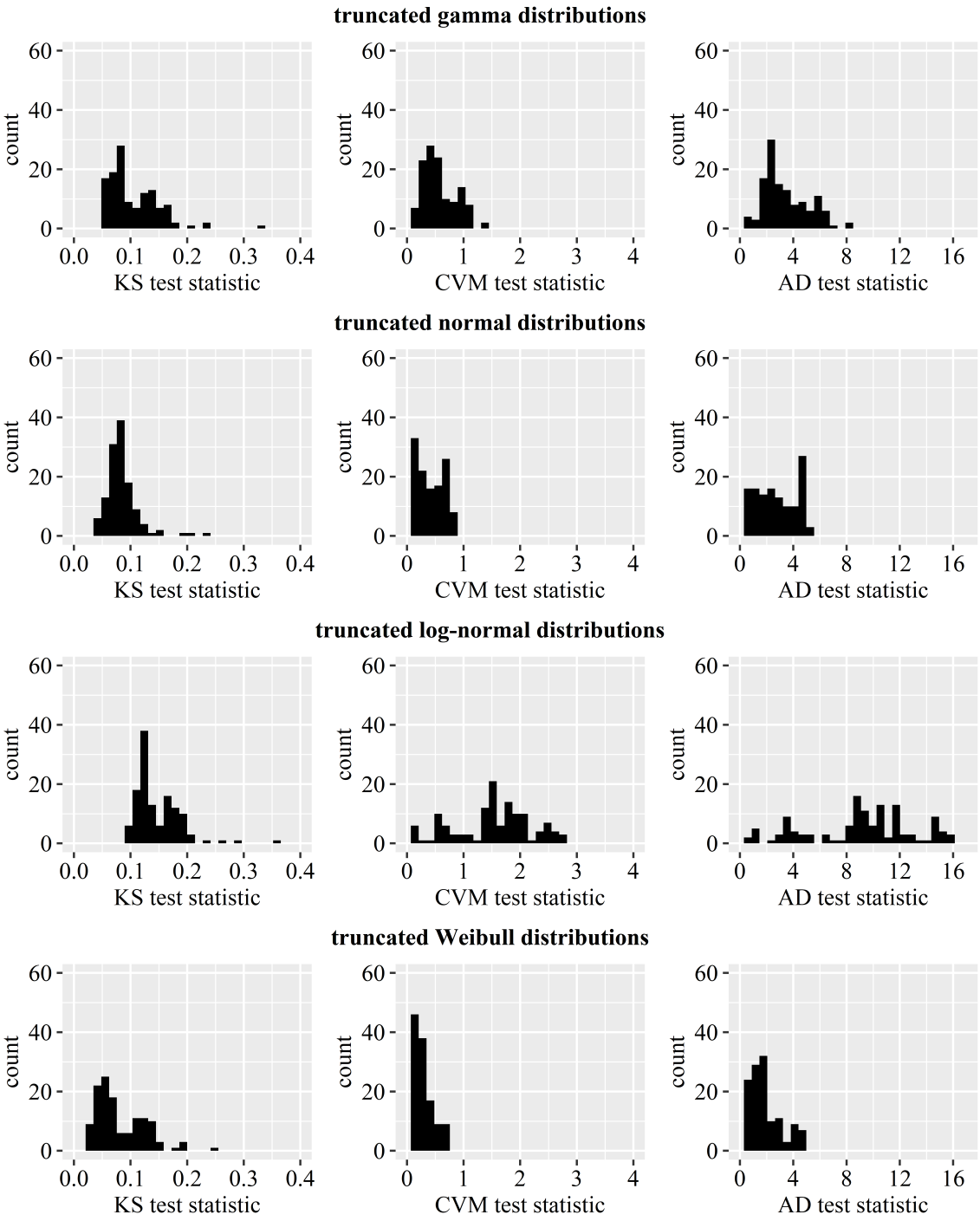

Figure 3.6: Histograms of test statistics

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	

distributions—although this is by no means a definitive assessment of each fit. The purpose of calculating these test statistics was to document the goodness of fit, which is commonly used in conjunction with graphical checks of Q-Q plots (Figure 3.3, Figure 3.5) to validate risk models in the nuclear industry [31].

3.1.2 Forward Sampling Algorithm

The forward sampling algorithm discussed in Chapter 2 (Algorithm 2) [75, 108] was used to generate samples, which were passed to the neural network metamodel. Conditional probability queries were initially formulated for plutonium masses ≥ 88 grams, based on an article written by W.J. Zywiec and A.J. Nelson [124]. These conditional probability queries were subsequently modified to increase the plutonium mass to ≥ 120 grams with no changes in risk. The conditional probability queries and their impact on process criticality accident risk estimates are discussed further in Chapter 4.

3.2 Preparing Data

A total of 1,542,792 MCNP calculations were performed as part of the effort to model fissile material operations in Building 332. These calculations were based on a sparse set of training parameters that represent approximately 2.5e-05% of the domain that the Bayesian network is capable of sampling (see Section 3.1 and Appendix A.1.4).

3.2.1 Running MCNP

The MCNP input deck from Chapter 1 (Figure 2.5) is shown in Figure 3.7, along with a portion of its corresponding output file (Figure 3.8). MCNP output files are divided into three columns of k_{eff} estimators, which are averaged for the "first half" and "second half" of cycles 50-500 (Figure 3.8) [19]. The k_{eff} values that were extracted from these output files are reported as the "final result" (Figure 3.8).

The increments and upper limits of the training parameters, shown in Table 3.7, were selected based on the rationale discussed in the previous section. The *Grid* (Appendix A.1.1) and *Build* (Appendix A.1.1.1) functions were used to build the MCNP input decks. A separate set of Python and Slurm [69] scripts were used to run MCNP calculations on the Quartz supercomputer at Lawrence Livermore National Laboratory (Figure 3.9). Once the MCNP input decks were run, output data was extracted, transformed, loaded, and written to CSV and RData files using the *Tabulate* (Appendix A.1.2) and *Scale* (Appendix A.1.2.1) functions.

```
500 alpha h2o 13.97 none sph $ <mass> <form> <mod> <rad> <ref> <shape>
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                       $ thermal scattering treatment for H2O
m2   1001.80c +2 $ H-1
     8016.80c +1 $ O-16
mt2  lwtr.20t    $ thermal scattering treatment for H2O
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 3.7: MCNP input deck

```
keff   neutron
cycle  histories   k(col)   sd        k(abs)   sd        k(tl)    sd
-------------------------------------------------------------------------
...          ... | ...      ...       ...      ...       ...      ...
-------------------------------------------------------------------------
491         9563 | 0.86871  0.00065   0.86936  0.00045   0.86867  0.00066
492         9924 | 0.86868  0.00065   0.86936  0.00045   0.86863  0.00066
493        10105 | 0.86874  0.00065   0.86936  0.00044   0.86869  0.00066
494        10508 | 0.86874  0.00065   0.86938  0.00044   0.86868  0.00066
495         9733 | 0.86875  0.00064   0.86934  0.00044   0.86869  0.00065
496        10169 | 0.86877  0.00064   0.86934  0.00044   0.86873  0.00065
497        10097 | 0.86880  0.00064   0.86935  0.00044   0.86875  0.00065
498        10031 | 0.86884  0.00064   0.86937  0.00044   0.86878  0.00065
499        10113 | 0.86882  0.00064   0.86939  0.00044   0.86877  0.00065
500         9630 | 0.86880  0.00064   0.86937  0.00044   0.86874  0.00065
-------------------------------------------------------------------------
problem       keff     sd
-------------------------------------------------------------------------
first half    0.86893  0.00060
second half   0.86947  0.00062
final result  0.86926  0.00043
```

Figure 3.8: MCNP output file

Table 3.7: Training parameters

parameter	description
mass	0-4 kilograms of Pu in 25-gram 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 (g/cm^3)
k_{eff}	effective neutron multiplication factor

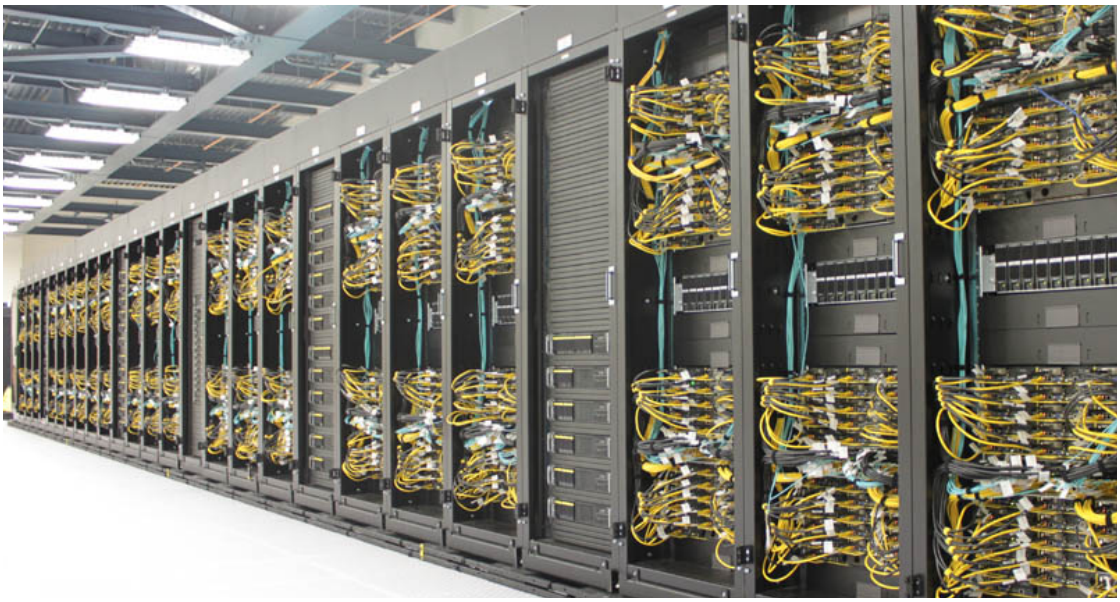

Figure 3.9: Quartz supercomputer at Lawrence Livermore National Laboratory

MCNP output data was initially converted from a CSV file into a data frame by removing rows with "N/A" values and rerunning calculations that terminated early, due to scheduled hardware maintenance or runtime errors. Then, the data frame was visualized using *ggplot2* [120] to look for unusually large standard deviations (Figure 3.10, Figure 3.11). Of the 1,542,792 MCNP output files that were generated, 395 had standard deviations (σ) > 0.001 (Figure 3.10). The mean MCNP standard deviations (σ) from Figure 3.10 and Figure 3.11 are 2.70e-04 and 3.51e-03, respectively.

The 395 MCNP output files with standard deviations (σ) > 0.001 (Figure 3.10) were all found to have low plutonium masses (25-125 grams), large volumes ($\geq$ 68.6 L), and thick depleted uranium reflectors. This combination of parameters is unusual because there was more fissile material (^{235}U) in the reflector than there was in the inner sphere, which caused the source points to migrate towards the reflector after each cycle. Once the source points reached the reflector, they settled there and had a much greater chance of escaping the system, which increased the standard deviation of k_{eff} estimator tallies [19].

Standard deviations (σ) > 0.001 are not indicative of a problem with MCNP, since more cycles and neutrons can be run to obtain better statistics. However, these outliers were found to negatively affect neural network performance. To demonstrate this, two neural networks were trained on two different sets of training data: one with all of the MCNP output files, and one in which the 395 MCNP output files with standard deviations (σ) > 0.001 were removed. The results shown in Figure 3.12 and Figure 3.13 indicate that even though 395 MCNP output files represent $< 0.026\%$ of the data, their presence increased the mean absolute error (3.9) on training data by approximately 4% (Figure 3.12, Figure 3.13). As a result, the 395 MCNP output files with standard deviations (σ) > 0.001 were removed

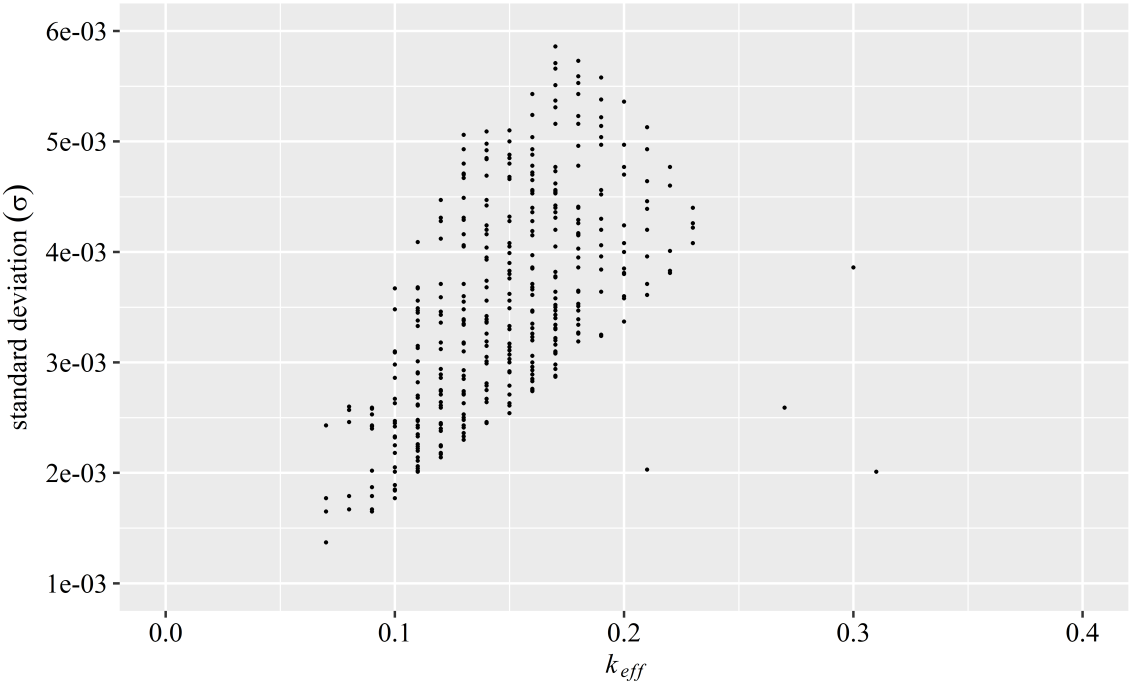

Figure 3.10: k_{eff} standard deviations from 395 MCNP output files ($\sigma > 0.001$)

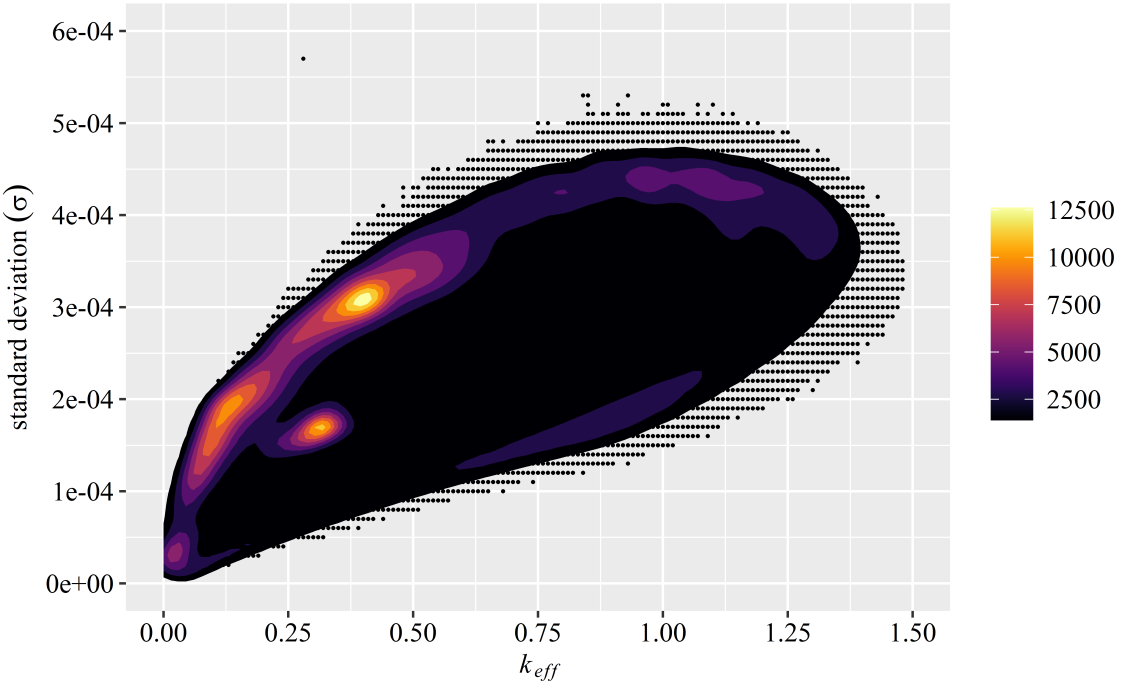

Figure 3.11: k_{eff} standard deviations from 1,542,397 MCNP output files ($\sigma \leq 0.001$)

from the dataset.

$$\text{mean absolute error (MAE)} = \frac{1}{n}\sum_{i=1}^{n}|x_{observed_i} - x_{predicted_i}| \tag{3.9}$$

3.2.2 Splitting and Scaling Data

1,542,397 MCNP output files were divided into training and test data using a standard 80-20 data split [78]. This was done by setting aside 20% of the data for testing and using the remaining 80% for training and cross-validation. The training and cross-validation data was scaled by calculating the column-wise mean and standard deviation of each discrete variable, subtracting each mean from its corresponding column, and 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 and cross-validation data.

The training, cross-validation, and test data were then one-hot-encoded using the *dummyVars* function in the *caret* software package [77, 78]. The *dummyVars* function converts categorical variables into multi-column dummy variables, where membership within each sub-category is either 0 or 1. The purpose of scaling and one-hot-encoding data was to increase stability and add predictors to the model, which improves the initial parameter (θ) updates and overall neural network performance [59, 78].

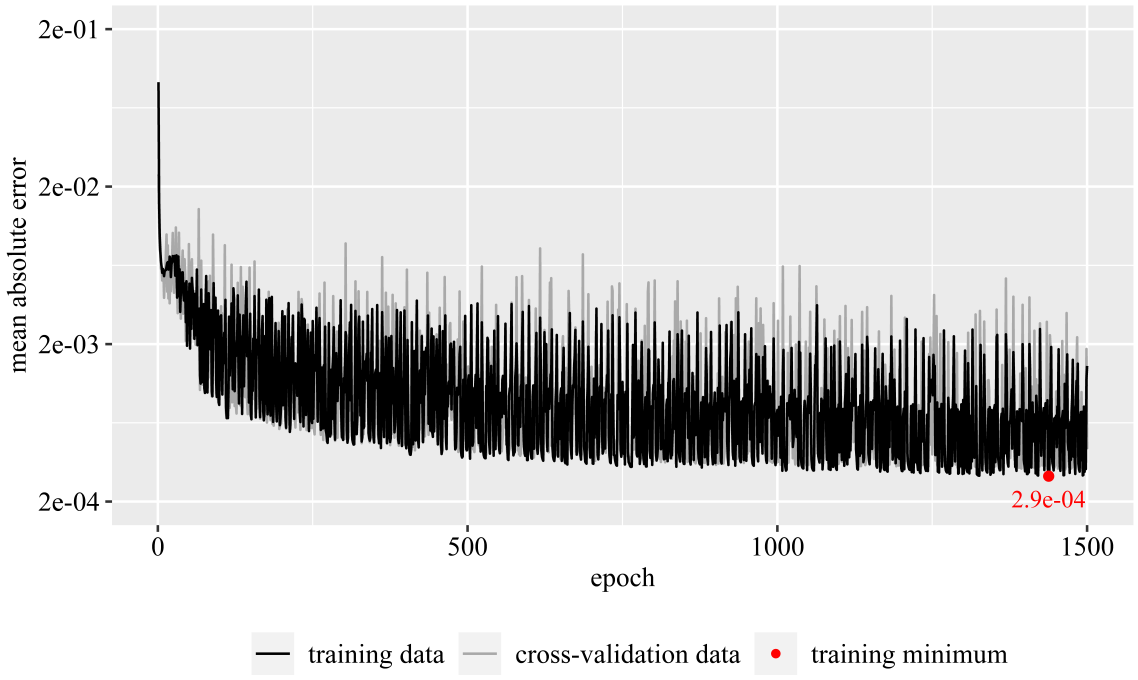

Figure 3.12: Neural network trained on 987,387 samples (all σ)

Figure 3.13: Neural network trained on 987,134 samples ($\sigma \leq 0.001$)

3.3 Neural Network Metamodel

The neural network metamodel is comprised of 15 deep feedforward neural networks that were built using the *Keras* [43] and *TensorFlow* [27] software packages. The number of hidden layers and the number of neurons in each hidden layer were determined using a manual training process, which is described below.

Manual training process

1. Add a hidden layer to the neural network.
2. Vary the number of neurons in the newly added hidden layer by powers of 2 (e.g., 4, 8, 16, ...).
3. Train each neural network for 1,500 epochs using the MSE loss function and Adamax optimization algorithm (Algorithm 3, Figure 3.14) [73].
4. Train each neural network for an additional 150 epochs, and save the model after each epoch (Figure 3.15).
5. Determine which neural network performs best by comparing the sum and weighted mean of mean absolute errors on training and cross-validation data.
6. If more than one neural network performs best, retrain the best-performing neural networks three more times and average the results.
7. Repeat these steps using the best-performing neural network as the iterative baseline for the next set of models.

Each neural network was trained using a "mini-batch" training approach [59], which consisted of splitting training data into batches of 8,192 samples each and iteratively updating parameters (θ) after each batch. The neural network with the lowest sum of mean absolute errors on training and cross-validation data was selected as the baseline for the neural network metamodel. The sum of mean absolute errors on training and cross-validation data was selected because it seemed reasonable to select a neural network that performed well on both sets of data.

The neural network with the lowest sum of mean absolute errors on training and cross-validation data had six hidden layers, with the following number of neurons in each hidden layer: 8,192-256-256-256-256-16. The weighted mean of mean absolute errors on training and cross-validation data was compared to the sum of mean absolute errors on training and cross-validation data to minimize bias in selecting a neural network that performed better on the sum of mean absolute errors, which favors training data (Figure 3.18, Figure 3.19). Although the sums and weighted means differ slightly, the results were the same (Figure 3.18, Figure 3.19).

During the early stages of this work, each neural network was trained for 2,750-5,500 epochs. However, it took 3-5 hours to train each neural network, which slowed the manual training process to a crawl. After testing several different combinations of learning rates, loss functions, and optimization algorithms, 1,650 epochs was determined to be a reasonable trade-off between neural network performance and training time. An initial training cycle of 1,500 epochs was selected because the mean absolute errors on training and cross-validation data leveled off between 1,000 and 1,200 epochs—and showed little improvement beyond 1,400 epochs (Figure 3.14, Figure 3.15). The final training cycle of 150 epochs was selected

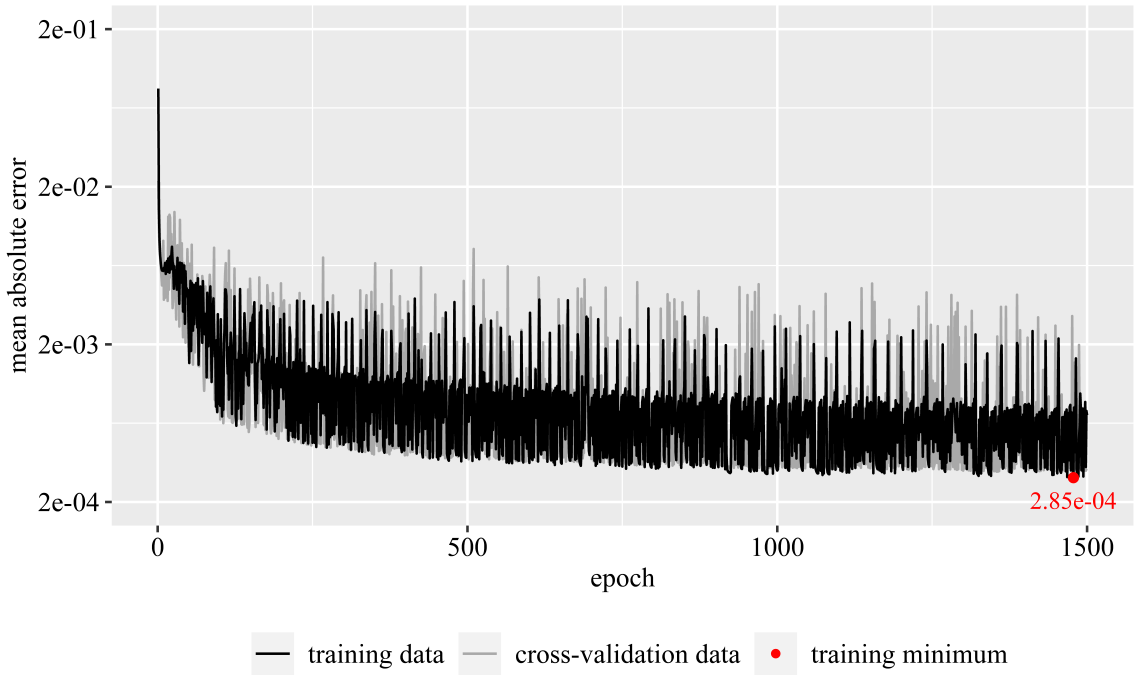

Figure 3.14: Neural network trained using the MSE loss function

The

Figure 3.15: Neural network from Figure 3.14 trained for an additional 150 epochs

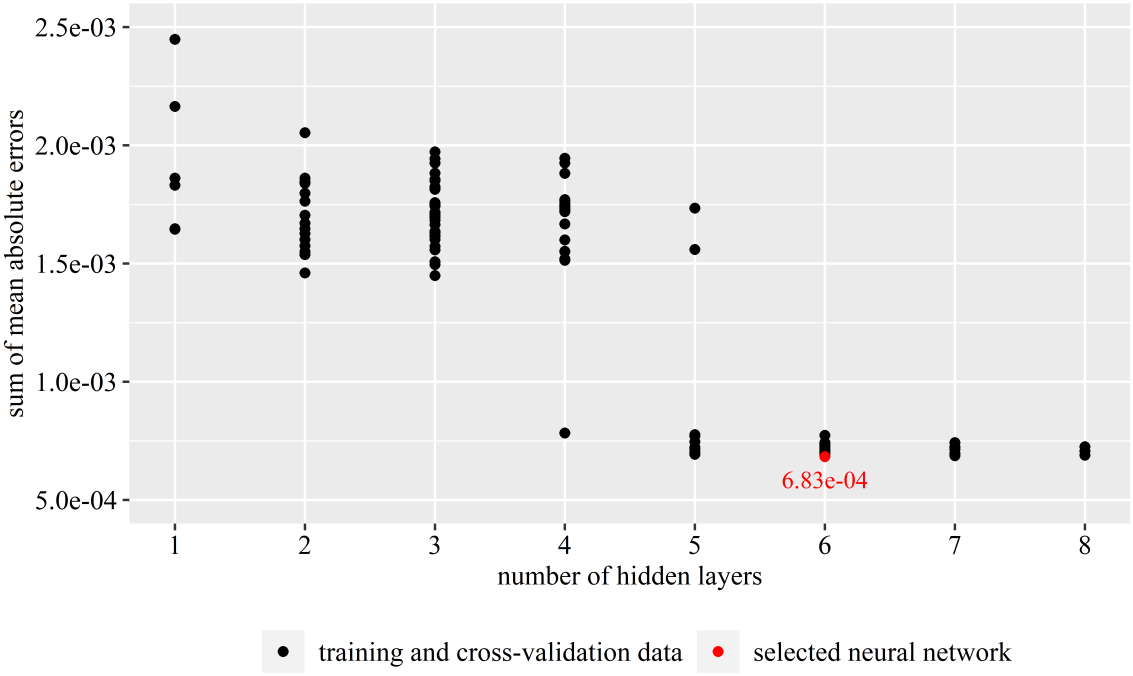

Figure 3.16: Neural networks trained for 1,650 epochs using the MSE loss function

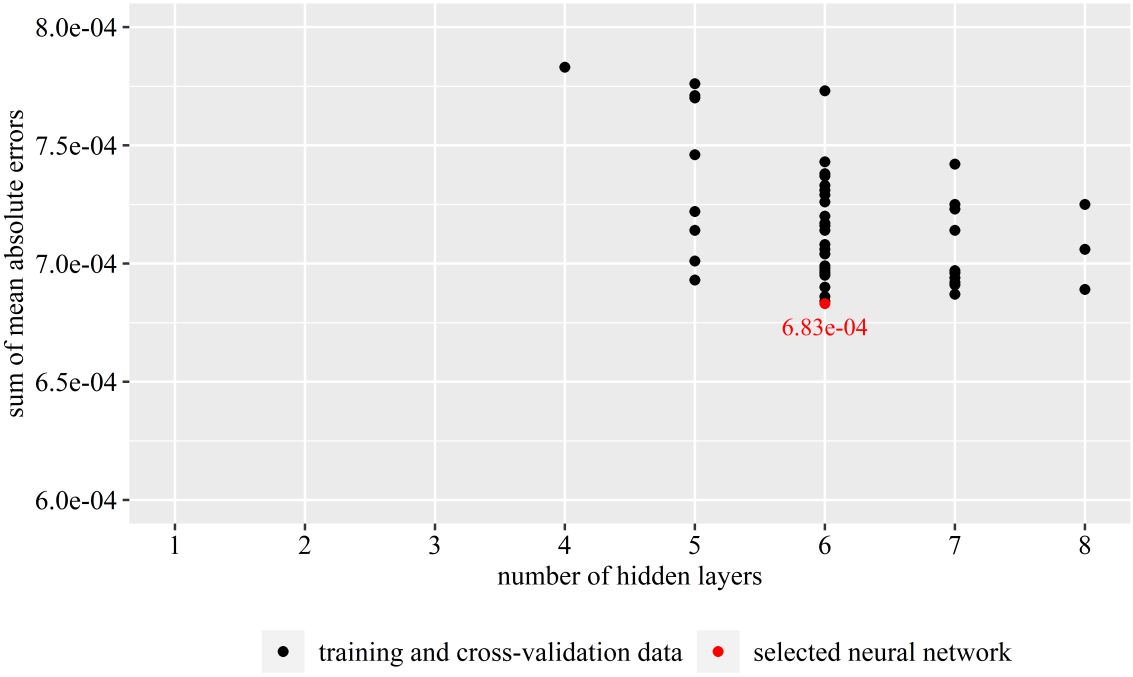

Figure 3.17: Neural networks trained for 1,650 epochs using the MSE loss function

Figure 3.18: Neural networks trained for 1,650 epochs using the MSE loss function

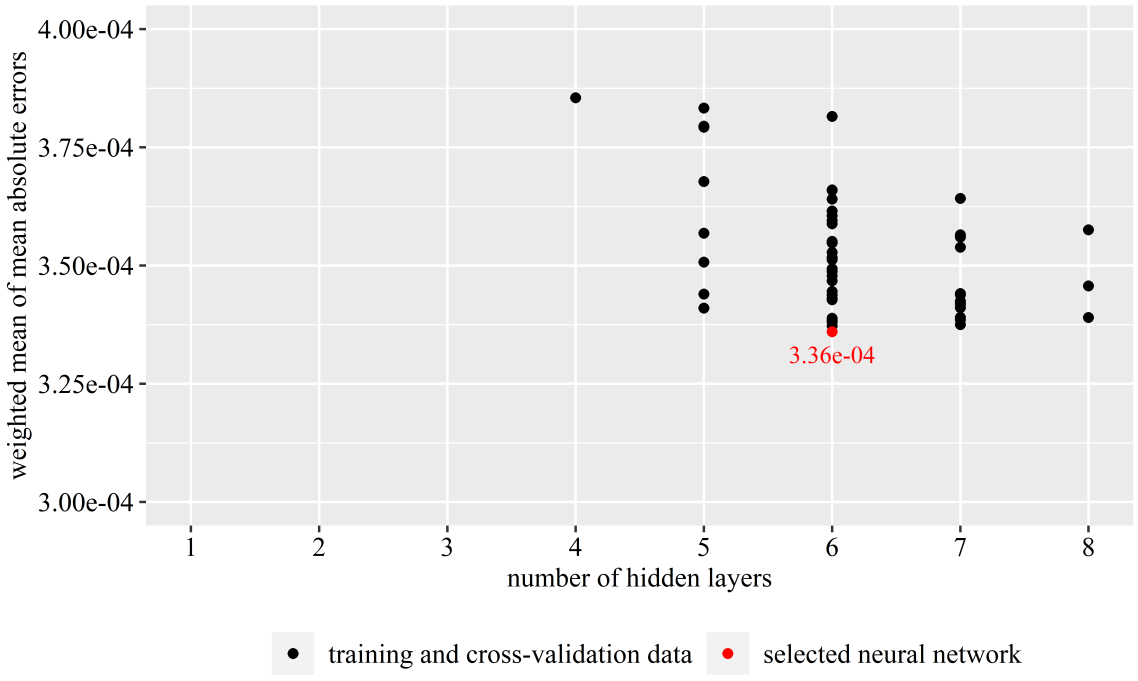

Figure 3.19: Neural networks trained for 1,650 epochs using the MSE loss function

because it generated local mean absolute error minima and reduced the number of models that needed to be saved (9.21 MB $\times$ 150 models = 4.38 GB). Training each neural network for 1,650 epochs also shortened the training time to 1.5 hours, which saved approximately 300-500 hours for the 195 mean absolute errors that are plotted and shown in Figures 3.16-3.19.

3.3.1 Applying the Sum of Squared Errors Loss Function

When a neural network is trained using a mini-batch training aproach [59], the number of samples in the last batch is determined by batch size (n) and the total number of training samples (a) (3.10).

$$\begin{aligned} \text{number of samples in the last batch} &= a \bmod n \\ &= 987{,}134 \bmod 8{,}192 \\ &= 4{,}094 \end{aligned} \tag{3.10}$$

When the MSE loss function (2.26) is used and $a \bmod n > 1$, each sample in the last batch is weighted more heavily than all other samples by a factor of $n/(a \bmod n)$. This weight difference exists because the summed error (or loss) is divided by the number of samples in each batch (2.26). This section attempts to quantify the effect that the MSE loss function (2.26) has on mini-batch training [59] by comparing it to the sum of squared errors (SSE) loss function, which lacks a leading $\frac{1}{n}$ term (3.11).

$$\text{sum of squared errors (SSE)} = \sum_{i=1}^{n} (x_{observed_i} - x_{predicted_i})^2 \tag{3.11}$$

where:

n = total number of variables that are observed or predicted

x = vector of observations or predictions

Several different optimization algorithms were tested using both loss functions. The results shown in Figure 3.20 and Figure 3.21 indicate that the SSE loss function and the Adamax optimization algorithm (Algorithm 3) [73] outperformed all other combinations of loss function and optimization algorithm for this case study. This combination worked well because the exponentially weighted infinity norm in the Adamax optimization algorithm (Algorithm 3) [73] tends to flatten the gradients of the smaller, more heavily weighted last batch.

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 3.22, Figure 3.23) [59]. The default Adamax learning rate is set to 0.002 [27, 44, 73]. When the Adamax learning rate was incrementally reduced to 0.00075, the mean absolute error on training data lowered to 2.72e-04, and the mean absolute errors on cross-validation and test data lowered to 2.99e-04 and 3.05e-04, respectively (Figure 3.23). As a result, the learning rate was set to 0.00075.

The MSE and SSE loss functions are shown in 3.12 and 3.13, respectively. To show how the difference between these loss functions is applied to the backpropagation algorithm and parameter (θ) updates, 2.28 and 3.11 were expanded (3.12, 3.13) and simplified (3.14-3.17),

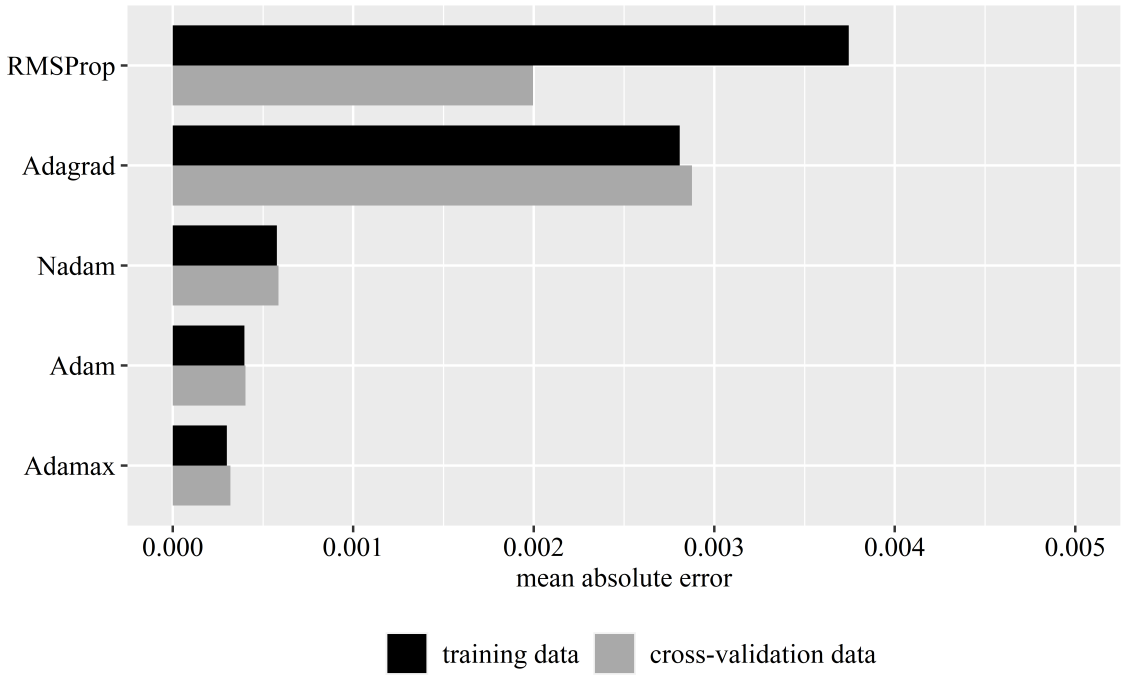

Figure 3.20: Neural networks trained for 1,650 epochs using the MSE loss function

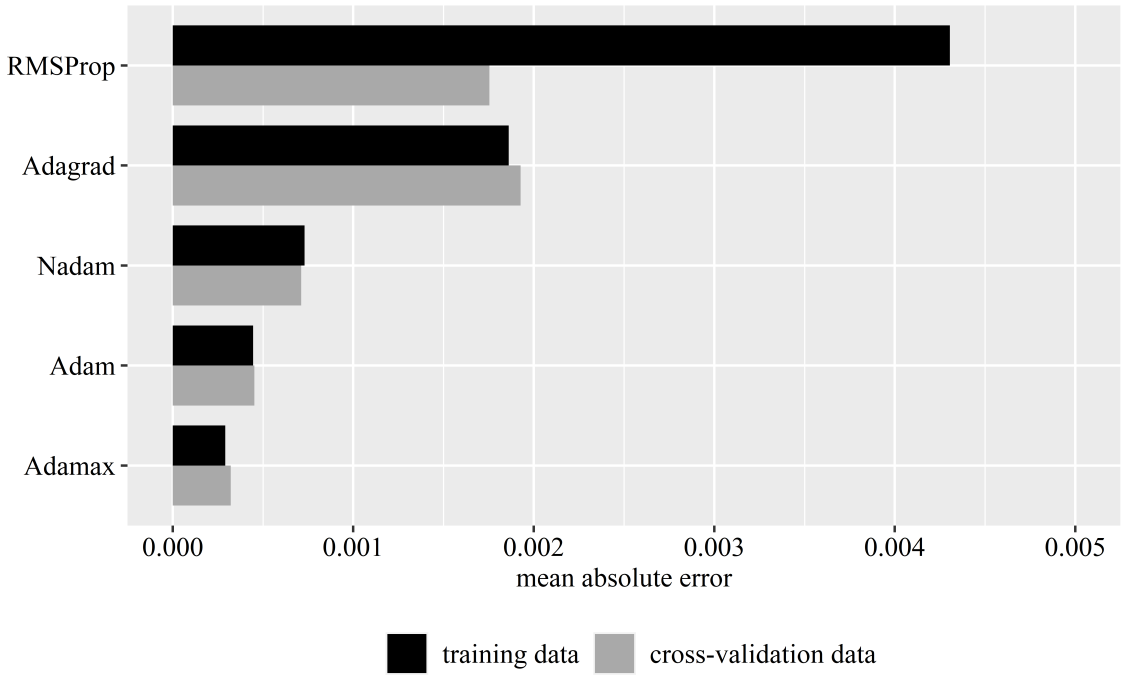

Figure 3.21: Neural networks trained for 1,650 epochs using the SSE loss function

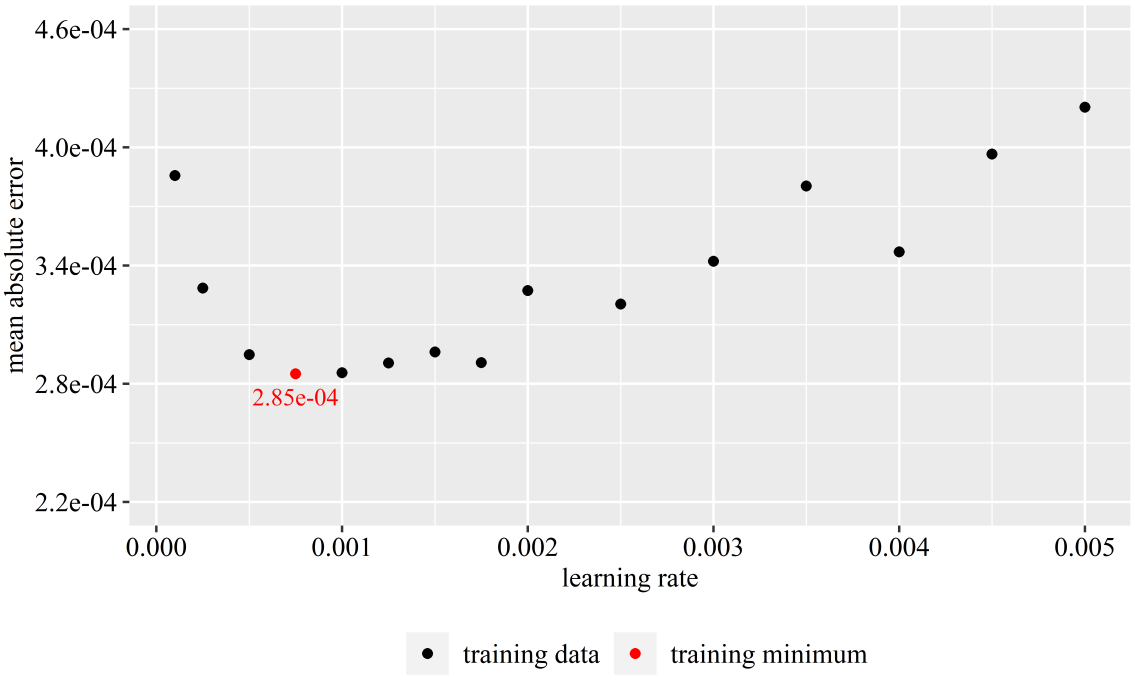

Figure 3.22: Neural networks trained for 1,650 epochs using the MSE loss function

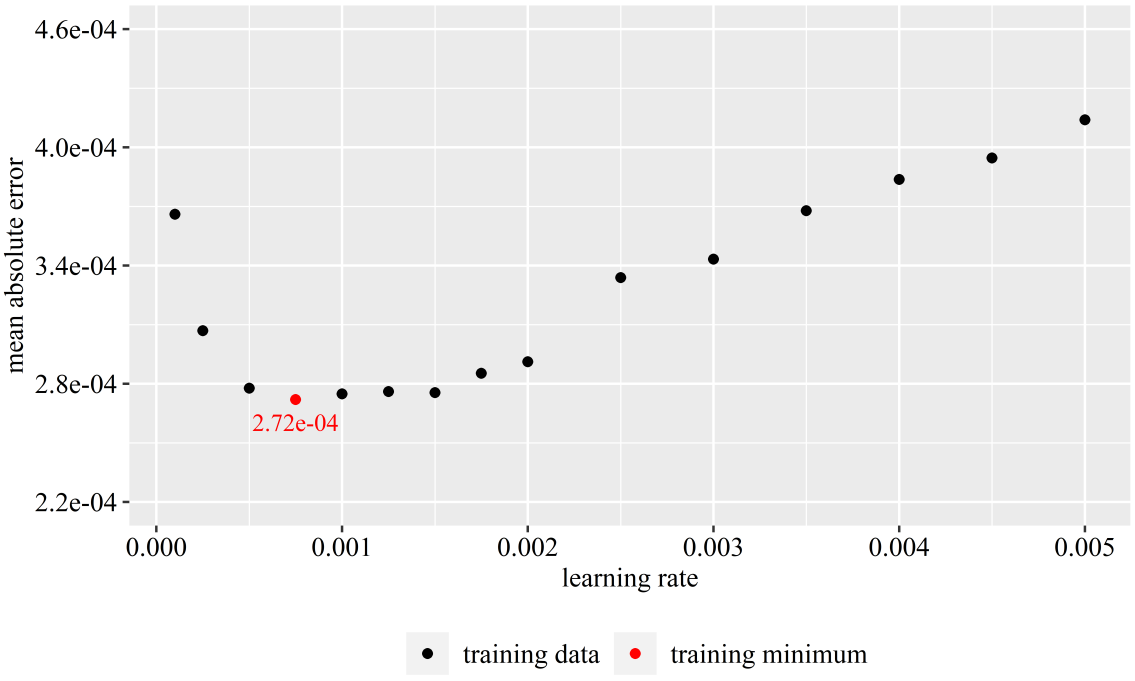

Figure 3.23: Neural networks trained for 1,650 epochs using the SSE loss function

and the partial derivatives with respect to v were calculated (3.15, 3.18, 3.19).

$$\text{MSE} = \frac{1}{n}\sum_{j=1}^{n}\left(y_j - \left[W_7^\top \prod_{i=1}^{6}\Big(max\{0, W_i^\top x_i + b_i\}\Big) + b_7\right]_j\right)^2 \tag{3.12}$$

$$\text{SSE} = \sum_{j=1}^{n}\left(y_j - \left[W_7^\top \prod_{i=1}^{6}\Big(max\{0, W_i^\top x_i + b_i\}\Big) + b_7\right]_j\right)^2 \tag{3.13}$$

By substituting the intermediate variables u (3.14) and v (3.15) into 3.12 and 3.13, both loss functions can be simplified.

$$u(W,b,x) = \left[W_7^\top \prod_{i=1}^{6}\Big(max\{0, W_i^\top x_i + b_i\}\Big) + b_7\right] \tag{3.14}$$

$$v(y,u) = y - u \tag{3.15}$$

The simplified MSE and SSE loss functions are shown in 3.16 and 3.17.

$$\text{MSE}(v) = \frac{1}{n}\sum_{j=1}^{n} v_j^2 \tag{3.16}$$

$$\text{SSE}(v) = \sum_{j=1}^{n} v_j^2 \tag{3.17}$$

The partial derivatives with respect to v were then calculated to compare the leading terms

of both loss functions (3.18, 3.19).

$$\begin{aligned}\frac{\partial \text{MSE}(v)}{\partial v} &= \frac{\partial}{\partial v}\frac{1}{n}\sum_{j=1}^{n} v_j^2 \\ &= \frac{1}{n}\sum_{j=1}^{n}\frac{\partial}{\partial v} v_j^2 \\ &= \frac{1}{n}\sum_{j=1}^{n} 2v \\ &= \frac{2}{n}\sum_{j=1}^{n} v\end{aligned} \tag{3.18}$$

$$\begin{aligned}\frac{\partial \text{SSE}(v)}{\partial v} &= \frac{\partial}{\partial v}\sum_{j=1}^{n} v_j^2 \\ &= \sum_{j=1}^{n}\frac{\partial}{\partial v} v_j^2 \\ &= \sum_{j=1}^{n} 2v \\ &= 2\sum_{j=1}^{n} v\end{aligned} \tag{3.19}$$

v is difficult to estimate because it is highly dependent on whether the errors have a slight positive or negative bias. However, if the signs of the errors are ignored, and all errors are assumed to be of equal magnitude across all batches, then it is reasonable to also assume that the partial derivatives of the SSE loss function are proportional to batch size—and the partial derivatives of the MSE loss function are equal, regardless of batch size. When the Adamax optimization algorithm (Algorithm 3) [73] is used, these assumptions can be extended to the parameter (θ) updates as well. The main takeaway from this section is that parameter (θ) updates *should* be proportional to batch size, as they are when the SSE loss

function is used. If parameter (θ) updates are weighted equally, as they are when the SSE loss function is used, then a batch with one sample is given the same weight as a batch with thousands of samples.

The MSE and SSE loss functions were compared by training neural networks on several different sets of training data (Table 3.8). The first set of training data in Table 3.8 was used to train the neural network metamodel. The others were random subsets of the first set of training data (Table 3.8) and were only used to compare the MSE and SSE loss functions. Each neural network was trained for 500 epochs using the Adamax optimization algorithm (Algorithm 3) [73] with the learning rate set to 0.00075. The bias (b_6) and mean absolute change in bias (b_6) are plotted and shown in Figures 3.24-3.26 and Appendix C.

The bias (b_6) and mean absolute change in bias (b_6) were selected because they are easier to track than the loss, which is confusingly tracked as a rolling, sample-weighted average in *TensorFlow* [26, 27]. The results shown in Figure 3.24 and Figure 3.25 indicate that the MSE loss function consistently weighted the samples in the last batch more heavily, which caused the neural networks to generalize poorly [59] and perform worse than neural networks that were trained using the SSE loss function. The results corresponding to the training data (Table 3.8) are plotted and shown in Figure 3.27 and Figure 3.28.

The results shown in Figure 3.27 and Figure 3.28 indicate that the MSE loss function caused neural network performance to degrade as $a \bmod n \to 1$. These results also indicate that the SSE loss function consistently outperformed the MSE loss function, regardless of the number of samples in the last batch ($a \bmod n$). As a result, the SSE loss function was used to train the neural network metamodel.

Table 3.8: Training data

number of training samples (a)	batch size (n)	$a \bmod n$	a/n
987,134	8,192	4,094	121
985,088	8,192	2,048	121
984,064	8,192	1,024	121
983,552	8,192	512	121
983,296	8,192	256	121
983,168	8,192	128	121
983,104	8,192	64	121
983,072	8,192	32	121
983,056	8,192	16	121
983,048	8,192	8	121
983,044	8,192	4	121
983,042	8,192	2	121
983,041	8,192	1	121
983,040	8,192	0	120

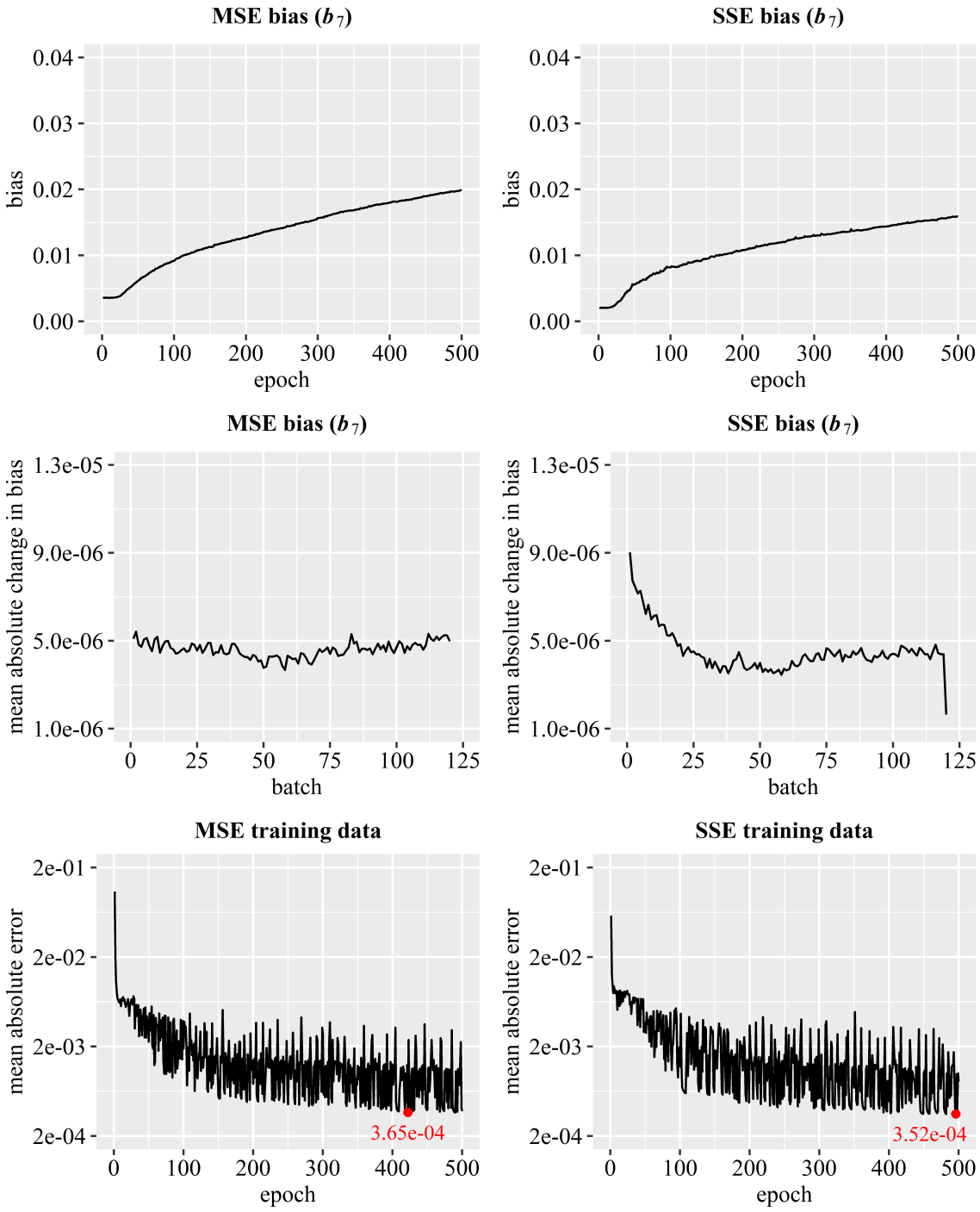

Figure 3.24: Neural networks trained on 987,134 samples (a mod n = 4,094)

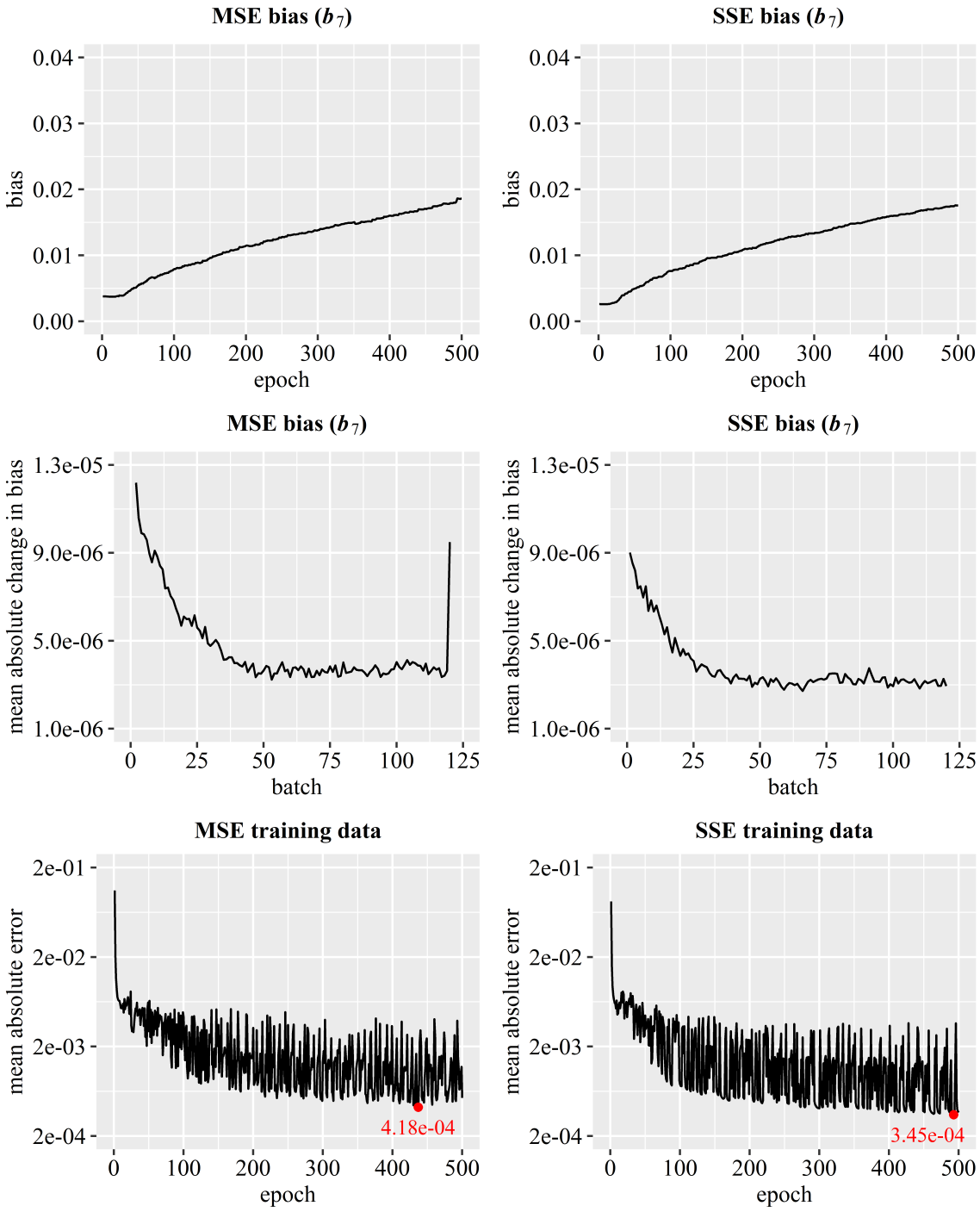

Figure 3.25: Neural networks trained on 983,041 samples (a mod n = 1)

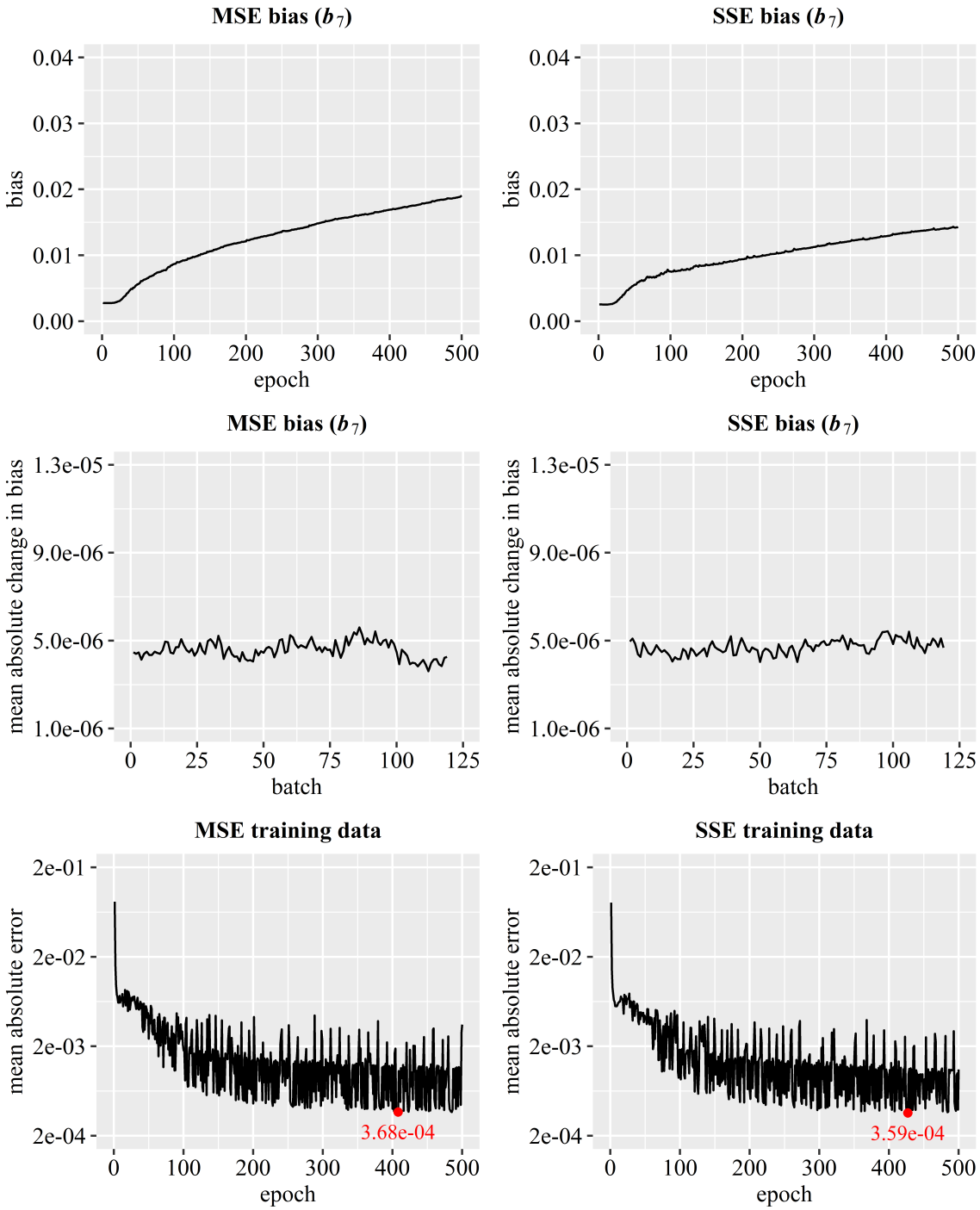

Figure 3.26: Neural networks trained on 983,040 samples (a mod n = 0)

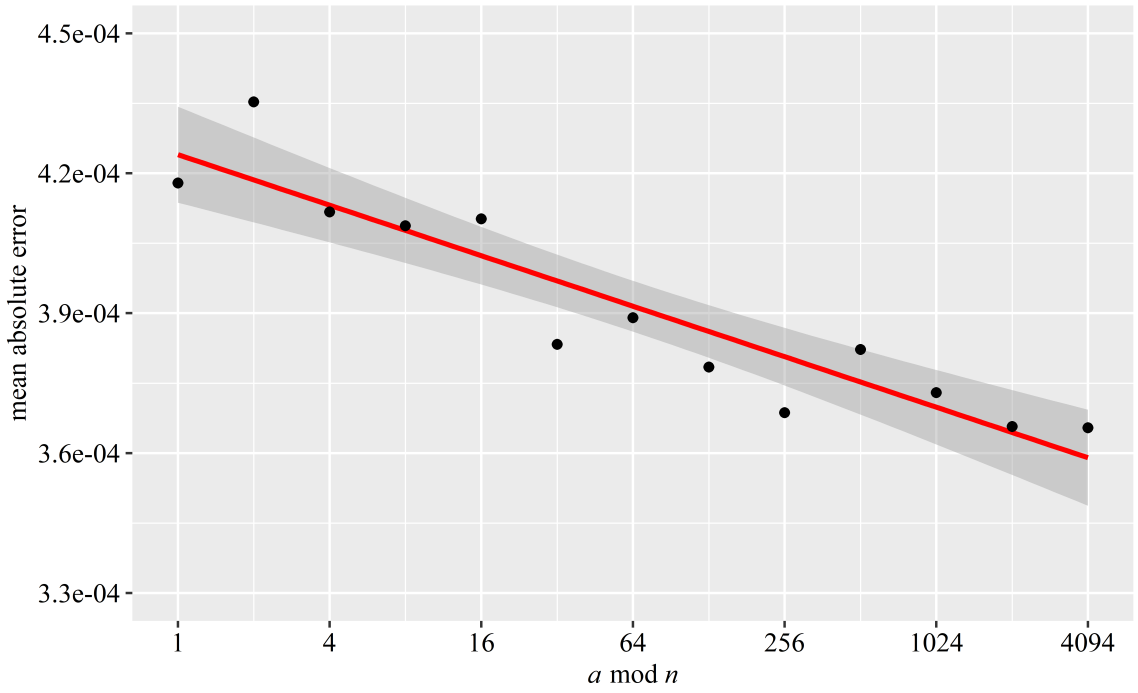

Figure 3.27: Neural networks trained for 500 epochs using the MSE loss function

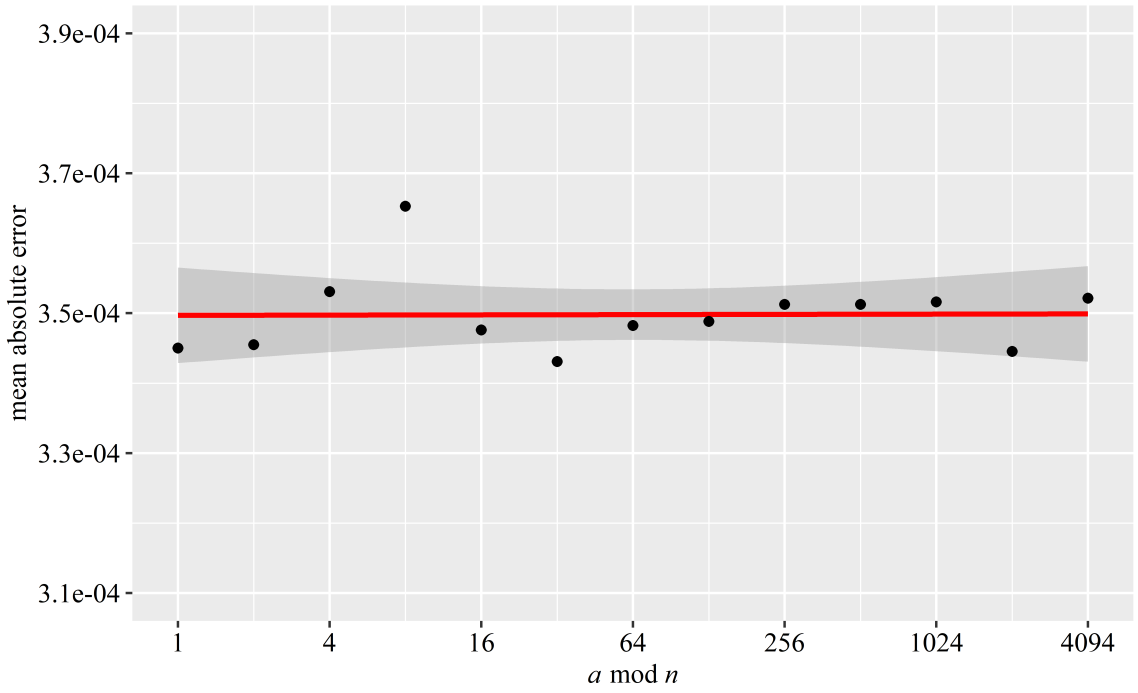

Figure 3.28: Neural networks trained for 500 epochs using the SSE loss function

3.3.2 Model Averaging

The purpose of using more than one neural network is to reduce *generalization error*, which is the expected error on new input [59]. The idea behind this technique, known as *model averaging*, is that each neural network will not make the exact same errors on test data [59]. The expected squared error of the neural network metamodel is shown in 3.20 [59] for a set of n regression models with errors (ε_i) that are drawn from a zero-mean multivariate distribution with variances $\mathbb{E}[\varepsilon_i] = v$ and covariances $\mathbb{E}[\varepsilon_i \varepsilon_j] = c$.

$$\mathbb{E}[(\frac{1}{n}\sum_i \varepsilon_i)^2] = \frac{1}{n^2}\mathbb{E}[\sum_i (\varepsilon_i^2 + \sum_{j \neg 1} \varepsilon_i \varepsilon_j)] \rightarrow \frac{1}{n}v + \frac{n-1}{n}c \tag{3.20}$$

If the errors (ε_i) are perfectly correlated, then $c = v$, and the expected squared error reduces to v [59]. Conversely, if the errors (ε_i) are perfectly uncorrelated, then $c = 0$, and the expected squared error reduces to v/n [59]. In this dissertation, metamodels comprised of 1-20 neural networks were trained for 1,650 epochs using the SSE loss function 3.11. Then, each metamodel was used to predict k_{eff} values for test data, which consisted of 246,784 output files the neural networks had never seen. The mean MAEs for all 20 neural network metamodels are plotted and shown in Figure 3.29 and Figure 3.30.

The Nelder-Mead [33], Broyden-Fletcher-Goldfarb-Shanno (BFGS) [58], and Simulated Annealing (SA) [90] algorithms were used to assign weights to each neural network, with better models given higher weights and worse models given lower weights [44, 59]. The weighted mean absolute errors for all 20 neural network metamodels are shown in Figure 3.29 and Figure 3.30. These results indicate that the errors on test data are partially correlated

(3.20), as evidenced by the mean absolute error asymptotically decreasing to a value of 2.58e-04 (Figure 3.29, Figure 3.30).

Although not shown in Figure 3.29 and Figure 3.30, the results also indicate that there was no advantage in using a specific algorithm to assign weights, since all of them assigned the exact same weights to each neural network. Based on these results, it was determined that 15 neural networks would be used, with weights assigned by the arbitrarily selected Nelder-Mead algorithm [90] (Figure 3.29, Figure 3.30). The weighted errors on training and test data are plotted and shown in Figure 3.31 and Figure 3.32.

The results shown in Figure 3.29 and Figure 3.32 indicate that the maximum error is $< |0.004|$ for an ensemble of 15 neural networks. These results also indicate that the mean absolute errors on training and test data are 2.26e-04 and 2.58e-04, respectively (Figure 3.31, Figure 3.32). These values are notable because they are less than the mean MCNP standard deviation (σ) of 2.70e-04 (Figure 3.11), which indicates that the neural network metamodel has reached a performance threshold, relative to MCNP [19]. However, all of these errors are built on top of the underlying uncertainties that are integral to MCNP [19] and ENDF/B-VII.1 [42]. To compensate for these uncertainties, all k_{eff} values ≥ 0.95 were assumed to be critical or supercritical ($k_{eff} \geq 1$). This limit is based on an article written by S.E. Coleman and W.J. Zywiec [46], which establishes a limit of 0.9615 for plutonium systems, based on critical experiment benchmark data [20]. This data is plotted and shown in Figure 3.33 [20, 46].

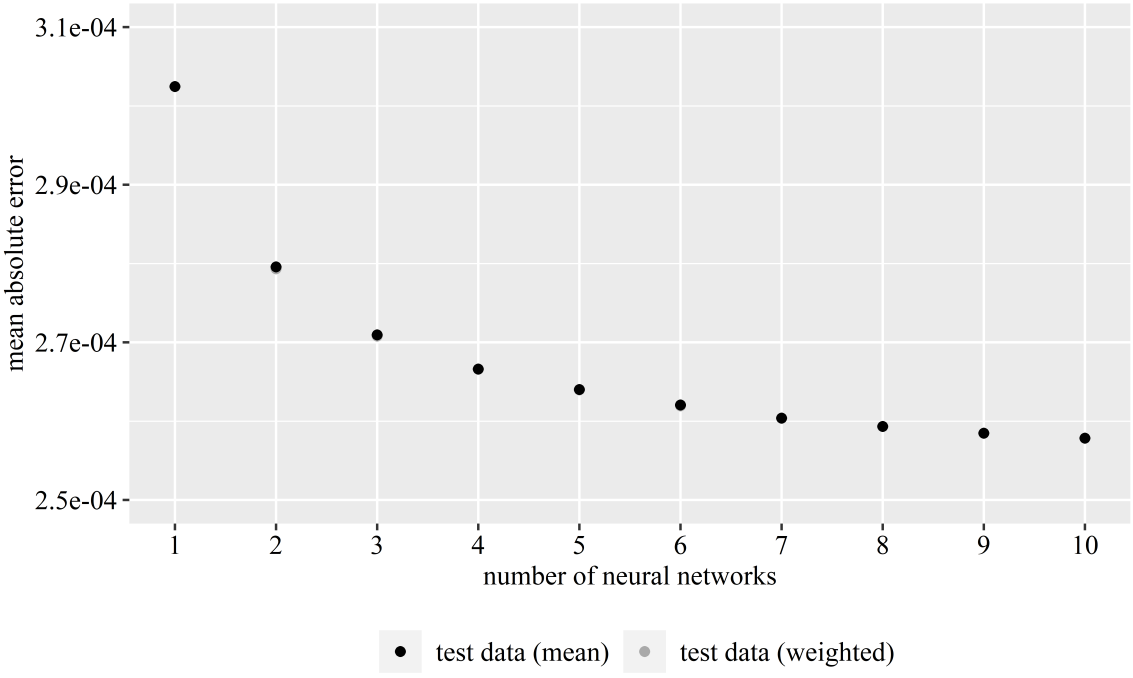

Figure 3.29: Neural network metamodels trained using the SSE loss function

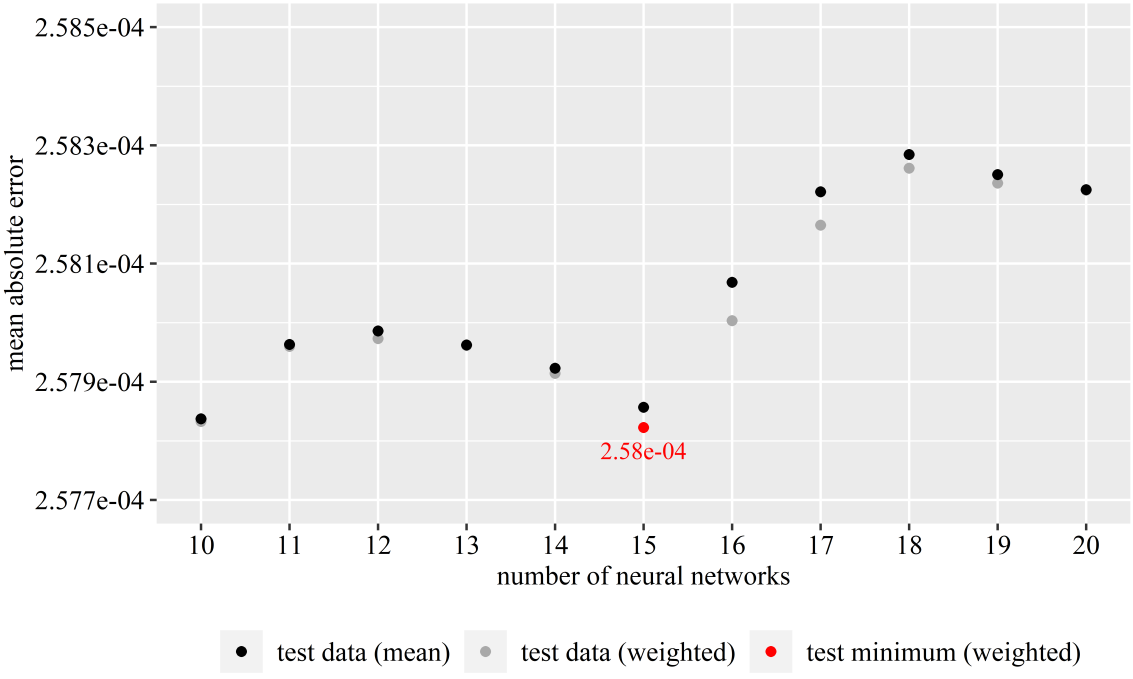

Figure 3.30: Neural network metamodels trained using the SSE loss function

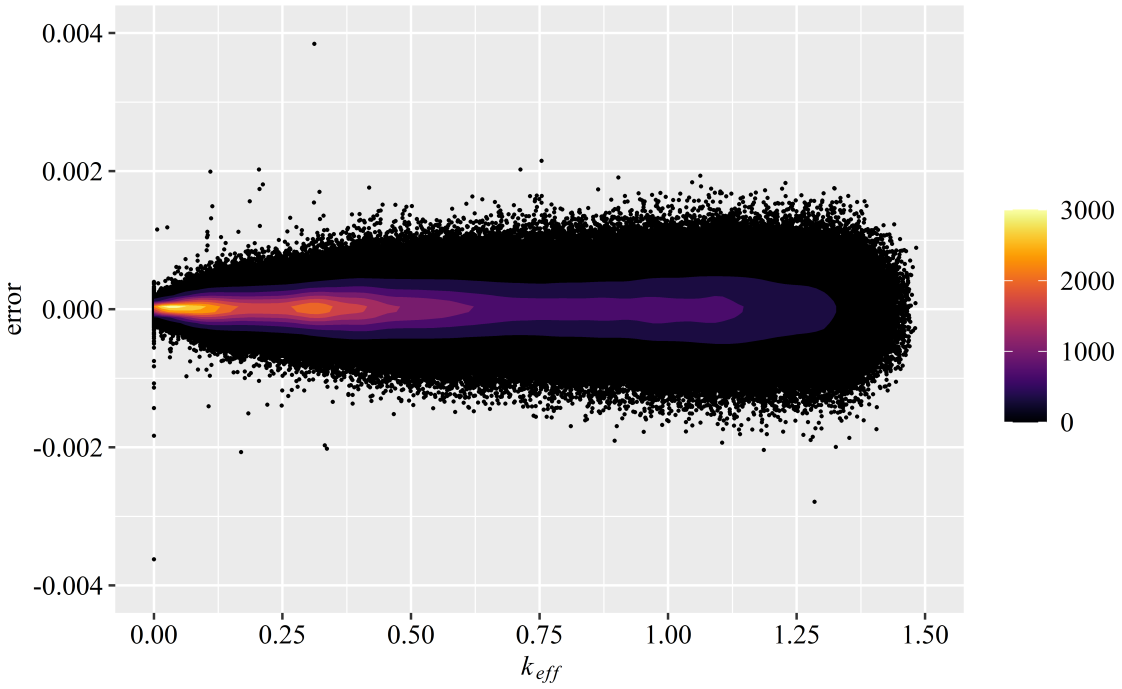

Figure 3.31: Weighted errors on training data (15 neural networks, MAE = 2.26e-04)

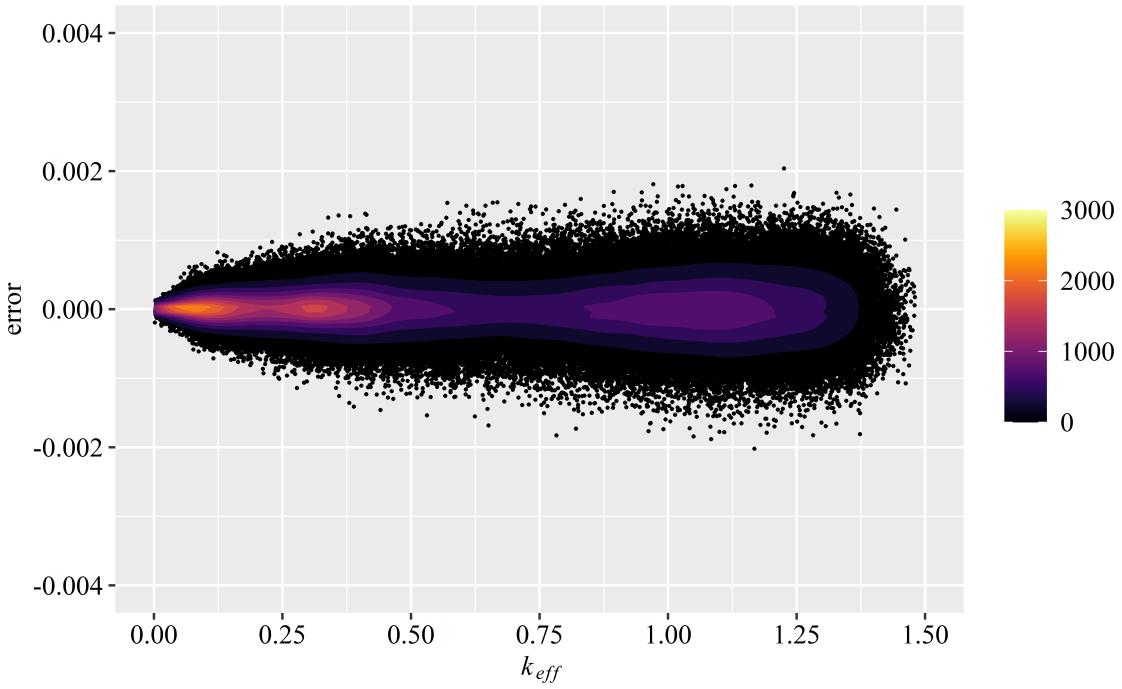

Figure 3.32: Weighted errors on test data (15 neural networks, MAE = 2.58e-04)

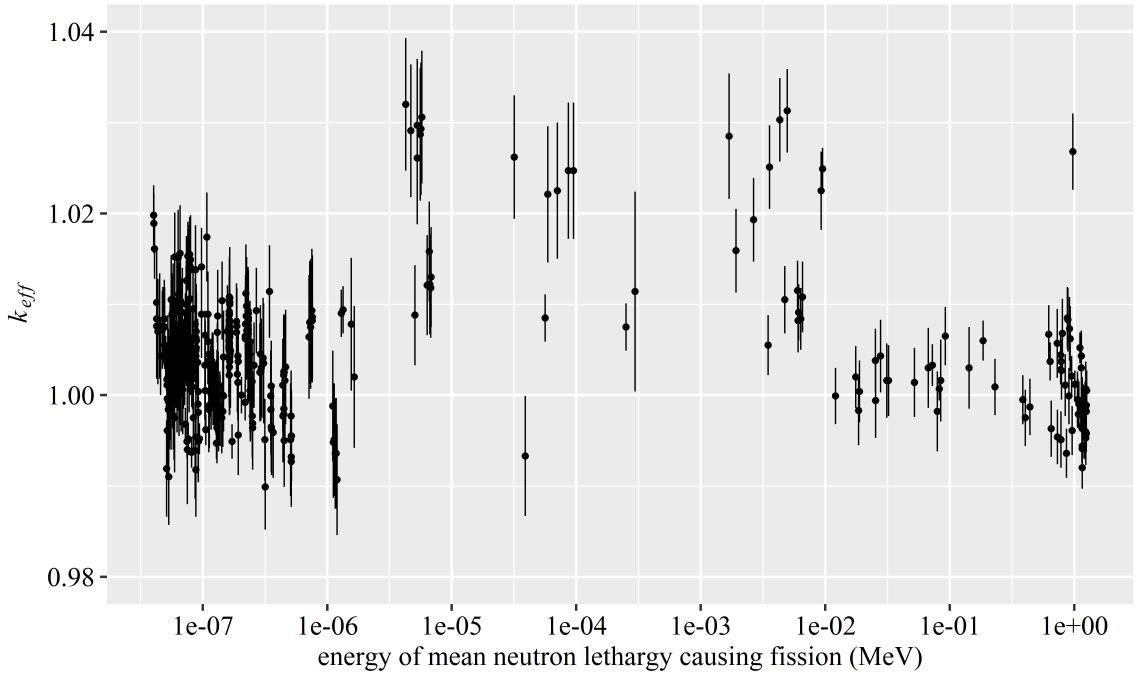

Figure 3.33: Plutonium critical experiment benchmarks (with uncertainties)

The critical experiment benchmark data shown in Figure 3.33 represents 488 plutonium critical experiments [20, 46], which were modeled using MCNP [19] and ENDF/B-VII.1 [41]. The uncertanties in this plot represent the combined experimental, MCNP [19], and nuclear data [42] uncertainties, which were used in a complex set of equations to calculate a upper subcritical limit (USL) of 0.9615 [46]. In this dissertation, a much simpler equation (3.21) was developed to calculate a slightly lower USL of 0.9524, which was conservatively rounded down to 0.95. 3.21 satisfies ANSI/ANS-8.24 [18], which is the national consensus standard that sets the requirements for validating radiation transport codes.

$$\text{USL} = \begin{cases} max\{|1-k_{eff}|+3\sigma\}-0.02 & \text{if } k_{eff} < 1 \\ \varnothing & \text{otherwise} \end{cases} \tag{3.21}$$

where:

σ = standard deviation

0.02 = safety margin [18]

3.4 Estimating Process Criticality Accident Risk

The *R Notebook* markdown file in Appendix A.1 controls all of the functions that were used to build the coupled Bayesian network and neural network metamodel and estimate process criticality accident risk. This markdown file was decomposed into the following high-level tasks:

1. Extract, transform, and load data from a CSV file, RData file, or directory containing MCNP output files
2. Build a neural network metamodel
3. Build a Bayesian network
4. Generate samples and estimate process criticality accident risk

These high-level tasks were decomposed further and assigned to the following functions and subfunctions, which are included in Appendix A:

1. *Tabulate* function ← Extract and load data from a CSV file, RData file, or directory containing MCNP output files
    1.1. *Scale* function ← Transform and save data as an RData file
2. *NN* function ← Build a neural network metamodel
    2.1. *Model* function ← Build neural networks
    2.2. *Fit* function ← Train neural networks

2.3. *Plot* function ← Plot training and cross-validation metrics and save plots as PNG files

2.4. *Test* function ← Predict k_{eff} values for test data and assign weights to the neural network metamodel

3. *BN* function ← Build a Bayesian network

4. *Risk* function ← Estimate process criticality accident risk

4.1. *Sample* function ← Generate samples

First, the *Tabulate* function A.1.2 is called to extract and load data from a CSV file, RData file, or directory containing MCNP output files. If a CSV file or MCNP output files are loaded, the *Scale* function (Appendix A.1.2.1) is called to transform and save training, cross-validation, and test data as an RData file. Then, the *NN* function (Appendix A.1.3) is called to build a neural network metamodel, which is passed to R's global environment as a list of models and weights [43, 27]. The neural network metamodel is essentially a performance function that leverages more than 1.5 million MCNP calculations to predict billions of k_{eff} values. These predictions bypass the computational bottleneck discussed in Chapter 1 and reduce the variance (v) of the risk estimate to a reasonably low value (3.22) [52].

$$v = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{3.22}$$

where:

v = variance

n = number of batches

x_i = risk estimate (e.g., 1e-06 accidents per year)

$\bar{x}$ = mean risk estimate

The *BN* function (Appendix A.1.4) is called to build a Bayesian network, which is passed to R's global environment as a large *bn.fit* object [107]. Then, the *Sample* function (Appendix A.1.5.1) is called to generate conditional probability queries [107]. These conditional probability queries were formulated based on evidence that a process criticality accident will involve $\geq$ 120 grams of plutonium, homogeneously dispersed in a $\geq$ 7-cm radius sphere.

This evidence is based on the minimum critical masses and radii of the most reactive system, which is 0-4 kilograms of alpha-phase plutonium (ρ = 19.86 g/cm^3, 95% ^{239}Pu, 5% ^{240}Pu by weight), homogeneously dispersed in a sphere of high-density polyethylene, with six inches of beryllium reflection, and one inch of water reflection (Figure 3.34). Although it is difficult to visualize the entire metamodel domain, there are no systems with $<$ 120 grams of plutonium or a radius $<$ 7 cm (1.44 liters) that have a $k_{eff} \geq 0.95$. These assumptions are consistent with the minimum critical masses and radii in J.K. Thompson [116], which require less plutonium to achieve criticality than the water-moderated spheres in *A Review of Criticality Accidents* [86, 115].

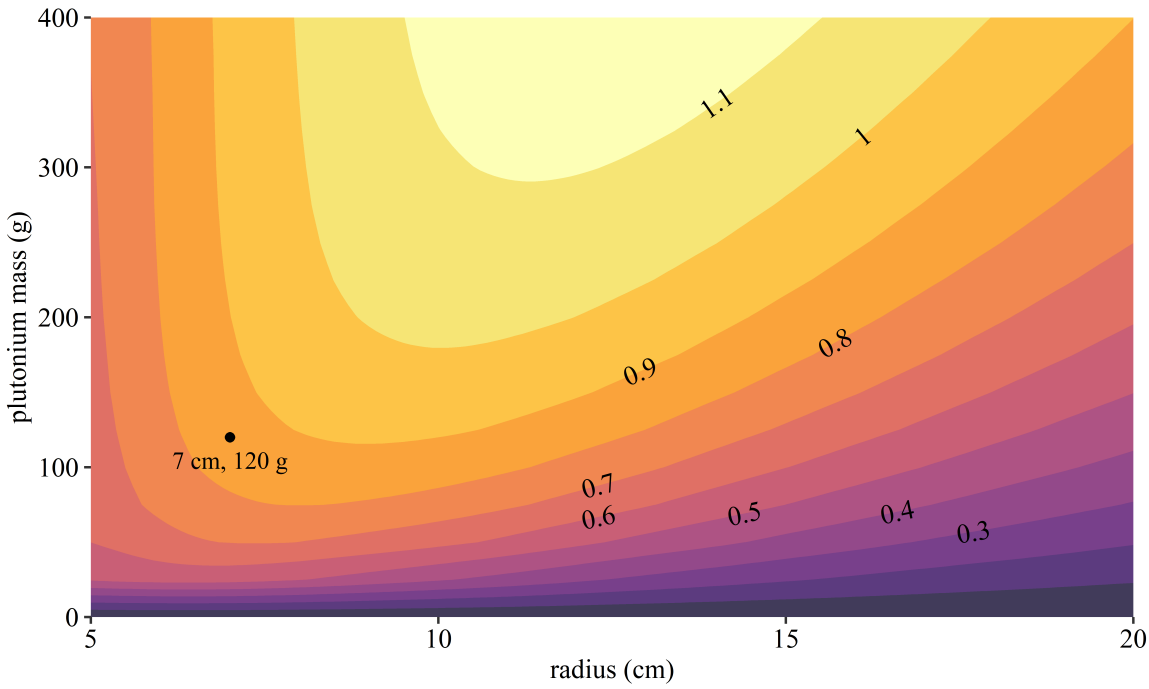

Figure 3.34: k_{eff} contour plot of 0-400 grams of alpha-phase plutonium, homogeneously dispersed in a sphere of high-density polyethylene, with six inches of beryllium reflection and one inch of water reflection

Next, the samples generated by the Bayesian network are passed to the neural network metamodel, which predicts k_{eff} values for each set of samples. Finally, the *Risk* function (Appendix A.1.5) tabulates these k_{eff} values and calculates process criticality accident risk based on the number of k_{eff} values ≥ 0.95. All of the function and subfunction calls were run in *RStudio* [114] using the *R Notebook* markdown file in Appendix A.1. An example of one of these runs is shown in Figure 3.35.

```
>
> # set variables
> code <- 'mcnp'
> source.dir <- 'D:/Dropbox/GWU/R/source'
> data.dir <- 'E:/data'
>
> # load function
> source(paste0(source.dir, '/tabulate.R'))
>
> # tabulate data
> dataset <- Tabulate(code, source.dir, data.dir)

Loaded mcnp-dataset.RData

> # set variables
> batch.size <- 8192
> ensemble.size <- 15
> epochs <- 1500
> layers <- '8192-256-256-256-256-16'
> loss <- 'sse'
> opt.alg <- 'adamax'
> learning.rate <- 0.00075
> replot <- TRUE
> test.dir <- paste0('E:/test/', loss)
> val.split <- 0.2
>
> # load function
> source(paste0(source.dir, '/nn.R'))
>
> # build metamodel
> metamodel <- NN(dataset, batch.size, ensemble.size, epochs, layers, loss, opt.alg,
  ↪ learning.rate, replot, val.split, source.dir, test.dir)

Mean Training MAE = 0.000273
Mean Cross-Validation MAE = 0.000300
Mean Test MAE = 0.000305

Ensemble Test MAE = 0.000257857
Ensemble Test MAE = 0.000257822 (Nelder-Mead)
Ensemble Test MAE = 0.000257822 (BFGS)
Ensemble Test MAE = 0.000257822 (SA)

> # set variables
> dist <- 'log-normal'
> risk.pool <- 100
> dist.dir <- 'E:/dist'
>
> if (dist == 'normal') {
+   sample.size <- 1e+10
+ } else if (dist == 'gamma' || dist == 'weibull') {
+   sample.size <- 1e+08
+ } else if (dist == 'log-normal') {
+   sample.size <- 1e+07
+ }
>
> # load functions
> source(paste0(source.dir, '/bn.R'))
> source(paste0(source.dir, '/risk.R'))
>
> # build Bayesian network
> bn <- BN(dist, dist.dir)
>
> # estimate risk
> risk <- Risk(bn, dataset, metamodel, risk.pool, sample.size, source.dir, test.dir)

Risk = 9.99e-05
Variance = 1.14e-11
```

Figure 3.35: *R Notebook* run

Chapter 4: Results

This chapter discusses the results of applying the methodology to fissile material operations in Building 332. A preliminary estimate of process criticality accident risk was made by running one million simulations based on truncated gamma distributions (Figure 4.1).

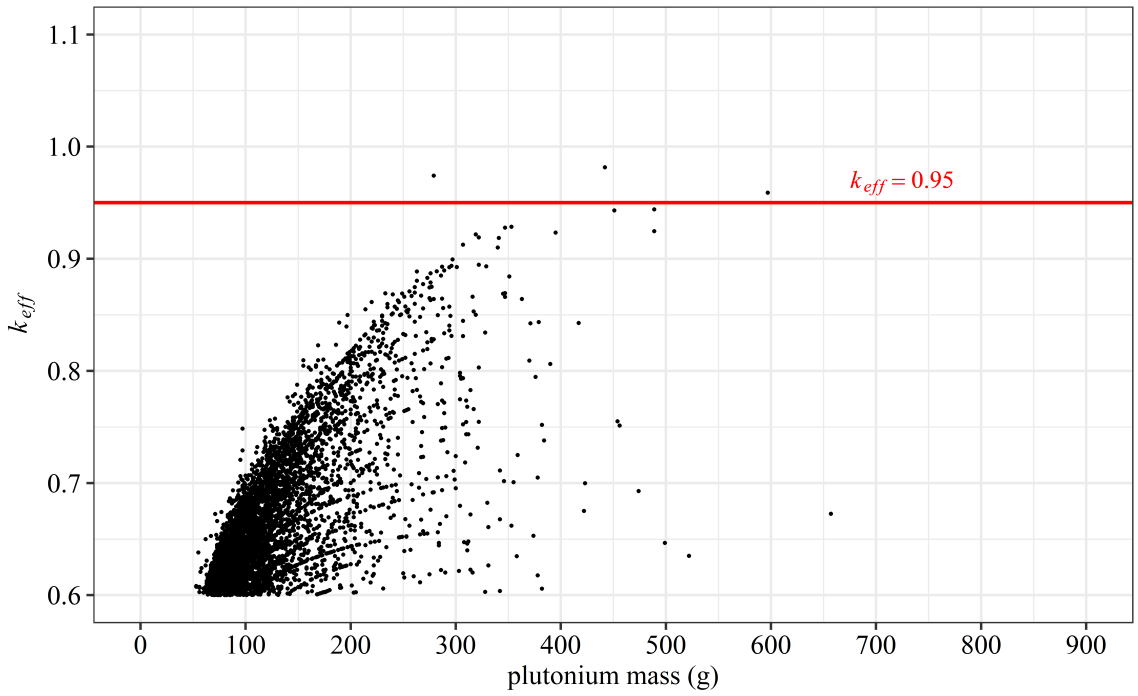

Figure 4.1: One million simulations based on truncated gamma distributions

The results shown in Figure 4.1 indicate a process criticality accident risk estimate of 3e-06 accidents per year, which was calculated by dividing the number of k_{eff} values ≥ 0.95 by the total number of simulations. To confirm these results and reduce the variance (v) of this risk estimate and others, a total of 1e+09 to 1e+12 simulations were run based on truncated gamma, normal, log-normal, and Weibull distributions (Table 4.1, Figure 4.2).

The results shown in Table 4.1 and Figure 4.2 indicate process criticality accident risk

Table 4.1: Process criticality accident risk estimates

distribution	risk	variance (v)	simulations per batch	batches
gamma	4.09e-06	4.25e-14	1e+08	100
normal	2.70e-09	6.06e-19	1e+10	100
log-normal	9.99e-05	1.14e-11	1e+07	100
Weibull	8.24e-07	8.75e-15	1e+08	100

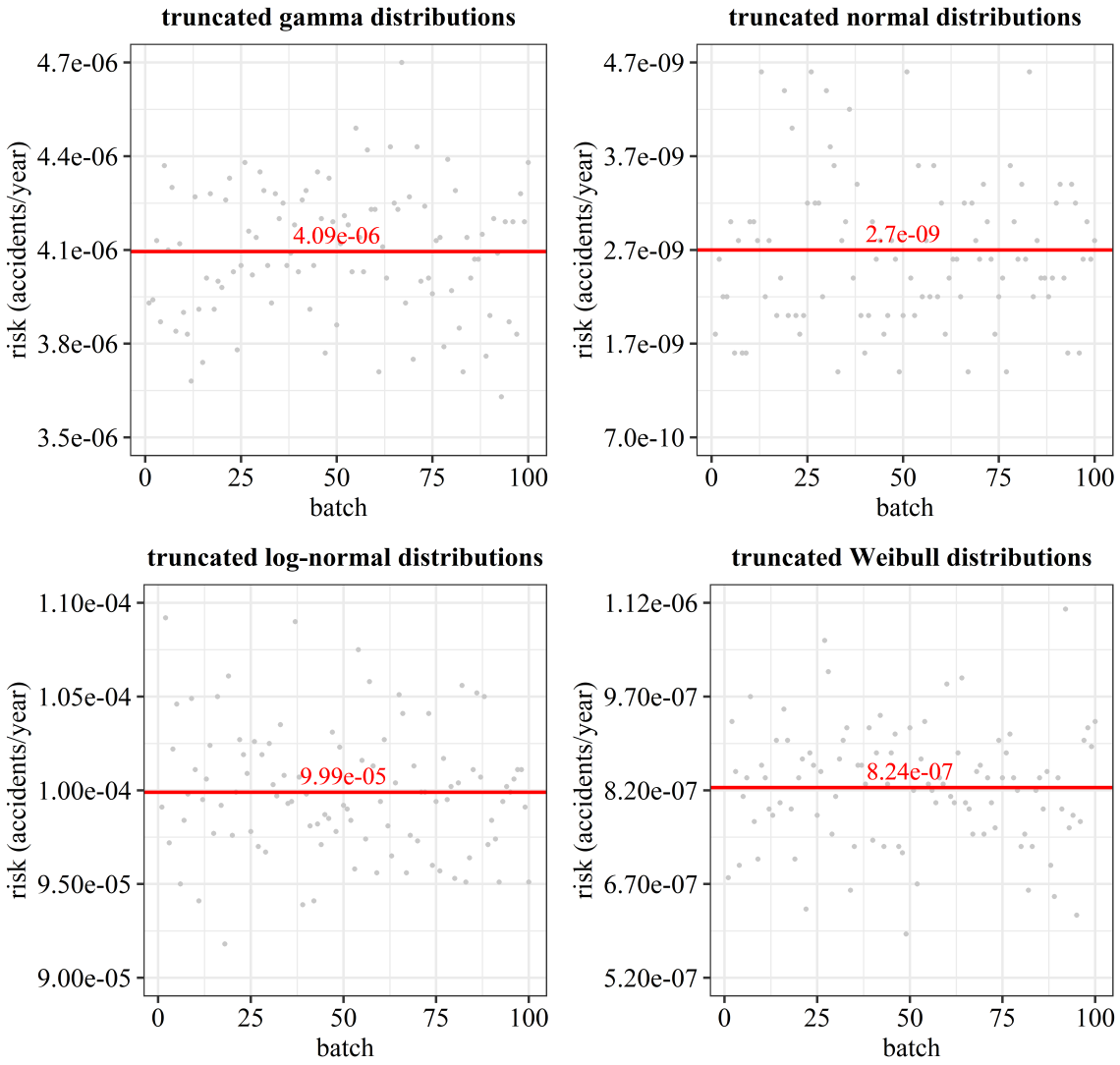

Figure 4.2: Process criticality accident risk estimates

estimates that vary between 9.99e-05 and 2.70e-09 accidents per year. The wide variation in these risk estimates is due to the truncated probability distribution tails, which differ by several orders of magnitude. The truncated probability distribution tails for plutonium mass are shown in Table 4.2. These probabilities were obtained by summing the marginal probabilities, which were calculated using the *gRain* [66] software package.

Table 4.2: Truncated probability distribution tails for plutonium mass

$P(x)$	gamma	normal	log-normal	Weibull
0-100 g	9.07e-01	8.92e-01	8.80e-01	9.13e-01
100-200 g	8.01e-02	9.97e-02	8.96e-02	7.73e-02
200-300 g	1.25e-02	6.72e-03	1.96e-02	1.05e-02
300-400 g	2.17e-03	5.85e-05	6.72e-03	1.10e-03
400-500 g	3.75e-04	4.68e-08	2.90e-03	8.10e-05
500-600 g	6.34e-05	3.01e-12	1.44e-03	4.37e-06
600-700 g	1.05e-05	1.48e-17	7.85e-04	1.78e-07
700-800 g	1.72e-06	5.48e-24	4.59e-04	5.61e-09
800-900 g	2.77e-07	1.51e-31	2.82e-04	1.40e-10
0.9-1 kg	4.44e-08	3.08e-40	1.82e-04	2.80e-12
1-2 kg	8.34e-09	4.61e-50	4.01e-04	4.56e-14
2-3 kg	6.96e-17	8.44e-211	2.86e-05	6.12e-16
3-4 kg	4.90e-25	0.00e+00	4.79e-06	6.93e-18

The truncated probability distribution tails for plutonium mass (Table 4.2) are relatively stable for the truncated gamma and log-normal distributions, although the truncated gamma distributions drop off sharply at 2-3 kilograms of plutonium. This sharp drop at 2-3 kilograms is justifiable because there is a 2-kilogram facility-wide limit on plutonium carbides, oxides,

nitrates, and solutions $\geq$ 25 g/L [16]. Similiarly, the truncated normal distributions drop off sharply at 500-600 grams of plutonium. However, this is somewhat problematic because there is no facility-wide limit or explanation that justifies this drop. For example, if there are two adjacent workstations in the same room, and each workstation contains $\geq$ 250 grams of plutonium, it strains credibility to suggest that there is only a 3.01e-12 per year probability that 250 grams of plutonium will (accidentally) be transferred from one workstation to the other. Realistically, this probability should be much higher.

Based on the results shown in Figure 4.2 and Table 4.2, the truncated normal and Weibull distributions were determined to be nonconservative because they have light tails (Table 4.2), which result in lower risk estimates. Conversely, the truncated gamma and log-normal distributions have heavy tails (Table 4.2), which result in higher risk estimates. The truncated log-normal distribution test statistics (Figure 3.6) are notable because they fit the data much worse than the other three probability distributions. This may be an indication that fissile material operations in Building 332 are following a slighly nonconservative or lower risk trend. Although not definitive, these test statistics are useful for performing graphical checks of Q-Q plots to determine if fissile material operations are trending toward higher or lower risk fits, which can serve as a leading indicator of recurrent operational failures—similar to those described in R.J. Mattson [83] and J.N. McKamy [85]. These checks can also be used to identify nonconservative outliers that would otherwise be subsumed by the rest of the data.

The mass, radius, volume, and concentration of each simulated accident are plotted and shown in Figure 4.3 and Figure 4.4 for the truncated gamma and log-normal distributions. The mean plutonium masses from these plots are 452 and 625 grams, respectively, which

are similar to several of the historical process criticality accidents shown in Table 4.3 [86]. The mean volumes from these plots are 12.01 and 15.12 liters, respectively, which are lower than the volumes shown in Table 4.3 [86]. This disparity exists because there are currently no large tanks in Building 332. In comparison, 21 out of 22 historical process criticality accidents occurred in large tanks with "unsafe" or "unfavorable" geometries (i.e., large tanks with height-to-diameter ratios close to 1) [86].

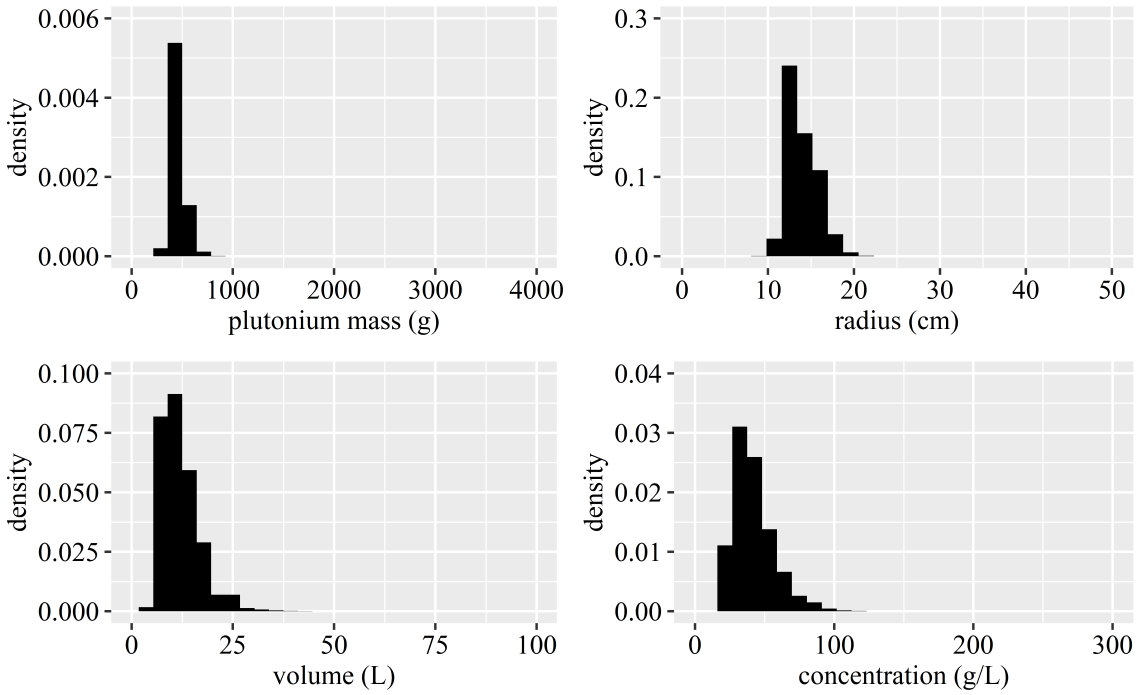

Figure 4.3: Process criticality accident data based on truncated gamma distributions

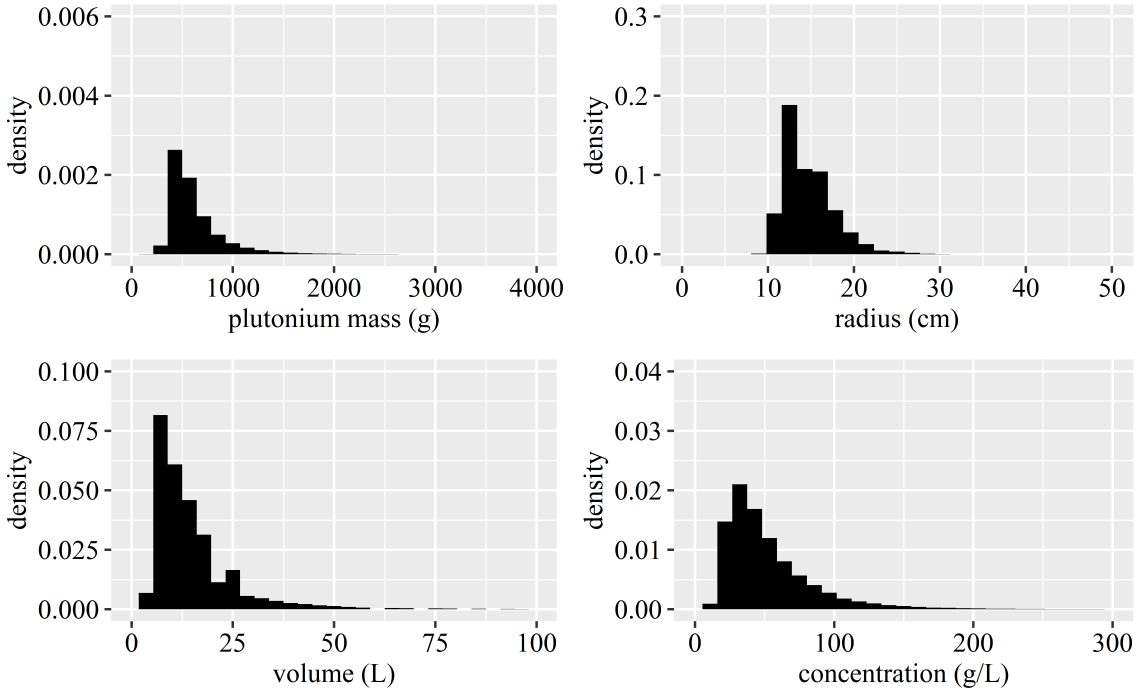

Figure 4.4: Process criticality accident data based on truncated log-normal distributions

Table 4.3: Historical process criticality accidents

number [1]	mass (g) [2]	volume (L)	plutonium	uranium	metal	solution
1	670	31	✓			✓
2	1,090	30		✓		✓
3	5,240	58		✓		✓
4	1,500	56		✓		✓
5	840	160	✓			✓
6	1,240	800		✓		✓
7	710	19	✓			✓
8	2,480	40		✓		✓
9	1,290	43		✓		✓
10	1,070	45	✓			✓
11	1,050	80	✓			✓
12	2,060	36		✓		✓
13	1,380	65		✓		✓
14	1,720	41		✓		✓
15	2,610	100		✓		✓
16	1,650	29		✓		✓
17	1,360	29	✓			✓
18	690	40	✓			✓
19	4,340	316		✓		✓
20	9,180	< 1	✓		✓	
21	unknown	unknown		✓		✓
22	2,900	45		✓		✓

[1] numbers were taken from *A Review of Criticality Accidents* [86]

[2] estimated spherical critical mass

Between the late 1960s and early 1980s, most DOE sites replaced unfavorable geometry tanks with tanks that are longer and thinner, which creates a larger surface area for neutrons to leak out of the system [86]. These "favorable" geometry tanks are not present in Building 332, which allows the methodology to default to spherical geometry. However, if favorable geometry tanks were used in Building 332, there are two approaches to modeling them.

The first approach is to use geometric buckling, which is essentially a set of equations that converts from one type of critical geometry to another [55]. The equations to convert from cylindrical geometry to spherical geometry are shown in 4.1-4.4 [55].

$$\text{cylinder} \quad B_g^2 = \frac{\pi^2}{(h+2\lambda)^2} + \frac{(2.405)^2}{(r+\lambda)^2} \tag{4.1}$$

where:

B_g^2 = geometric buckling

h = height (cm)

r = radius (cm)

λ = extrapolation distance (cm) [95]

$$\text{sphere} \quad B_g^2 = \frac{\pi^2}{(r+\lambda)^2} \tag{4.2}$$

where:

B_g^2 = geometric buckling

r = radius (cm)

λ = extrapolation distance (cm) [95]

4.3 sets the geometric buckling equations (4.1, 4.2) equal to each other, and 4.4 rearranges 4.3 to solve for the radius of sphere (r_{sph}).

$$\frac{\pi^2}{(h_{cyl}+2\lambda)^2}+\frac{(2.405)^2}{(r_{cyl}+\lambda)^2}=\frac{\pi^2}{(r_{sph}+\lambda)^2} \tag{4.3}$$

$$r_{sph}=\pi\left[\frac{\pi^2}{(h_{cyl}+2\lambda)^2}+\frac{(2.405)^2}{(r_{cyl}+\lambda)^2}\right]^{-1/2}-\lambda \tag{4.4}$$

The main problem with geometric buckling is that it does not take into account different reflector materials [55, 95]. This makes it difficult to determine the extrapolation distance (λ), which is an empirically derived value that typically varies between 2 and 5 [95]. The second approach is to add cylindrical geometry and height-to-diameter ratios to the training parameters (Table 3.7) and perform additional MCNP calculations. However, this approach potentially requires millions of additional MCNP calculations for varying cylinder heights and radii, which could take several months to perform.

The risk estimates shown in Table 4.1 and Figure 4.2 are lower than the previous risk estimate of 3.4e-05 accidents per year, which was calculated using FTA [125]. The disparity between these risk estimates is partially due to the removal of most of the fissile material from Building 332 between 2008 and 2012 [103]—as well as the downgrade from Security Category I/II to Security Category III operations, which places more restrictive limits on the quantities and types of fissile material that are allowed in Building 332 [16]. However, these explanations are somewhat superficial. The differences between FTA and the methodology described in this dissertation are distinct and fundamental. For example, the minimal cut

set for Building 332 (Figure 4.5) assigns 73.5% of the risk to one accident scenario (2.5e-05/3.4e-05 = 0.735). In comparison, the risk estimates shown in Figure 4.2 are based on millions of simulated accident and non-accident scenarios, most of which involve $\leq$ 500 grams of plutonium.

The results shown in Table 4.2 also suggest 3.58e-03 and 7.59e-05 per year probabilities that $\geq$ 500 grams of plutonium will exist in a workstation (Table 4.2), which are much lower than the probability of spilling $>$ 500 grams of powder/slurry/solution (Figure 4.5). The minimal cut set for Building 332 (Figure 4.5) can also be easily (and alarmingly) invalidated. For example, if excess moderation or reflection is accidentally introduced into a workstation once or twice a year, the minimal cut set defaults to a value of 0.1-1, which is unrealistic. Also, the minimal cut set shown in Figure 4.5 assigns a 0.1 probability to "optimal critical conditions achieved", which doesn't make sense, considering there are an infinite number of suboptimal accident scenarios that likely have a cumulative probability $\gg$ 0.1.

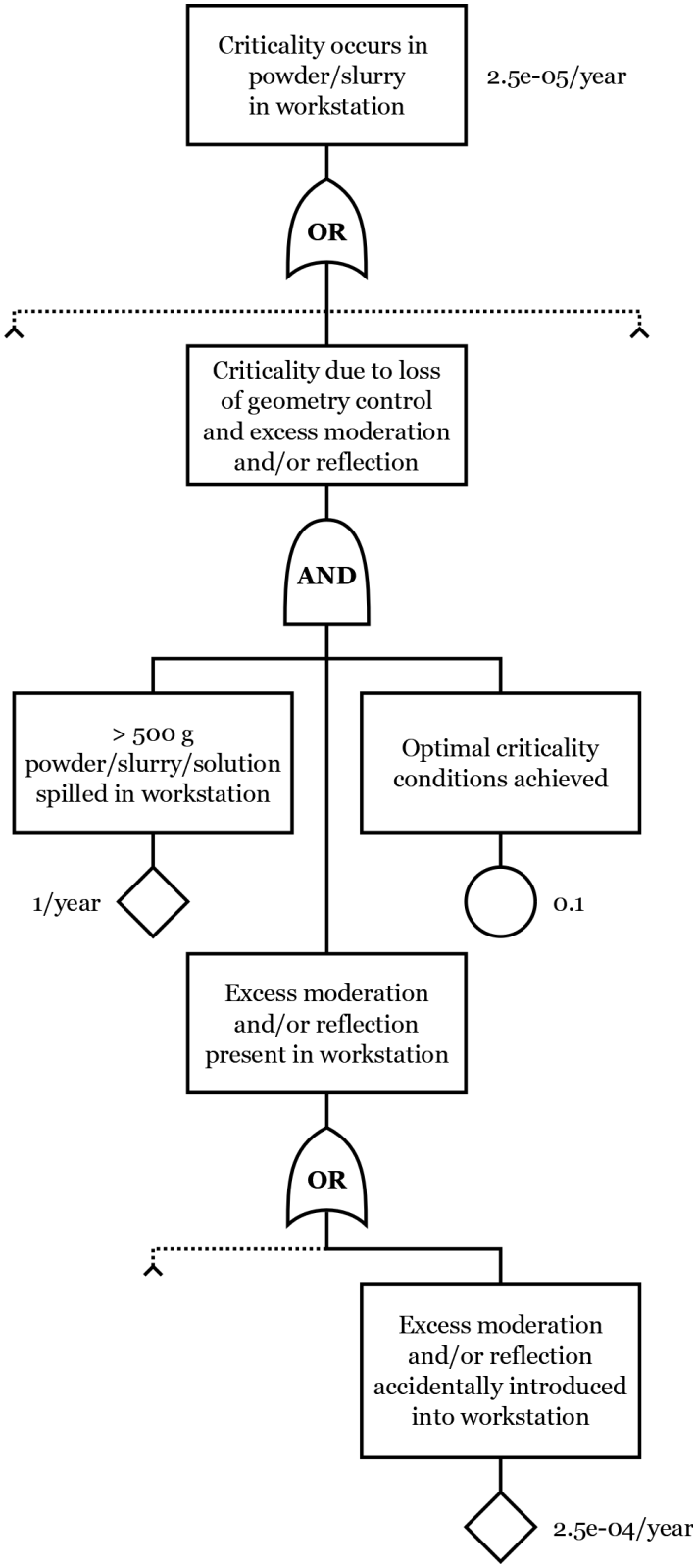

Figure 4.5: Minimal cut set for Building 332

Chapter 5: Conclusions

This chapter discusses the conclusions of the case study, the contributions this dissertation makes to the broader field of risk analysis, and opportunities for future work.

5.1 Discussion

Overall, the coupled Bayesian network and neural network metamodel worked extremely well, and the results demonstrate that the following objectives were achieved:

1. Develop a process for training neural networks to predict k_{eff} to high precision (✓)
2. Develop a methodology that uses a coupled Bayesian network and neural network metamodel to estimate process criticality accident risk (✓)
3. Establish trade-offs between neural network performance and training time (✓)
4. Compare process criticality accident risk estimates to other sources of data, such as critical mass studies and historical process criticality accidents (✓)

The process for training neural networks was performed manually. The maximum error for an ensemble of 15 neural networks was $< |0.004|$, and the mean absolute errors on training and test data were 2.26e-04 and 2.58e-04, respectively (Figure 3.31, Figure 3.32). These values are notable because they are less than the mean MCNP standard deviation (σ) of 2.70e-04 (Figure 3.11), which demonstrates that the first objective of training neural networks to predict k_{eff} to high precision was achieved. It also only took approximately 22.5 hours to train the neural network metamodel, which demonstrates that the third objective of establishing trade-offs between neural network performance and training time was achieved.

Using the new methodology, the risk of a process criticality accident in Building 332 was estimated to be between 9.99e-05 and 2.70e-09 accidents per year (Figure 4.2), which demonstrates that the second objective of using a coupled Bayesian network and neural network metamodel to estimate process criticality accident risk was achieved. These results include both conservative and nonconservative risk estimates that are consistent with critical mass studies [95, 115, 116] and historical process criticality accidents [86], which demonstrates that the fourth objective of comparing process criticality accident risk estimates to other sources of data was achieved. Although these risk estimates are specific to Building 332, the methodology described in this dissertation can easily be applied to fissile material operations in any nuclear or radiological facility.

5.2 Contributions

This dissertation makes two key contributions to the broader field of risk analysis. The first contribution is the development of a manual process for training neural networks to predict a latent variable to high precision. This process is based on empirical modeling that was performed throughout this work, as well as recommendations from the books *Deep Learning* [59], *Deep Learning with R* [44], and *Applied Predictive Modeling* [78]. The one notable deviation from these books was the use of a single neural network as the baseline for the metamodel. *Deep Learning with R* [44] recommends using a more diverse set of models, such as combining neural networks with random forests [81] and support vector machines [34, 113], with the idea being that averaging a diverse set of models is better than averaging just one type of model. This "multi-model" approach was initially tested using the

caret [77] software package. However, it was quickly abandoned because the out-of-the-box performance of the other models was poor, and the inability to use a GPU meant that each model would have taken several weeks to train.

The second contribution is the application of the SSE loss function to a neural network regression problem. During the early stages of this work, several different combinations of learning rates, loss functions, and optimization algorithms were tested, and a large amount of instability was observed with some of the newer optimization algorithms, such as Adam and Adamax [73]. An investigation of this instability led deep into the *TensorFlow* [27] source code, where it was discovered that the MSE loss function gives parameter (θ) updates equal weight, regardless of batch size. The SSE loss function was initially substituted into the *NN* function (Appendix A.1.3) to determine if the MSE loss function was the cause of the instability.

Several neural networks were trained using the SSE loss function, and the instability was still observed, but the mean absolute errors on training and cross-validation data were noticeably lower. After sharing these results with other code developers and scientists, a test was proposed to randomly remove training samples and track the mean absolute errors on training and cross-validation data, with the idea being that if *last batch bias* did exist, then the neural networks that were trained using the MSE loss function would perform poorly, and the neural networks that were trained using the SSE loss function would be unaffected—both of which occurred (Figure 3.27, Figure 3.28). These results were communicated to members of the *TensorFlow* team at Google, and the error was fixed (https://github.com/tensorflow).

5.3 Future Work

This section provides additional objectives and opportunities for future work, which are described below.

- Build a function for the nuclear criticality slide rule [56, 68] and merge it with the existing model
- Add cylindrical geometry to the model
- Apply this methodology to other nuclear and radiological facilities and compare the results

Achieving the first objective will enable the model to estimate total fission yield and radiation doses, both of which are important for consequence analysis. The second objective can be achieved by reanalyzing facility data, adding cylindrical geometry and height-to-diameter ratios to the training parameters (Table 3.7), and performing additional MCNP calculations. Achieving the second and third objectives will extend this methodology to other sites and facilities and enable additional improvements to be made to the methodology.

Bibliography

[1] Accidental Radiation Excursion at the Y-12 Plant. Union Carbide Nuclear Company, Y-1234, 1958.

[2] Site-Wide Environmental Impact Statement for the Rocky Flats Plant Site. U.S. Department of Energy, National Nuclear Security Administration, DOE/EIS-0064, 1980.

[3] Site-Wide Environmental Impact Statement for the Pantex Plant Site. U.S. Department of Energy, National Nuclear Security Administration, DOE/EIS-0098, 1983.

[4] Site-Wide Environmental Impact Statement for Continued Operation of Los Alamos National Laboratory. U.S. Department of Energy, National Nuclear Security Administration, DOE/EIS-0238, 1999.

[5] Site-Wide Environmental Impact Statement for Continued Operation of Sandia National Laboratories, New Mexico. U.S. Department of Energy, National Nuclear Security Administration, DOE/EIS-0281, 1999.

[6] Recommendations for Analyzing Accidents Under the National Environmental Policy Act. U.S. Department of Energy, 2002.

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

[8] 10 CFR § 830, Safety Basis Requirements. U.S. Office of the Federal Register, 2011.

[9] Site-Wide Environmental Impact Statement for the Y-12 National Security Complex. U.S. Department of Energy, National Nuclear Security Administration, DOE/EIS-0387, 2011.

[10] 40 CFR § 1502, Environmental Impact Statement. U.S. Office of the Federal Register, 2012.

[11] ANSI/HPS N13.3-2013, Dosimetry for Criticality Accidents. American National Standards Institute, Health Physics Society, 2013.

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

[13] Site-Wide Environmental Impact Statement for Continued Operation of Nevada

National Security Site and Off-Site Locations in the State of Nevada. U.S. Department of Energy, National Nuclear Security Administration, DOE/EIS-0426, 2013.

[14] ANSI/ANS-8.1-2014, Nuclear Criticality Safety in Operations with Fissionable Materials Outside Reactors. American Nuclear Society, 2014.

[15] DOE-STD-3009-2014, Preparation of Nonreactor Nuclear Facility Documented Safety Analysis. U.S. Department of Energy, 2014.

[16] DOE O 474.2, Nuclear Material Control and Accountability. U.S. Department of Energy, 2016.

[17] DOE-STD-1104-2016, Review and Approval of Nuclear Facility Safety Basis and Safety Design Basis Documents. U.S. Department of Energy, 2016.

[18] ANSI/ANS-8.24-2017, Validation of Neutron Transport Methods for Nuclear Criticality Safety. American Nuclear Society, 2017.

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

[20] International Criticality Safety Benchmark Evaluation Project Handbook. Nuclear Energy Agency, NEA-7497, 2019.

[21] ANSI/ANS-6.1.1-2020, Photon and Neutron Fluence-to-Dose Conversion Coefficients. American Nuclear Society, 2020.

[22] Site-Wide Environmental Impact Statement for Plutonium Pit Production at the Savannah River Site. U.S. Department of Energy, National Nuclear Security Administration, DOE/EIS-0541, 2020.

[23] Work Planning and Control Assessment at Lawrence Livermore National Laboratory. U.S. Department of Energy, Office of Enterprise Assessments, 2020.

[24] International Nuclear and Radiological Event Scale (INES). International Atomic Energy Agency, 2021.

[25] Rare events. https://en.wikipedia.org/wiki/rare_events, 2021.

[26] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D.G. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine

learning on heterogeneous systems, 2016.

[27] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D.G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: A system for large-scale machine learning. USENIX, 2016.

[28] R. Alvarez. Y-12: Poster child for a dysfunctional nuclear weapons complex. Bulletin of the Atomic Scientists, August 2014.

[29] 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.

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

[31] C.L. Atwood, J.L. LaChance, H.F. Martz, D.J. Anderson, M. Englehardt, D. Whitehead, and T. Wheeler. *Handbook of Parameter Estimation for Probabilistic Risk Assessment*. U.S. Nuclear Regulatory Commission, NUREG/CR-6823, 2003.

[32] F.Y. Barbry. A Review of the SILENE Criticality Excursions Experiments. Institut de Protection et de Sûreté Nucléaire (IRSN), 1993.

[33] C.J.P. Bélisle. Convergence theorems for a class of simulated annealing algorithms on R^d. *Journal of Applied Probability*, 1992.

[34] K.P. Bennett and A. Demiriz. Semi-supervised support vector machines. *Proceedings of Neural Information Processing Systems*, 1999.

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

[36] 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.

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

[38] 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.

[39] C.C. Cappiello, K.B. Butterfield, R.G. Sanchez, J.A. Bounds, R.H. Kimpland, R.P.

Damjanovich, and P.J. Jaegers. Solution High-Energy Burst Assembly (SHEBA) Results from Subprompt Critical Experiments with Uranyl Fluoride Fuel. Los Alamos National Laboratory, LA-13373-MS, 1997.

[40] R.D. Carter, G.R. Kiel, and K.R. Ridgway. *Criticality Handbook, Volumes I-III*. Atlantic Richfield Hanford Company, 1968.

[41] M.B. Chadwick. Fission product yields from fission spectrum n+^{239}Pu for ENDF/B-VII.1. *Nuclear Data Sheets*, 2011.

[42] M.B. Chadwick, M. Herman, P. Obložinský, M.E. Dunn, Y. Danon, A.C. Kahler, D.L. Smith, B. Pritychenko, G. Arbanas, R. Arcilla, R. Brewer, D.A. Brown, R. Capote, A.D. Carlson, Y.S. Cho, H. Derrien, K. Guber, G.M. Hale, S. Hoblit, S. Holloway, T.D. Johnson, T. Kawano, B.C. Kiedrowski, H. Kim, S. Kunieda, N.M. Larson, L. Leal, J.P. Lestone, R.C. Little, E.A. McCutchan, R.E. MacFarlane, M. MacInnes, C.M. Mattoon, R.D. McKnight, S.F. Mughabghab, G.P.A. Nobre, G. Palmiotti, A. Palumbo, M.T. Pigni, V.G. Pronyaev, R.O. Sayer, A.A. Sonzogni, N.C. Summers, P. Talou, I.J. Thompson, A. Trkov, R.L. Vogt, S.C. van der Mark, A. Wallner, M.C. White, D. Wiarda, and P.G. Young. ENDF/B-VII.1 Nuclear Data for Science and Technology: Cross Sections, Covariances, Fission Product Yields, and Decay Data. *Nuclear Data Sheets*, 2011.

[43] F. Chollet. Keras. https://keras.io, 2021.

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

[45] E.D. Clayton. *Anomalies of Nuclear Criticality*. Pacific Northwest National Laboratory, 2010.

[46] S.E. Coleman and W.J. Zywiec. Validation of MCNP6 and ENDF/B-VII.1 for General Application to Plutonium and Highly Enriched Uranium Systems. Lawrence Livemore National Laboratory, LLNL-CONF-777543, 2019.

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

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

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

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

[51] M.L. Delignette-Muller and C. Dutang. fitdistrplus: An R package for fitting distributions. *Journal of Statistical Software*, 2015.

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

[53] P. Diaconis and F. Mosteller. Methods for studying coincidences. *Journal of the American Statistical Association*, 1989.

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

[55] J.J. Duderstadt and L.J. Hamilton. *Nuclear Reactor Analysis*. John Wiley & Sons, Inc., 1976.

[56] M. Duluc, D.P. Heinrichs, Kim. S.S., T.M. Miller, C. Celik, C.M. Hopper, A. Brown, C. Wilson, and M. Troisne. Introduction of Plutonium Systems to the Nuclear Criticality Slide Rule. U.S. Department of Energy, National Nuclear Security Administration, 2017.

[57] E. Fermi. The development of the first chain reacting pile. *Proceedings of the American Philosophical Society*, 1946.

[58] R. Fletcher and C.M. Reeves. Function minimization by conjugate gradients. *Computer Journal*, 1967.

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

[60] L.E. Gordon-Hagerty. Preliminary Notice of Violation, Consolidated Nuclear Security, LLC. U.S. Department of Energy, National Nuclear Security Administration, April 2020.

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

[62] B. Hamilton. Los Alamos National Laboratory Nuclear Criticality Safety Program. Defense Nuclear Facilities Safety Board, November 2018.

[63] B. Hamilton. Y-12 Nuclear Criticality Safety Program. Defense Nuclear Facilities Safety Board, July 2019.

[64] H.O. Hartley. Smallest composite designs for quadratic response functions. *Biomet-*

rics, 1959.

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

[66] S. Højsgaard. Graphical independence networks with the grain package for R. *Journal of Statistical Software*, 2012.

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

[68] C.M. Hopper and B.L. Broadhead. An Updated Nuclear Criticality Slide Rule. Oak Ridge National Laboratory, ORNL/TM-13322, 1998.

[69] M.A. Jette and J.D. Auble. Slurm. https://slurm.schedmd.com, 2021.

[70] T.E. Johnson. *Introduction to Health Physics*. McGraw-Hill Companies, Inc., 5th edition, 2017.

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

[72] 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.

[73] D.P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *International Conference for Learning Representations*, 2011.

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

[75] D. Koller and N. Friedman. *Probabilistic Graphical Models*. The MIT Press, 2009.

[76] F.E. Kruesi, J.O. Erkman, and D.D. Lanning. Critical Mass Studies of Plutonium Solutions. Hanford Works, HW-24514 DEL, 1952.

[77] M. Kuhn. Building predictive models in R using the caret package. *Journal of Statistical Software*, 2008.

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

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

[80] S.L. Lauritzen and D. Spiegelhalter. Local computations with probabilities on graphi-

cal structures and their application to expert systems. *Journal of the Royal Statistical Society*, 1988.

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

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

[83] R.J. Mattson. An Assessment of Criticality Safety at the Rocky Flats Plant. Scientech, Inc., 1989.

[84] A. McCarthy and D. Kupferer. Uranium Accumulation Discoveries at Y-12. Y-12 National Security Complex, August 2018.

[85] J.N. McKamy. Lessons Learned from the Rocky Flats Building 771 Near Miss Criticality Accident. U.S. Department of Energy, National Nuclear Security Administration, 2014.

[86] T.P. McLaughlin, S.P. Monahan, N.L. Pruvost, V.V. Frolov, B.G. Ryazanov, and V.I. Sviridov. *A Review of Criticality Accidents*. Los Alamos National Laboratory, LA-13638, 2000.

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

[88] M. Murazaki, K. Tonoike, and G. Uchiyama. Measurement of Neutron Dose Under Criticality Accident Conditions at TRACY Using TLDs. Japan Atomic Energy Research Institute, 2012.

[89] K. Nakajima, Y. Yamane, K. Ogawa, E. Aizawa, H. Yanagisawa, and Y. Miyoshi. TRACY Transient Experiment Databooks. Japan Atomic Energy Research Institute, 2002.

[90] J.A. Nelder and R. Mead. A simplex algorithm for function minimization. *Computer Journal*, 1965.

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

[92] R.S. Olson and M.A. Robkin. A new small mass critical configuration. *Transactions of the American Nuclear Society*, 1970.

[93] M. Papadrakakis and N.D. Lagaros. Reliability-based structural optimization us-

ing neural networks and Monte Carlo simulation. *Computer Methods in Applied Mechanics and Engineering*, 2002.

[94] H.C. Paxton. A History of Critical Experiments at Pajarito Site. Los Alamos National Laboratory, LA-9685-H, 1983.

[95] H.C. Paxton and N.L. Pruvost. Critical Dimensions of Systems Containing ^{235}U, ^{239}Pu, and ^{233}U. Los Alamos National Laboratory, LA-10860-MS, 1986.

[96] J. Pearl. Bayesian networks: A model of self-activated memory for evidential reasoning. *Conference of the Cognitive Science Society*, 1985.

[97] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann Publishers, Inc., 2nd edition, 1988.

[98] C.M. Percher and D.P. Heinrichs. An Experimental Study of the Effect of Operator Hands On the Reactivity of a Fast Metal System. Lawrence Livermore National Laboratory, LLNL-CONF-491345, 2011.

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

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

[101] S. Rebello, H. Yu, and L. Ma. An integrated approach for real-time hazard mitigation in complex industrial processes. *Reliability Engineering and System Safety*, 2019.

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

[103] D.C. Riley. Overview of the De-Inventory Effort at Lawrence Livermore National Laboratory. Lawrence Livermore National Laboratory, LLNL-CONF-638972, 2013.

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

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

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

[107] M. Scutari. Learning Bayesian networks with the bnlearn R package. *Journal of*

Statistical Software, 2010.

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

[109] R.M. Sinnamon and J.D. Andrews. Fault tree analysis and binary decision diagrams. *Proceedings of 1996 Annual Reliability and Maintainability Symposium*, 1996.

[110] R.J. Smith and P. Malone. Safety problems at Los Alamos National Laboratory delay U.S. nuclear warhead testing and production. The Center for Public Integrity, June 2017.

[111] A.J. Smola and P. Bartlett. Sparse greedy Gaussian process regression. *Advances in Neural Information Processing Systems*, 2001.

[112] B. Sudret. Global sensitivity analysis using polynomial chaos expansions. *Reliability Engineering and System Safety*, 2008.

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

[114] RStudio Team. RStudio: Integrated Development for R. https://www.rstudio.com, 2021.

[115] J.T. Thomas. Nuclear Safety Guide, 1978.

[116] J.K. Thompson. Minimum critical mass of plutonium-polyethylene system found to be significantly lower than plutonium-water system. *Nuclear Technology*, 1977.

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

[118] 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.

[119] D.G. Vasilik, R.W. Martin, and D. Fuller. Nuclear Accident Dosimetry: Measurements at the Los Alamos SHEBA Critical Assembly. Los Alamos Scientific Laboratory, LA-8911-MS, 1981.

[120] H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer, 2016.

[121] P.S. Winokur. Los Alamos National Laboratory Nuclear Criticality Safety Program. Defense Nuclear Facilities Safety Board, July 2013.

[122] P.S. Winokur. Los Alamos National Laboratory Nuclear Criticality Safety Program. Defense Nuclear Facilities Safety Board, May 2014.

[123] K.R. Yates. Criticality of thin flat foils versus spherical shells of ^{239}Pu. *Transactions of the American Nuclear Society*, 1977.

[124] W.J. Zywiec and A.J. Nelson. Redux analysis of D_2O reflected plutonium foils at low temperature. *Transactions of the American Nuclear Society*, 2019.

[125] W.J. Zywiec, S. Sarkani, and T.A. Mazzuchi. A Bayesian network-based framework for probabilistic risk assessment. *Transactions of the American Nuclear Society*, 2018.

Appendix A: Code

A.1 R Notebook

````
---
title : 'R Notebook'
output : html_document
---

# notebook.Rmd
#
# William John Zywiec
# The George Washington University

```{r, include = FALSE, warning = FALSE}

# initialize environment
if (!is.null(dev.list())) dev.off()
rm(list = ls())
cat('\014')

```

```{r, setup, include = FALSE, warning = FALSE}

# set variables
code <- 'mcnp'
source.dir <- 'D:/Dropbox/GWU/R/source'
data.dir <- 'E:/data'

```

```{r, warning = FALSE}

# load function
source(paste0(source.dir, '/tabulate.R'))

# tabulate data
dataset <- Tabulate(code, source.dir, data.dir)

```

```{r, warning = FALSE}

# set variables
batch.size <- 8192
ensemble.size <- 15 # 15
epochs <- 1500 # 1500
layers <- '8192-256-256-256-256-16'
loss <- 'sse'
opt.alg <- 'adamax'
learning.rate <- 0.00075
replot <- TRUE
test.dir <- paste0('E:/test/', loss)
val.split <- 0.2

# load function
source(paste0(source.dir, '/nn.R'))

# build metamodel
metamodel <- NN(dataset, batch.size, ensemble.size, epochs, layers, loss, opt.alg,
 ↪ learning.rate, replot, val.split, source.dir, test.dir)

```

```{r, warning = FALSE}

# set variables
dist <- 'normal'
````
````

```
risk.pool <- 100
dist.dir <- 'E:/dist'

if (dist == 'normal') {
  sample.size <- 1e+10
} else if (dist == 'gamma' || dist == 'weibull') {
  sample.size <- 1e+08
} else if (dist == 'log-normal') {
  sample.size <- 1e+07
}

# load functions
source(paste0(source.dir, '/bn.R'))
source(paste0(source.dir, '/risk.R'))

# build Bayesian network
bn <- BN(dist, dist.dir)

# estimate risk
risk <- Risk(bn, dataset, metamodel, risk.pool, sample.size, source.dir, test.dir)

```
```
```

A.1.1 Grid

```
# grid.R
#
# William John Zywiec
# The George Washington University

# initialize environment
if (!is.null(dev.list())) dev.off()
rm(list = ls())
cat('\014')

# load function
source('E:/Dropbox/GWU/R/source/build.R')

build.dir <- 'F:/data'

setwd(build.dir)

# set variables
mass <- seq(25, 4000, 25)
form <- c('alpha', 'puo2')
mod <- c('mgo', 'sepiolite', 'ch2', 'h2o', 'none')
rad <- seq(0, 18, 1) * 2.54
ref <- c('al', 'be', 'du', 'graphite', 'pb', 'mgo', 'ch2', 'ss304', 'h2o', 'none')
thk <- seq(0, 6, 1) * 2.54
shape <- 'sph'

# build input decks and run MCNP
df <- expand.grid(mass, form, mod, rad, ref, thk, shape, stringsAsFactors = FALSE)

for (i in 1:nrow(df)) Build(df[i, 1], df[i, 2], df[i, 3], df[i, 4], df[i, 5], df[i,
 ↪ 6], df[i, 7])
```

A.1.1.1 Build

```
# build.R
#
# William John Zywiec
# The George Washington University
#
# model parameters
# ----------------
# mass:  Pu mass (95% Pu-239, 5% Pu-240) (g)
#
# form:  alpha-phase Pu
#        delta-phase Pu
#        Pu oxide (PuO2)
#
# mod:   aluminum oxide
#        beryllium
#        beryllium oxide
#        graphite
#        magnesium oxide
#        polyethylene
#        sepiolite
#        silicon dioxide
#        water
#
# rad:   radius (cm)
#
# ref:   aluminum
#        aluminum oxide
#        beryllium
#        beryllium oxide
#        carbon steel
#        copper
#        depleted uranium
#        granite
#        graphite
#        iron
#        lead
#        magnesium oxide
#        molybdenum
#        nickel
#        niobium
#        platinum
#        polyethylene
#        stainless steel 304
#        stainless steel 304L
#        stainless steel 316
#        stainless steel 316L
#        tantalum
#        titanium
#        tungsten
#        vanadium
#        water
#
# thk:   reflector thickness (cm)
#
# shape: rcc
#        sph
#
# ht:    height (cm)

Build <- function(mass, form, mod, rad, ref, thk, shape, ht) {

  library(magrittr)
  library(parallel)

  # set format and precision
  Format <- function(x) formatC(x, format = 'e', digits = 14)

```

```
  # set Pu mass (g) and density (g/cc)
  if (form == 'alpha') {
    pu.mass <- mass
    pu.density <- 19.86
  } else if (form == 'puo2') {
    pu.mass <- mass
    pu.density <- 11.5
  }

  # calculate vol (cc)
  if (shape == 'sph') {
    vol <- 4/3 * pi * rad^3
  } else if (shape == 'rcc') {
    vol <- pi * rad^2 * ht
  }

  # fix mod, vol (cc), and rad (cm)
  if (vol <= pu.mass / pu.density) {
    mod <- 'none'
    vol <- pu.mass / pu.density
    if (shape == 'sph') {
      rad <- (3/4 * vol / pi)^(1/3)
    } else if (shape == 'rcc') {
      rad <- (vol / ht / pi)^(1/2)
    }
  }

  # fix ref and thk (cm)
  if (ref == 'none' || thk == 0) {
    ref <- 'none'
    thk <- 0
  }

  # set material density (g/cc)
  matl.density <- c(
    'al', 2.6989,
    'al2o3', 3.97,
    'be', 1.848,
    'beo', 3.01,
    'cs', 7.82,
    'cu', 8.96,
    'du', 19.0,
    'granite', 2.69,
    'graphite', 1.7,
    'fe', 7.874,
    'pb', 11.35,
    'mgo', 3.58,
    'mo', 10.22,
    'ni', 8.902,
    'nb', 8.57,
    'pt', 21.45,
    'ch2', 0.965,
    'sepiolite', 2.14,
    'sio2', 2.648,
    'ss304', 8.0,
    'ss304L', 8.0,
    'ss316', 8.0,
    'ss316L', 8.0,
    'ta', 16.654,
    'ti', 4.54,
    'w', 19.3,
    'v', 6.0,
    'h2o', 0.998027,
    'none', 0)

  mod.density <- as.numeric(matl.density[match(mod, matl.density) + 1])
  ref.density <- as.numeric(matl.density[match(ref, matl.density) + 1])

  # calculate mod mass (g)
  mod.mass <- mod.density * (vol - pu.mass / pu.density)

```

```
  # calculate average density (g/cc)
  avg.density <- (pu.mass + mod.mass) / vol

  # calculate weighted mass fractions
  pu.wt <- pu.mass / (pu.mass + mod.mass)
  mod.wt <- mod.mass / (pu.mass + mod.mass)

  # calculate ref radii (cm)
  ref.rad <- rad + thk
  h2o.rad <- rad + thk + 2.54

  # build title card
  if (ref != 'none' && shape == 'sph') {
    title.card <- paste(mass, form, mod, rad, ref, thk, shape)
  } else if (ref != 'none') {
    title.card <- paste(mass, form, mod, rad, ref, thk, shape, ht)
  } else if (ref == 'none' && shape == 'sph') {
    title.card <- paste(mass, form, mod, rad, ref, shape)
  } else if (ref == 'none') {
    title.card <- paste(mass, form, mod, rad, ref, shape, ht)
  }

  # build cell and surface cards
  if (ref == 'none' || thk == 0) {
    cell.cards <- paste0(
      '1  1 ', -avg.density %>% Format(), ' -1',
      '\n2  2 ', -as.numeric(matl.density[match('h2o', matl.density) + 1]) %>%
      ↪ Format(), ' +1 -2',
      '\n3  0 +2')
    if (shape == 'sph') {
      surface.cards <- paste0(
        '\n1  so  ', rad,
        '\n2  so  ', h2o.rad)
    } else if (shape == 'rcc') {
      surface.cards <- paste0(
        '\n1  rcc  0 0 0 0 0 ', ht, ' ', rad,
        '\n2  rcc  0 0 -2.54 0 0 ', ht + 2.54, ' ', h2o.rad)
    }
  } else {
    cell.cards <- paste0(
      '1  1 ', -avg.density %>% Format(), ' -1',
      '\n2  2 ', -ref.density %>% Format(), ' +1 -2',
      '\n3  3 ', -as.numeric(matl.density[match('h2o', matl.density) + 1]) %>%
      ↪ Format(), ' +2 -3',
      '\n4  0 +3')
    if (shape == 'sph') {
      surface.cards <- paste0(
        '\n1  so  ', rad,
        '\n2  so  ', ref.rad,
        '\n3  so  ', h2o.rad)
    } else if (shape == 'rcc') {
      surface.cards <- paste0(
        '\n1  rcc  0 0 0 0 0 ', ht, ' ', rad,
        '\n2  rcc  0 0 ', -thk, ' 0 0 ', ht + 2 * thk, ' ', ref.rad,
        '\n3  rcc  0 0 ', -thk - 2.54, ' 0 0 ', ht + 2 * thk + 5.08, ' ', h2o.rad)
    }
  }

  # build material cards
  # material 1
  if (form != 'puo2' && mod == 'al2o3') {
    matl.cards <- paste0(
      '\nm1   8016.80c ', (-4.70679444984350e-01 * mod.wt) %>% Format(), ' $ O-16',
      '\n\t  13027.80c ', (-5.29320555015650e-01 * mod.wt) %>% Format(), ' $ Al-27',
      '\n\t  94239.80c ', (-0.95 * pu.wt) %>% Format(), ' $ Pu-239',
      '\n\t  94240.80c ', (-0.05 * pu.wt) %>% Format(), ' $ Pu-240',
      '\nmt1  al27.22t')
  } else if (form != 'puo2' && mod == 'be') {
    matl.cards <- paste0(
      '\nm1   4009.80c ', (-mod.wt) %>% Format(), ' $ Be-9',
      '\n\t  94239.80c ', (-0.95 * pu.wt) %>% Format(), ' $ Pu-239',
```

```
      '\n\t  94240.80c ', (-0.05 * pu.wt) %>% Format(), ' $ Pu-240',
      '\nmt1    be.20t')
  } else if (form != 'puo2' && mod == 'beo') {
    matl.cards <- paste0(
      '\nm1   4009.80c ', (-3.60384984537289e-01 * mod.wt) %>% Format(), ' $ Be-9',
      '\n\t   8016.80c ', (-6.39615015462711e-01 * mod.wt) %>% Format(), ' $ O-16',
      '\n\t  94239.80c ', (-0.95 * pu.wt) %>% Format(), ' $ Pu-239',
      '\n\t  94240.80c ', (-0.05 * pu.wt) %>% Format(), ' $ Pu-240',
      '\nmt1  be-o.20t',
      '\n\t   o-be.20t')
  } else if (form != 'puo2' && mod == 'graphite') {
    matl.cards <- paste0(
      '\nm1   6000.80c ', (-mod.wt) %>% Format(), ' $ C',
      '\n\t  94239.80c ', (-0.95 * pu.wt) %>% Format(), ' $ Pu-239',
      '\n\t  94240.80c ', (-0.05 * pu.wt) %>% Format(), ' $ Pu-240',
      '\nmt1  grph.20t')
  } else if (form != 'puo2' && mod == 'mgo') {
    matl.cards <- paste0(
      '\nm1   8016.80c ', (-3.96896476983704e-01 * mod.wt) %>% Format(), ' $ O-16',
      '\n\t  12024.80c ', (-4.70119114481253e-01 * mod.wt) %>% Format(), ' $ Mg-24',
      '\n\t  12025.80c ', (-6.19996472798894e-02 * mod.wt) %>% Format(), ' $ Mg-25',
      '\n\t  12026.80c ', (-7.09847612551540e-02 * mod.wt) %>% Format(), ' $ Mg-26',
      '\n\t  94239.80c ', (-0.95 * pu.wt) %>% Format(), ' $ Pu-239',
      '\n\t  94240.80c ', (-0.05 * pu.wt) %>% Format(), ' $ Pu-240')
  } else if (form != 'puo2' && mod == 'ch2') {
    matl.cards <- paste0(
      '\nm1   1001.80c ', (-1.43701457933504e-01 * mod.wt) %>% Format(), ' $ H-1',
      '\n\t   6000.80c ', (-8.56298542066496e-01 * mod.wt) %>% Format(), ' $ C',
      '\n\t  94239.80c ', (-0.95 * pu.wt) %>% Format(), ' $ Pu-239',
      '\n\t  94240.80c ', (-0.05 * pu.wt) %>% Format(), ' $ Pu-240',
      '\nmt1  poly.20t')
  } else if (form != 'puo2' && mod == 'sepiolite') {
    matl.cards <- paste0(
      '\nm1   1001.80c ', (-2.17829111286340e-02 * mod.wt) %>% Format(), ' $ H-1',
      '\n\t   8016.80c ', (-5.67953140203876e-01 * mod.wt) %>% Format(), ' $ O-16',
      '\n\t  12024.80c ', (-1.16997161651148e-01 * mod.wt) %>% Format(), ' $ Mg-24',
      '\n\t  12025.80c ', (-1.54296699106214e-02 * mod.wt) %>% Format(), ' $ Mg-25',
      '\n\t  12026.80c ', (-1.76657688052133e-02 * mod.wt) %>% Format(), ' $ Mg-26',
      '\n\t  14028.80c ', (-2.39030664873055e-01 * mod.wt) %>% Format(), ' $ Si-28',
      '\n\t  14029.80c ', (-1.25696735473918e-02 * mod.wt) %>% Format(), ' $ Si-29',
      '\n\t  14030.80c ', (-8.57100988006093e-03 * mod.wt) %>% Format(), ' $ Si-30',
      '\n\t  94239.80c ', (-0.95 * pu.wt) %>% Format(), ' $ Pu-239',
      '\n\t  94240.80c ', (-0.05 * pu.wt) %>% Format(), ' $ Pu-240')
  } else if (form != 'puo2' && mod == 'sio2') {
    matl.cards <- paste0(
      '\nm1   8016.80c ', (-5.32481915390405e-01 * mod.wt) %>% Format(), ' $ O-16',
      '\n\t  14028.80c ', (-4.29529075105271e-01 * mod.wt) %>% Format(), ' $ Si-28',
      '\n\t  14029.80c ', (-2.25872285300873e-02 * mod.wt) %>% Format(), ' $ Si-29',
      '\n\t  14030.80c ', (-1.54017809742357e-02 * mod.wt) %>% Format(), ' $ Si-30',
      '\n\t  94239.80c ', (-0.95 * pu.wt) %>% Format(), ' $ Pu-239',
      '\n\t  94240.80c ', (-0.05 * pu.wt) %>% Format(), ' $ Pu-240',
      '\nmt1  sio2.30t')
  } else if (form != 'puo2' && mod == 'h2o') {
    matl.cards <- paste0(
      '\nm1   1001.80c ', (-1.11914873272364e-01 * mod.wt) %>% Format(), ' $ H-1',
      '\n\t   8016.80c ', (-8.88085126727636e-01 * mod.wt) %>% Format(), ' $ O-16',
      '\n\t  94239.80c ', (-0.95 * pu.wt) %>% Format(), ' $ Pu-239',
      '\n\t  94240.80c ', (-0.05 * pu.wt) %>% Format(), ' $ Pu-240',
      '\nmt1  lwtr.20t')
  } else if (form != 'puo2' && mod == 'none') {
    matl.cards <- paste0(
      '\nm1  94239.80c ', -0.95, ' $ Pu-239',
      '\n\t  94240.80c ', -0.05, ' $ Pu-240')
  } else if (form == 'puo2' && mod == 'al2o3') {
    matl.cards <- paste0(
      '\nm1   8016.80c ', (-4.70679444984350e-01 * mod.wt - 1.18003643801257e-01 *
        ↪ pu.wt) %>% Format(), ' $ O-16',
      '\n\t  13027.80c ', (-5.29320555015650e-01 * mod.wt) %>% Format(), ' $ Al-27',
      '\n\t  94239.80c ', (-8.37896538388806e-01 * pu.wt) %>% Format(), ' $ Pu-239',
      '\n\t  94240.80c ', (-4.40998178099372e-02 * pu.wt) %>% Format(), ' $ Pu-240',
      '\nmt1  al27.22t')
```

```
278   } else if (form == 'puo2' && mod == 'be') {
279     matl.cards <- paste0(
280       '\nm1    4009.80c ', (-mod.wt) %>% Format(), ' $ Be-9',
281       '\n\t    8016.80c ', (-1.18003643801257e-01 * pu.wt) %>% Format(), ' $ O-16',
282       '\n\t   94239.80c ', (-8.37896538388806e-01 * pu.wt) %>% Format(), ' $ Pu-239',
283       '\n\t   94240.80c ', (-4.40998178099372e-02 * pu.wt) %>% Format(), ' $ Pu-240',
284       '\nmt1     be.20t')
285   } else if (form == 'puo2' && mod == 'beo') {
286     matl.cards <- paste0(
287       '\nm1    4009.80c ', (-3.60384984537289e-01 * mod.wt) %>% Format(), ' $ Be-9',
288       '\n\t    8016.80c ', (-6.39615015462711e-01 * mod.wt - 1.18003643801257e-01 *
        ↪ pu.wt) %>% Format(), ' $ O-16',
289       '\n\t   94239.80c ', (-8.37896538388806e-01 * pu.wt) %>% Format(), ' $ Pu-239',
290       '\n\t   94240.80c ', (-4.40998178099372e-02 * pu.wt) %>% Format(), ' $ Pu-240',
291       '\nmt1   be-o.20t',
292       '\n\t    o-be.20t')
293   } else if (form == 'puo2' && mod == 'graphite') {
294     matl.cards <- paste0(
295       '\nm1    6000.80c ', (-mod.wt) %>% Format(), ' $ C',
296       '\n\t    8016.80c ', (-1.18003643801257e-01 * pu.wt) %>% Format(), ' $ O-16',
297       '\n\t   94239.80c ', (-8.37896538388806e-01 * pu.wt) %>% Format(), ' $ Pu-239',
298       '\n\t   94240.80c ', (-4.40998178099372e-02 * pu.wt) %>% Format(), ' $ Pu-240',
299       '\nmt1   grph.20t')
300   } else if (form == 'puo2' && mod == 'mgo') {
301     matl.cards <- paste0(
302       '\nm1    8016.80c ', (-3.96896476983704e-01 * mod.wt - 1.18003643801257e-01 *
        ↪ pu.wt) %>% Format(), ' $ O-16',
303       '\n\t   12024.80c ', (-4.70119114481253e-01 * mod.wt) %>% Format(), ' $ Mg-24',
304       '\n\t   12025.80c ', (-6.19996472798894e-02 * mod.wt) %>% Format(), ' $ Mg-25',
305       '\n\t   12026.80c ', (-7.09847612551540e-02 * mod.wt) %>% Format(), ' $ Mg-26',
306       '\n\t   94239.80c ', (-8.37896538388806e-01 * pu.wt) %>% Format(), ' $ Pu-239',
307       '\n\t   94240.80c ', (-4.40998178099372e-02 * pu.wt) %>% Format(), ' $ Pu-240')
308   } else if (form == 'puo2' && mod == 'ch2') {
309     matl.cards <- paste0(
310       '\nm1    1001.80c ', (-1.43701457933504e-01 * mod.wt) %>% Format(), ' $ H-1',
311       '\n\t    6000.80c ', (-8.56298542066496e-01 * mod.wt) %>% Format(), ' $ C',
312       '\n\t    8016.80c ', (-1.18003643801257e-01 * pu.wt) %>% Format(), ' $ O-16',
313       '\n\t   94239.80c ', (-8.37896538388806e-01 * pu.wt) %>% Format(), ' $ Pu-239',
314       '\n\t   94240.80c ', (-4.40998178099372e-02 * pu.wt) %>% Format(), ' $ Pu-240',
315       '\nmt1   poly.20t')
316   } else if (form == 'puo2' && mod == 'sepiolite') {
317     matl.cards <- paste0(
318       '\nm1    1001.80c ', (-2.17829111286340e-02 * mod.wt) %>% Format(), ' $ H-1',
319       '\n\t    8016.80c ', (-5.67953140203876e-01 * mod.wt - 1.18003643801257e-01 *
        ↪ pu.wt) %>% Format(), ' $ O-16',
320       '\n\t   12024.80c ', (-1.16997161651148e-01 * mod.wt) %>% Format(), ' $ Mg-24',
321       '\n\t   12025.80c ', (-1.54296699106214e-02 * mod.wt) %>% Format(), ' $ Mg-25',
322       '\n\t   12026.80c ', (-1.76657688052133e-02 * mod.wt) %>% Format(), ' $ Mg-26',
323       '\n\t   14028.80c ', (-2.39030664873055e-01 * mod.wt) %>% Format(), ' $ Si-28',
324       '\n\t   14029.80c ', (-1.25696735473918e-02 * mod.wt) %>% Format(), ' $ Si-29',
325       '\n\t   14030.80c ', (-8.57100988006093e-03 * mod.wt) %>% Format(), ' $ Si-30',
326       '\n\t   94239.80c ', (-8.37896538388806e-01 * pu.wt) %>% Format(), ' $ Pu-239',
327       '\n\t   94240.80c ', (-4.40998178099372e-02 * pu.wt) %>% Format(), ' $ Pu-240')
328   } else if (form == 'puo2' && mod == 'sio2') {
329     matl.cards <- paste0(
330       '\nm1    8016.80c ', (-5.32481915390405e-01 * mod.wt - 1.18003643801257e-01 *
        ↪ pu.wt) %>% Format(), ' $ O-16',
331       '\n\t   14028.80c ', (-4.29529075105271e-01 * mod.wt) %>% Format(), ' $ Si-28',
332       '\n\t   14029.80c ', (-2.25872285300873e-02 * mod.wt) %>% Format(), ' $ Si-29',
333       '\n\t   14030.80c ', (-1.54017809742357e-02 * mod.wt) %>% Format(), ' $ Si-30',
334       '\n\t   94239.80c ', (-8.37896538388806e-01 * pu.wt) %>% Format(), ' $ Pu-239',
335       '\n\t   94240.80c ', (-4.40998178099372e-02 * pu.wt) %>% Format(), ' $ Pu-240',
336       '\nmt1   sio2.30t')
337   } else if (form == 'puo2' && mod == 'h2o') {
338     matl.cards <- paste0(
339       '\nm1    1001.80c ', (-1.11914873272364e-01 * mod.wt) %>% Format(), ' $ H-1',
340       '\n\t    8016.80c ', (-8.88085126727636e-01 * mod.wt - 1.18003643801257e-01 *
        ↪ pu.wt) %>% Format(), ' $ O-16',
341       '\n\t   94239.80c ', (-8.37896538388806e-01 * pu.wt) %>% Format(), ' $ Pu-239',
342       '\n\t   94240.80c ', (-4.40998178099372e-02 * pu.wt) %>% Format(), ' $ Pu-240',
343       '\nmt1   lwtr.20t')
```

```
  } else if (form == 'puo2' && mod == 'none') {
    matl.cards <- paste0(
      '\nm1    8016.80c ', -1.18003643801257e-01 %>% Format(), ' $ O-16',
      '\n\t   94239.80c ', -8.37896538388806e-01 %>% Format(), ' $ Pu-239',
      '\n\t   94240.80c ', -4.40998178099372e-02 %>% Format(), ' $ Pu-240')
  }
  # materials 2 and 3
  if (ref == 'al') {
    matl.cards <- paste0(
      matl.cards,
      '\nm2   13027.80c +1 $ Al-27',
      '\nmt2  al27.22t',
      '\nm3    1001.80c +2 $ H-1',
      '\n\t    8016.80c +1 $ O-16',
      '\nmt3  lwtr.20t')
  } else if (ref == 'al2o3') {
    matl.cards <- paste0(
      matl.cards,
      '\nm2    8016.80c +3 $ O-16',
      '\n\t   13027.80c +2 $ Al-27',
      '\nmt2  al27.22t',
      '\nm3    1001.80c +2 $ H-1',
      '\n\t    8016.80c +1 $ O-16',
      '\nmt3  lwtr.20t')
  } else if (ref == 'be') {
    matl.cards <- paste0(
      matl.cards,
      '\nm2    4009.80c +1 $ Be-9',
      '\nmt2     be.20t',
      '\nm3    1001.80c +2 $ H-1',
      '\n\t    8016.80c +1 $ O-16',
      '\nmt3  lwtr.20t')
  } else if (ref == 'beo') {
    matl.cards <- paste0(
      matl.cards,
      '\nm2    4009.80c +1 $ Be-9',
      '\n\t    8016.80c +1 $ O-16',
      '\nmt2  be-o.20t',
      '\n\t    o-be.20t',
      '\nm3    1001.80c +2 $ H-1',
      '\n\t    8016.80c +1 $ O-16',
      '\nmt3  lwtr.20t')
  } else if (ref == 'cs') {
    matl.cards <- paste0(
      matl.cards,
      '\nm2    6000.80c ', -0.005000 %>% Format(), ' $ C',
      '\n\t   26054.80c ', -5.61733047504445e-02 %>% Format(), ' $ Fe-54',
      '\n\t   26056.80c ', -9.14420210917578e-01 %>% Format(), ' $ Fe-56',
      '\n\t   26057.80c ', -2.14956677132355e-02 %>% Format(), ' $ Fe-57',
      '\n\t   26058.80c ', -2.91081661874234e-03 %>% Format(), ' $ Fe-58',
      '\nmt2  fe56.22t',
      '\nm3    1001.80c +2 $ H-1',
      '\n\t    8016.80c +1 $ O-16',
      '\nmt3  lwtr.20t')
  } else if (ref == 'cu') {
    matl.cards <- paste0(
      matl.cards,
      '\nm2   29063.80c ', -6.84994320997273e-01 %>% Format(), ' $ Cu-63',
      '\n\t   29065.80c ', -3.15005679002727e-01 %>% Format(), ' $ Cu-65',
      '\nm3    1001.80c +2 $ H-1',
      '\n\t    8016.80c +1 $ O-16',
      '\nmt3  lwtr.20t')
  } else if (ref == 'du') {
    matl.cards <- paste0(
      matl.cards,
      '\nm2   92234.80c ', -0.000005 %>% Format(), ' $ U-234',
      '\n\t   92235.80c ', -0.002500 %>% Format(), ' $ U-235',
      '\n\t   92238.80c ', -0.997495 %>% Format(), ' $ U-238',
      '\nm3    1001.80c +2 $ H-1',
      '\n\t    8016.80c +1 $ O-16',
      '\nmt3  lwtr.20t')
```

```
415   } else if (ref == 'granite') {
416     matl.cards <- paste0(
417       matl.cards,
418       '\nm2    8016.80c ', -0.484170 %>% Format(), ' $ O-16',
419       '\n\t   11023.80c ', -0.027328 %>% Format(), ' $ Na-23',
420       '\n\t   12024.80c ', -3.33158242094796e-03 %>% Format(), ' $ Mg-24',
421       '\n\t   12025.80c ', -4.39371488246284e-04 %>% Format(), ' $ Mg-25',
422       '\n\t   12026.80c ', -5.03046090805758e-04 %>% Format(), ' $ Mg-26',
423       '\n\t   13027.80c ', -0.076188 %>% Format(), ' $ Al-27',
424       '\n\t   14028.80c ', -3.08852992862601e-01 %>% Format(), ' $ Si-28',
425       '\n\t   14029.80c ', -1.62413525330718e-02 %>% Format(), ' $ Si-29',
426       '\n\t   14030.80c ', -1.10746546043271e-02 %>% Format(), ' $ Si-30',
427       '\n\t   19039.80c ', -3.17324306083595e-02 %>% Format(), ' $ K-39',
428       '\n\t   19040.80c ', -4.08330016373087e-06 %>% Format(), ' $ K-40',
429       '\n\t   19041.80c ', -2.40748609147673e-03 %>% Format(), ' $ K-41',
430       '\n\t   20040.80c ', -1.25509788921567e-02 %>% Format(), ' $ Ca-40',
431       '\n\t   20042.80c ', -8.79521381703543e-05 %>% Format(), ' $ Ca-42',
432       '\n\t   20043.80c ', -1.87891233810042e-05 %>% Format(), ' $ Ca-43',
433       '\n\t   20044.80c ', -2.97632447849203e-04 %>% Format(), ' $ Ca-44',
434       '\n\t   20046.80c ', -5.95526856619387e-07 %>% Format(), ' $ Ca-46',
435       '\n\t   20048.80c ', -2.90518715861230e-05 %>% Format(), ' $ Ca-48',
436       '\n\t   22046.80c ', -1.42165708671302e-04 %>% Format(), ' $ Ti-46',
437       '\n\t   22047.80c ', -1.30995193080019e-04 %>% Format(), ' $ Ti-47',
438       '\n\t   22048.80c ', -1.32551868873429e-03 %>% Format(), ' $ Ti-48',
439       '\n\t   22049.80c ', -9.93028199522752e-05 %>% Format(), ' $ Ti-49',
440       '\n\t   22050.80c ', -9.70175895621157e-05 %>% Format(), ' $ Ti-50',
441       '\n\t   25055.80c ', -0.000387 %>% Format(), ' $ Mn-55',
442       '\n\t   26054.80c ', -1.21690008431742e-03 %>% Format(), ' $ Fe-54',
443       '\n\t   26056.80c ', -1.98093745189230e-02 %>% Format(), ' $ Fe-56',
444       '\n\t   26057.80c ', -4.65667454832955e-04 %>% Format(), ' $ Fe-57',
445       '\n\t   26058.80c ', -6.30579419266243e-05 %>% Format(), ' $ Fe-58',
446       '\n\t   82204.80c ', -1.38359612318662e-05 %>% Format(), ' $ Pb-204',
447       '\n\t   82206.80c ', -2.40513219139613e-04 %>% Format(), ' $ Pb-206',
448       '\n\t   82207.80c ', -2.21625930181222e-04 %>% Format(), ' $ Pb-207',
449       '\n\t   82208.80c ', -5.28024889447298e-04 %>% Format(), ' $ Pb-208',
450       '\nmt2   al27.22t',
451       '\n\t    sio2.30t',
452       '\n\t    fe56.22t',
453       '\nm3    1001.80c +2 $ H-1',
454       '\n\t    8016.80c +1 $ O-16',
455       '\nmt3   lwtr.20t')
456   } else if (ref == 'graphite') {
457     matl.cards <- paste0(
458       matl.cards,
459       '\nm2    6000.80c +1 $ C',
460       '\nmt2   grph.20t',
461       '\nm3    1001.80c +2 $ H-1',
462       '\n\t    8016.80c +1 $ O-16',
463       '\nmt3   lwtr.20t')
464   } else if (ref == 'fe') {
465     matl.cards <- paste0(
466       matl.cards,
467       '\nm2   26054.80c ', -5.64555826637633e-02 %>% Format(), ' $ Fe-54',
468       '\n\t   26056.80c ', -9.19015287354349e-01 %>% Format(), ' $ Fe-56',
469       '\n\t   26057.80c ', -2.16036861439553e-02 %>% Format(), ' $ Fe-57',
470       '\n\t   26058.80c ', -2.92544383793200e-03 %>% Format(), ' $ Fe-58',
471       '\nm3    1001.80c +2 $ H-1',
472       '\n\t    8016.80c +1 $ O-16',
473       '\nmt3   lwtr.20t')
474   } else if (ref == 'pb') {
475     matl.cards <- paste0(
476       matl.cards,
477       '\nm2   82204.80c ', -1.37808378803449e-02 %>% Format(), ' $ Pb-204',
478       '\n\t   82206.80c ', -2.39554999143041e-01 %>% Format(), ' $ Pb-206',
479       '\n\t   82207.80c ', -2.20742958347831e-01 %>% Format(), ' $ Pb-207',
480       '\n\t   82208.80c ', -5.25921204628783e-01 %>% Format(), ' $ Pb-208',
481       '\nm3    1001.80c +2 $ H-1',
482       '\n\t    8016.80c +1 $ O-16',
483       '\nmt3   lwtr.20t')
484   } else if (ref == 'mgo') {
485     matl.cards <- paste0(
```

```
      matl.cards,
      '\nm2   8016.80c ', -3.96896476983704e-01 %>% Format(), ' $ O-16',
      '\n\t  12024.80c ', -4.70119114481253e-01 %>% Format(), ' $ Mg-24',
      '\n\t  12025.80c ', -6.19996472798894e-02 %>% Format(), ' $ Mg-25',
      '\n\t  12026.80c ', -7.09847612551540e-02 %>% Format(), ' $ Mg-26',
      '\nm3   1001.80c +2 $ H-1',
      '\n\t   8016.80c +1 $ O-16',
      '\nmt3  lwtr.20t')
  } else if (ref == 'mo') {
    matl.cards <- paste0(
      matl.cards,
      '\nm2  42092.80c ', -1.42174367179782e-01 %>% Format(), ' $ Mo-92',
      '\n\t  42094.80c ', -9.05462698581737e-02 %>% Format(), ' $ Mo-94',
      '\n\t  42095.80c ', -1.57498244126087e-01 %>% Format(), ' $ Mo-95',
      '\n\t  42096.80c ', -1.66753727400932e-01 %>% Format(), ' $ Mo-96',
      '\n\t  42097.80c ', -9.64703471057269e-02 %>% Format(), ' $ Mo-97',
      '\n\t  42098.80c ', -2.46265576821488e-01 %>% Format(), ' $ Mo-98',
      '\n\t  42100.80c ', -1.00291467507811e-01 %>% Format(), ' $ Mo-100',
      '\nm3   1001.80c +2 $ H-1',
      '\n\t   8016.80c +1 $ O-16',
      '\nmt3  lwtr.20t')
  } else if (ref == 'ni') {
    matl.cards <- paste0(
      matl.cards,
      '\nm2  28058.80c ', -6.71977983510585e-01 %>% Format(), ' $ Ni-58',
      '\n\t  28060.80c ', -2.67758585123918e-01 %>% Format(), ' $ Ni-60',
      '\n\t  28061.80c ', -1.18346279615680e-02 %>% Format(), ' $ Ni-61',
      '\n\t  28062.80c ', -3.83429436164925e-02 %>% Format(), ' $ Ni-62',
      '\n\t  28064.80c ', -1.00858597874365e-02 %>% Format(), ' $ Ni-64',
      '\nm3   1001.80c +2 $ H-1',
      '\n\t   8016.80c +1 $ O-16',
      '\nmt3  lwtr.20t')
  } else if (ref == 'nb') {
    matl.cards <- paste0(
      matl.cards,
      '\nm2  41093.80c +1 $ Nb-93',
      '\nm3   1001.80c +2 $ H-1',
      '\n\t   8016.80c +1 $ O-16',
      '\nmt3  lwtr.20t')
  } else if (ref == 'pt') {
    matl.cards <- paste0(
      matl.cards,
      '\nm2  78000.40c +1 $ Pt',
      '\nm3   1001.80c +2 $ H-1',
      '\n\t   8016.80c +1 $ O-16',
      '\nmt3  lwtr.20t')
  } else if (ref == 'ch2') {
    matl.cards <- paste0(
      matl.cards,
      '\nm2   1001.80c +2 $ H-1',
      '\n\t   6000.80c +1 $ C',
      '\nmt2  poly.20t',
      '\nm3   1001.80c +2 $ H-1',
      '\n\t   8016.80c +1 $ O-16',
      '\nmt3  lwtr.20t')
  } else if (ref == 'ss304') {
    matl.cards <- paste0(
      matl.cards,
      '\nm2   6000.80c ', -0.000400 %>% Format(), ' $ C',
      '\n\t  14028.80c ', -4.59371614965391e-03 %>% Format(), ' $ Si-28',
      '\n\t  14029.80c ', -2.41565292056552e-04 %>% Format(), ' $ Si-29',
      '\n\t  14030.80c ', -1.64718558289537e-04 %>% Format(), ' $ Si-30',
      '\n\t  15031.80c ', -0.000230 %>% Format(), ' $ P-31',
      '\n\t  16032.80c ', -1.42151263009614e-04 %>% Format(), ' $ S-32',
      '\n\t  16033.80c ', -1.15708257503575e-06 %>% Format(), ' $ S-33',
      '\n\t  16034.80c ', -6.69137391313452e-06 %>% Format(), ' $ S-34',
      '\n\t  16036.80c ', -2.80502215984187e-10 %>% Format(), ' $ S-36',
      '\n\t  24050.80c ', -7.93000447879800e-03 %>% Format(), ' $ Cr-50',
      '\n\t  24052.80c ', -1.59028788463595e-01 %>% Format(), ' $ Cr-52',
      '\n\t  24053.80c ', -1.83798150490731e-02 %>% Format(), ' $ Cr-53',
      '\n\t  24054.80c ', -4.66139200853358e-03 %>% Format(), ' $ Cr-54',
```

```
      '\n\t   25055.80c ', -0.010000 %>% Format(), ' $ Mn-55',
      '\n\t   26054.80c ', -3.96165760226426e-02 %>% Format(), ' $ Fe-54',
      '\n\t   26056.80c ', -6.44900597595167e-01 %>% Format(), ' $ Fe-56',
      '\n\t   26057.80c ', -1.51599546777977e-02 %>% Format(), ' $ Fe-57',
      '\n\t   26058.80c ', -2.05287170439202e-03 %>% Format(), ' $ Fe-58',
      '\n\t   28058.80c ', -6.21579634747292e-02 %>% Format(), ' $ Ni-58',
      '\n\t   28060.80c ', -2.47676691239624e-02 %>% Format(), ' $ Ni-60',
      '\n\t   28061.80c ', -1.09470308644504e-03 %>% Format(), ' $ Ni-61',
      '\n\t   28062.80c ', -3.54672228452556e-03 %>% Format(), ' $ Ni-62',
      '\n\t   28064.80c ', -9.32942030337874e-04 %>% Format(), ' $ Ni-64',
      '\nmt2  fe56.22t',
      '\nm3    1001.80c +2 $ H-1',
      '\n\t    8016.80c +1 $ O-16',
      '\nmt3  lwtr.20t')
  } else if (ref == 'ss304L') {
    matl.cards <- paste0(
      matl.cards,
      '\nm2    6000.80c ', -0.000150 %>% Format(), ' $ C',
      '\n\t   14028.80c ', -4.59371614965391e-03 %>% Format(), ' $ Si-28',
      '\n\t   14029.80c ', -2.41565292056552e-04 %>% Format(), ' $ Si-29',
      '\n\t   14030.80c ', -1.64718558289537e-04 %>% Format(), ' $ Si-30',
      '\n\t   15031.80c ', -0.000230 %>% Format(), ' $ P-31',
      '\n\t   16032.80c ', -1.42151263009614e-04 %>% Format(), ' $ S-32',
      '\n\t   16033.80c ', -1.15708257503575e-06 %>% Format(), ' $ S-33',
      '\n\t   16034.80c ', -6.69137391313452e-06 %>% Format(), ' $ S-34',
      '\n\t   16036.80c ', -2.80502215984187e-10 %>% Format(), ' $ S-36',
      '\n\t   24050.80c ', -7.93000447879800e-03 %>% Format(), ' $ Cr-50',
      '\n\t   24052.80c ', -1.59028788463595e-01 %>% Format(), ' $ Cr-52',
      '\n\t   24053.80c ', -1.83798150490731e-02 %>% Format(), ' $ Cr-53',
      '\n\t   24054.80c ', -4.66139200853358e-03 %>% Format(), ' $ Cr-54',
      '\n\t   25055.80c ', -0.010000 %>% Format(), ' $ Mn-55',
      '\n\t   26054.80c ', -3.92072730483303e-02 %>% Format(), ' $ Fe-54',
      '\n\t   26056.80c ', -6.38237736761848e-01 %>% Format(), ' $ Fe-56',
      '\n\t   26057.80c ', -1.50033279532540e-02 %>% Format(), ' $ Fe-57',
      '\n\t   26058.80c ', -2.03166223656702e-03 %>% Format(), ' $ Fe-58',
      '\n\t   28058.80c ', -6.71977983510585e-02 %>% Format(), ' $ Ni-58',
      '\n\t   28060.80c ', -2.67758585123918e-02 %>% Format(), ' $ Ni-60',
      '\n\t   28061.80c ', -1.18346279615680e-03 %>% Format(), ' $ Ni-61',
      '\n\t   28062.80c ', -3.83429436164925e-03 %>% Format(), ' $ Ni-62',
      '\n\t   28064.80c ', -1.00858597874365e-03 %>% Format(), ' $ Ni-64',
      '\nmt2  fe56.22t',
      '\nm3    1001.80c +2 $ H-1',
      '\n\t    8016.80c +1 $ O-16',
      '\nmt3  lwtr.20t')
  } else if (ref == 'ss316') {
    matl.cards <- paste0(
      matl.cards,
      '\nm2    6000.80c ', -0.000410 %>% Format(), ' $ C',
      '\n\t   14028.80c ', -4.65802817574907e-03 %>% Format(), ' $ Si-28',
      '\n\t   14029.80c ', -2.44947206145344e-04 %>% Format(), ' $ Si-29',
      '\n\t   14030.80c ', -1.67024618105591e-04 %>% Format(), ' $ Si-30',
      '\n\t   15031.80c ', -0.000230 %>% Format(), ' $ P-31',
      '\n\t   16032.80c ', -1.42151263009614e-04 %>% Format(), ' $ S-32',
      '\n\t   16033.80c ', -1.15708257503575e-06 %>% Format(), ' $ S-33',
      '\n\t   16034.80c ', -6.69137391313452e-06 %>% Format(), ' $ S-34',
      '\n\t   16036.80c ', -2.80502215984187e-10 %>% Format(), ' $ S-36',
      '\n\t   24050.80c ', -7.09526716524032e-03 %>% Format(), ' $ Cr-50',
      '\n\t   24052.80c ', -1.42288915993743e-01 %>% Format(), ' $ Cr-52',
      '\n\t   24053.80c ', -1.64450976754864e-02 %>% Format(), ' $ Cr-53',
      '\n\t   24054.80c ', -4.17071916553004e-03 %>% Format(), ' $ Cr-54',
      '\n\t   25055.80c ', -0.010140 %>% Format(), ' $ Mn-55',
      '\n\t   26054.80c ', -3.77687848020577e-02 %>% Format(), ' $ Fe-54',
      '\n\t   26056.80c ', -6.14821227240060e-01 %>% Format(), ' $ Fe-56',
      '\n\t   26057.80c ', -1.44528660303061e-02 %>% Format(), ' $ Fe-57',
      '\n\t   26058.80c ', -1.95712192757651e-03 %>% Format(), ' $ Fe-58',
      '\n\t   28058.80c ', -8.06373580212703e-02 %>% Format(), ' $ Ni-58',
      '\n\t   28060.80c ', -3.21310302148701e-02 %>% Format(), ' $ Ni-60',
      '\n\t   28061.80c ', -1.42015535538816e-03 %>% Format(), ' $ Ni-61',
      '\n\t   28062.80c ', -4.60115323397910e-03 %>% Format(), ' $ Ni-62',
      '\n\t   28064.80c ', -1.21030317449238e-03 %>% Format(), ' $ Ni-64',
      '\n\t   42092.80c ', -3.55435917949455e-03 %>% Format(), ' $ Mo-92',
```

```
      '\n\t  42094.80c ', -2.26365674645434e-03 %>% Format(), ' $ Mo-94',
      '\n\t  42095.80c ', -3.93745610315217e-03 %>% Format(), ' $ Mo-95',
      '\n\t  42096.80c ', -4.16884318502330e-03 %>% Format(), ' $ Mo-96',
      '\n\t  42097.80c ', -2.41175867764317e-03 %>% Format(), ' $ Mo-97',
      '\n\t  42098.80c ', -6.15663942053720e-03 %>% Format(), ' $ Mo-98',
      '\n\t  42100.80c ', -2.50728668769527e-03 %>% Format(), ' $ Mo-100',
      '\nmt2  fe56.22t',
      '\nm3   1001.80c +2 $ H-1',
      '\n\t   8016.80c +1 $ O-16',
      '\nmt3  lwtr.20t')
  } else if (ref == 'ss316L') {
    matl.cards <- paste0(
      matl.cards,
      '\nm2   6000.80c ', -0.000300 %>% Format(), ' $ C',
      '\n\t  14028.80c ', -9.18743229930782e-03 %>% Format(), ' $ Si-28',
      '\n\t  14029.80c ', -4.83130584113104e-04 %>% Format(), ' $ Si-29',
      '\n\t  14030.80c ', -3.29437116579075e-04 %>% Format(), ' $ Si-30',
      '\n\t  15031.80c ', -0.000450 %>% Format(), ' $ P-31',
      '\n\t  16032.80c ', -2.84302526019227e-04 %>% Format(), ' $ S-32',
      '\n\t  16033.80c ', -2.31416515007149e-06 %>% Format(), ' $ S-33',
      '\n\t  16034.80c ', -1.33827478262690e-05 %>% Format(), ' $ S-34',
      '\n\t  16036.80c ', -5.61004431968375e-10 %>% Format(), ' $ S-36',
      '\n\t  24050.80c ', -7.09526716524032e-03 %>% Format(), ' $ Cr-50',
      '\n\t  24052.80c ', -1.42288915993743e-01 %>% Format(), ' $ Cr-52',
      '\n\t  24053.80c ', -1.64450976754864e-02 %>% Format(), ' $ Cr-53',
      '\n\t  24054.80c ', -4.17071916553004e-03 %>% Format(), ' $ Cr-54',
      '\n\t  25055.80c ', -0.020000 %>% Format(), ' $ Mn-55',
      '\n\t  26054.80c ', -3.69191282829680e-02 %>% Format(), ' $ Fe-54',
      '\n\t  26056.80c ', -6.00990047165377e-01 %>% Format(), ' $ Fe-56',
      '\n\t  26057.80c ', -1.41277305538395e-02 %>% Format(), ' $ Fe-57',
      '\n\t  26058.80c ', -1.91309399781563e-03 %>% Format(), ' $ Fe-58',
      '\n\t  28058.80c ', -8.06373580212703e-02 %>% Format(), ' $ Ni-58',
      '\n\t  28060.80c ', -3.21310302148701e-02 %>% Format(), ' $ Ni-60',
      '\n\t  28061.80c ', -1.42015535538816e-03 %>% Format(), ' $ Ni-61',
      '\n\t  28062.80c ', -4.60115323397910e-03 %>% Format(), ' $ Ni-62',
      '\n\t  28064.80c ', -1.21030317449238e-03 %>% Format(), ' $ Ni-64',
      '\n\t  42092.80c ', -3.55435917949455e-03 %>% Format(), ' $ Mo-92',
      '\n\t  42094.80c ', -2.26365674645434e-03 %>% Format(), ' $ Mo-94',
      '\n\t  42095.80c ', -3.93745610315217e-03 %>% Format(), ' $ Mo-95',
      '\n\t  42096.80c ', -4.16884318502330e-03 %>% Format(), ' $ Mo-96',
      '\n\t  42097.80c ', -2.41175867764317e-03 %>% Format(), ' $ Mo-97',
      '\n\t  42098.80c ', -6.15663942053720e-03 %>% Format(), ' $ Mo-98',
      '\n\t  42100.80c ', -2.50728668769527e-03 %>% Format(), ' $ Mo-100',
      '\nmt2  fe56.22t',
      '\nm3   1001.80c +2 $ H-1',
      '\n\t   8016.80c +1 $ O-16',
      '\nmt3  lwtr.20t')
  } else if (ref == 'ta') {
    matl.cards <- paste0(
      matl.cards,
      '\nm2  73181.80c +1 $ Ta-181',
      '\nm3   1001.80c +2 $ H-1',
      '\n\t   8016.80c +1 $ O-16',
      '\nmt3  lwtr.20t')
  } else if (ref == 'ti') {
    matl.cards <- paste0(
      matl.cards,
      '\nm2  22046.80c ', -7.92009519060180e-02 %>% Format(), ' $ Ti-46',
      '\n\t  22047.80c ', -7.29778234429076e-02 %>% Format(), ' $ Ti-47',
      '\n\t  22048.80c ', -7.38450522971748e-01 %>% Format(), ' $ Ti-48',
      '\n\t  22049.80c ', -5.53219052658915e-02 %>% Format(), ' $ Ti-49',
      '\n\t  22050.80c ', -5.40487964134349e-02 %>% Format(), ' $ Ti-50',
      '\nm3   1001.80c +2 $ H-1',
      '\n\t   8016.80c +1 $ O-16',
      '\nmt3  lwtr.20t')
  } else if (ref == 'w') {
    matl.cards <- paste0(
      matl.cards,
      '\nm2  74180.80c ', -1.17457548331097e-03 %>% Format(), ' $ W-180',
      '\n\t  74182.80c ', -2.62270494784839e-01 %>% Format(), ' $ W-182',
      '\n\t  74183.80c ', -1.42406025314416e-01 %>% Format(), ' $ W-183',
```

```
      '\n\t  74184.80c ', -3.06581920073334e-01 %>% Format(), ' $ W-184',
      '\n\t  74186.80c ', -2.87566984344099e-01 %>% Format(), ' $ W-186',
      '\nm3   1001.80c +2 $ H-1',
      '\n\t   8016.80c +1 $ O-16',
      '\nmt3  lwtr.20t')
  } else if (ref == 'v') {
    matl.cards <- paste0(
      matl.cards,
      '\nm2  23050.80c ', -2.45120335808159e-03 %>% Format(), ' $ V-50',
      '\n\t  23051.80c ', -9.97548796641918e-01 %>% Format(), ' $ V-51',
      '\nm3   1001.80c +2 $ H-1',
      '\n\t   8016.80c +1 $ O-16',
      '\nmt3  lwtr.20t')
  } else if (ref == 'h2o') {
    matl.cards <- paste0(
      matl.cards,
      '\nm2   1001.80c +2 $ H-1',
      '\n\t   8016.80c +1 $ O-16',
      '\nmt2  lwtr.20t',
      '\nm3   1001.80c +2 $ H-1',
      '\n\t   8016.80c +1 $ O-16',
      '\nmt3  lwtr.20t')
  } else if (ref == 'none') {
    matl.cards <- paste0(
      matl.cards,
      '\nm2   1001.80c +2 $ H-1',
      '\n\t   8016.80c +1 $ O-16',
      '\nmt2  lwtr.20t')
  }

  # build data cards
  if (ref == 'none') {
    imp <- 'imp:n 1 1 0'
  } else {
    imp <- 'imp:n 1 1 1 0'
  }

  kcode <- 'kcode 10000 1 50 500'

  source <- (2/3 * rad) %>% round(2)

  if (shape == 'sph') {
    data.cards <- paste0(
      imp,
      '\n',
      kcode,
      '\nksrc  0 0 0',
      '\n\t    ', source, ' 0 0',
      '\n\t    0 ', source, ' 0',
      '\n\t    0 0 ', source,
      '\n\t    ', -source, ' 0 0',
      '\n\t    0 ', -source, ' 0',
      '\n\t    0 0 ', -source)
  } else if (shape == 'rcc') {
    data.cards <- paste0(
      imp,
      '\n',
      kcode,
      '\nksrc  0 0 ', (ht / 2) %>% round(2),
      '\n\t    ', source, ' 0 ', (ht / 2) %>% round(2),
      '\n\t    0 ', source, ' ', (ht / 2) %>% round(2),
      '\n\t    0 0 ', (5/6 * ht) %>% round(2),
      '\n\t    ', -source, ' 0 ', (ht / 2) %>% round(2),
      '\n\t    0 ', -source, ' ', (ht / 2) %>% round(2),
      '\n\t    0 0 ', (ht / 6) %>% round(2))
  }

  # write input to file
  file.name <- paste(gsub(' ', '-', title.card))
  input.deck <- paste(title.card, 'c', cell.cards, surface.cards, matl.cards, 'c',
   ↪ data.cards, 'c\nprint', sep = '\n')
```

```
  write(input.deck, file = paste0(file.name, '.i'))

  # run MCNP
  # if (!file.exists(paste0(file.name, '.o'))) {
  #   system(paste0(
  #     'C:/MCNP/MCNP_CODE/bin/mcnp6 inp=', file.name, '.i ',
  #     'outp=', file.name, '.o ',
  #     'runtpe=', file.name, '.runtpe ',
  #     'srctp=', file.name, '.srctp ',
  #     'tasks ', detectCores()))
  # }

}
```

A.1.2 Tabulate

```
# tabulate.R
#
# William John Zywiec
# The George Washington University

Tabulate <- function(code, source.dir, data.dir) {

  library(magrittr)

  setwd(data.dir)

  if (file.exists(paste0(code, '-dataset.RData'))) {

    dataset <- readRDS(paste0(code, '-dataset.RData'))
    cat('Loaded ', code, '-dataset.RData\n', sep = '')

  } else {

    # load function
    source(paste0(source.dir, '/scale.R'))

    output.files <- list.files(pattern = '\\.o$')

    # load output
    if (file.exists(paste0(code, '-output.csv'))) {
      output <- read.csv(paste0(code, '-output.csv'), fileEncoding = 'UTF-8-BOM')
       ↪ %>% na.omit()
      if (nrow(output) >= length(output.files)) {
        output <- output[sample(nrow(output)), ]
        dataset <- Scale(code, output)
        cat('Loaded ', code, '-dataset.RData\n', sep = '')
      } else {
        remove(output)
      }
    }

    if (!exists(paste0(code, '-output.csv')) && length(output.files) > 0) {

      mass <- rad <- thk <- ht <- vol <- conc <- hd <- keff <- sd <- numeric()
      form <- mod <- ref <- shape <- character()

      for (i in 1:length(output.files)) {

        if (any(grep('final result', readLines(output.files[i])))) {

          # set Pu mass (g), form, mod, rad (cm), and ref
          file.name <- gsub('\\.o', '', output.files[i]) %>% strsplit('-') %>%
           ↪ unlist()
          mass[i] <- as.numeric(file.name[1])
          form[i] <- file.name[2]
          mod[i] <- file.name[3]
          rad[i] <- as.numeric(file.name[4])
          ref[i] <- file.name[5]

          # set thk (cm) and shape
          if (ref[i] == 'none') {
            thk[i] <- 0
            shape[i] <- file.name[6]
          } else {
            thk[i] <- as.numeric(file.name[6])
            shape[i] <- file.name[7]
          }

          # set ht (cm)
          if (shape[i] == 'sph') {
            ht[i] <- 2 * rad[i]
          } else if (ref[i] == 'none') {
```

```
66            ht[i] <- as.numeric(file.name[7])
67          } else {
68            ht[i] <- as.numeric(file.name[8])
69          }
70
71          # calculate vol (cc)
72          if (shape[i] == 'sph') {
73            vol[i] <- 4/3 * pi * rad[i]^3
74          } else if (shape[i] == 'rcc') {
75            vol[i] <- pi * rad[i]^2 * ht[i]
76          }
77
78          # calculate conc (g/cc) and h/d
79          conc[i] <- (mass[i] / vol[i])
80          hd[i] <- (ht[i] / (2 * rad[i]))
81
82          # set keff and sd
83          final.result <- grep('final result', readLines(output.files[i]), value =
              ↪ TRUE) %>% strsplit('\\s+') %>% unlist()
84          keff[i] <- final.result[4]
85          sd[i] <- final.result[5]
86
87        }
88
89      }
90
91      output <- data.frame(
92        mass = mass,
93        form = form,
94        mod = mod,
95        rad = rad,
96        ref = ref,
97        thk = thk,
98        shape = shape,
99        ht = ht,
100       vol = vol,
101       conc = conc,
102       hd = hd,
103       keff = keff,
104       sd = sd)
105
106     output <- output[sample(nrow(output)), ]
107     write.csv(output, file = paste0(code, '-output.csv'), row.names = FALSE)
108
109     dataset <- Scale(code, output)
110     cat('Loaded ', code, '-dataset.RData\n', sep = '')
111
112   } else if (!exists('data') && length(output.files) == 0) {
113
114     stop('Could not find data\n')
115
116   }
117
118 }
119
120 return(dataset)
121
122 }
```

A.1.2.1 Scale

```r
# scale.R
#
# William John Zywiec
# The George Washington University

Scale <- function(code, output) {

  library(caret)
  library(dplyr)

  if (nrow(output) > 0) {

    output$shape <- output$ht <- output$hd <- NULL

    # one-hot encode categorical variables
    dummy <- dummyVars(~ ., data = output, sep = '')
    training.data <- data.frame(predict(dummy, newdata = output))
    training.data <- subset(training.data, sd < 0.001)

    # partition data
    # test.data <- subset(training.data, mass > 100 & rad > 7.62 & rad < 45.72)
    # test.data <- sample_n(test.data, round(nrow(training.data) * 0.2))
    # training.data <- anti_join(training.data, test.data)
    test.data <- sample_n(training.data, round(nrow(training.data) * 0.2))
    training.data <- anti_join(training.data, test.data)

    # scale data
    index <- c(1, 9, 20:22) # mass, rad, thk, vol, conc

    training.mean <- apply(training.data[index], 2, mean)
    training.sd <- apply(training.data[index], 2, sd)

    training.df <- training.data[-c(23, 24)]
    test.df <- test.data[-c(23, 24)]

    for (i in 1:length(index)) {
      training.df[index[i]] <- scale(training.df[index[i]], center =
       ↪ training.mean[i], scale = training.sd[i])
      test.df[index[i]] <- scale(test.df[index[i]], center = training.mean[i], scale
       ↪ = training.sd[i])
    }

    # convert data frames to matrices (Keras requirement)
    training.df <- as.matrix(training.df)
    test.df <- as.matrix(test.df)

    dataset <- list(output, training.data, training.mean, training.sd, training.df,
     ↪ test.data, test.df)
    names(dataset) <- c('output', 'training.data', 'training.mean', 'training.sd',
     ↪ 'training.df', 'test.data', 'test.df')
    saveRDS(dataset, file = paste0(code, '-dataset.RData'))

    return(dataset)

  }

}
```

A.1.3 Neural Network Metamodel

```
# nn.R
#
# William John Zywiec
# The George Washington University

NN <- function(dataset, batch.size, ensemble.size, epochs, layers, loss, opt.alg,
 ↪ learning.rate, replot, val.split, source.dir, test.dir) {

  library(keras)
  library(magrittr)

  # load functions
  source(paste0(source.dir, '/model.R'))
  source(paste0(source.dir, '/fit.R'))
  source(paste0(source.dir, '/plot.R'))
  source(paste0(source.dir, '/test.R'))

  # build loss function
  if (loss == 'sse') loss <- SSE <- function(y_true, y_pred) k_sum(k_pow(y_true -
   ↪ y_pred, 2))

  model.dir <- paste0(test.dir, '/model')
  dir.create(model.dir, recursive = TRUE, showWarnings = FALSE)

  setwd(model.dir)

  model.files <- list.files(pattern = '\\.h5$')

  # build and train metamodel
  metamodel <- history <- rep(list(0), length(ensemble.size))

  if (length(model.files) < ensemble.size) {
    for (i in (length(model.files) + 1):ensemble.size) {
      metamodel[[i]] <- Model(dataset, layers, loss, opt.alg, learning.rate)
      history[[i]] <- Fit(dataset, metamodel[[i]], batch.size, epochs, val.split)
      Plot(i, history[[i]])
      save_model_hdf5(metamodel[[i]], paste0(i, '.h5'))
    }
  } else if (replot == TRUE) {
    for (i in 1:ensemble.size) Plot(i)
  }

  model.files <- list.files(pattern = '\\.h5$')

  for (i in 1:ensemble.size) metamodel[[i]] <- load_model_hdf5(model.files[i],
   ↪ custom_objects = c(loss = loss))

  # rebuild metamodel
  remodel.dir <- paste0(test.dir, '/remodel')
  dir.create(remodel.dir, showWarnings = FALSE)

  setwd(remodel.dir)

  remodel.files <- list.files(pattern = '\\.h5$')
  history <- list()

  if (length(remodel.files) < ensemble.size * epochs / 10) {
    for (i in 1:ensemble.size) {
      remodel.files <- list.files(pattern = paste0(i, '-.+\\.h5$'))
      if (length(remodel.files) < epochs / 10) {
        history[[i]] <- Fit(dataset, metamodel[[i]], batch.size, epochs / 10,
         ↪ val.split, remodel.dir, i)
        Plot(i, history[[i]])
      } else {
        Plot(i)
      }
    }
```

```
  } else if (replot == TRUE) {
    for (i in 1:ensemble.size) Plot(i)
  }

  # test metamodel
  mae <- val.mae <- numeric()

  for (i in 1:ensemble.size) {
    metrics <- read.csv(paste0(i, '.csv'))
    mae[i] <- metrics$mae[which.min(metrics$mae + metrics$val.mae)]
    val.mae[i] <- metrics$val.mae[which.min(metrics$mae + metrics$val.mae)]
    metamodel[[i]] <- load_model_hdf5(paste0(i, '-',
     ↪ metrics$epoch[which.min(metrics$mae + metrics$val.mae)], '.h5'),
     ↪ custom_objects = c(loss = loss))
  }

  wt <- Test(dataset, metamodel, mae, val.mae, test.dir)

  return(list(metamodel, wt))

}
```

A.1.3.1 Model

```
# model.R
#
# William John Zywiec
# The George Washington University

Model <- function(dataset, layers, loss, opt.alg, learning.rate) {

  # library(keras)
  # library(magrittr)

  layers <- strsplit(layers, '-') %>% unlist() %>% as.integer()

  if (length(layers) == 1) {
    model <- keras_model_sequential() %>%
    layer_dense(units = layers[1], activation = 'relu', input_shape =
     ↪ dim(dataset$training.df)[2]) %>%
    layer_dense(units = 1, activation = 'linear')
  } else if (length(layers) == 2) {
    model <- keras_model_sequential() %>%
    layer_dense(units = layers[1], activation = 'relu', input_shape =
     ↪ dim(dataset$training.df)[2]) %>%
    layer_dense(units = layers[2], activation = 'relu') %>%
    layer_dense(units = 1, activation = 'linear')
  } else if (length(layers) == 3) {
    model <- keras_model_sequential() %>%
    layer_dense(units = layers[1], activation = 'relu', input_shape =
     ↪ dim(dataset$training.df)[2]) %>%
    layer_dense(units = layers[2], activation = 'relu') %>%
    layer_dense(units = layers[3], activation = 'relu') %>%
    layer_dense(units = 1, activation = 'linear')
  } else if (length(layers) == 4) {
    model <- keras_model_sequential() %>%
    layer_dense(units = layers[1], activation = 'relu', input_shape =
     ↪ dim(dataset$training.df)[2]) %>%
    layer_dense(units = layers[2], activation = 'relu') %>%
    layer_dense(units = layers[3], activation = 'relu') %>%
    layer_dense(units = layers[4], activation = 'relu') %>%
    layer_dense(units = 1, activation = 'linear')
  } else if (length(layers) == 5) {
    model <- keras_model_sequential() %>%
    layer_dense(units = layers[1], activation = 'relu', input_shape =
     ↪ dim(dataset$training.df)[2]) %>%
    layer_dense(units = layers[2], activation = 'relu') %>%
    layer_dense(units = layers[3], activation = 'relu') %>%
    layer_dense(units = layers[4], activation = 'relu') %>%
    layer_dense(units = layers[5], activation = 'relu') %>%
    layer_dense(units = 1, activation = 'linear')
  } else if (length(layers) == 6) {
    model <- keras_model_sequential() %>%
    layer_dense(units = layers[1], activation = 'relu', input_shape =
     ↪ dim(dataset$training.df)[2]) %>%
    layer_dense(units = layers[2], activation = 'relu') %>%
    layer_dense(units = layers[3], activation = 'relu') %>%
    layer_dense(units = layers[4], activation = 'relu') %>%
    layer_dense(units = layers[5], activation = 'relu') %>%
    layer_dense(units = layers[6], activation = 'relu') %>%
    layer_dense(units = 1, activation = 'linear')
  } else if (length(layers) == 7) {
    model <- keras_model_sequential() %>%
    layer_dense(units = layers[1], activation = 'relu', input_shape =
     ↪ dim(dataset$training.df)[2]) %>%
    layer_dense(units = layers[2], activation = 'relu') %>%
    layer_dense(units = layers[3], activation = 'relu') %>%
    layer_dense(units = layers[4], activation = 'relu') %>%
    layer_dense(units = layers[5], activation = 'relu') %>%
    layer_dense(units = layers[6], activation = 'relu') %>%
    layer_dense(units = layers[7], activation = 'relu') %>%
```

```
    layer_dense(units = 1, activation = 'linear')
  } else if (length(layers) == 8) {
    model <- keras_model_sequential() %>%
    layer_dense(units = layers[1], activation = 'relu', input_shape =
     ↪ dim(dataset$training.df)[2]) %>%
    layer_dense(units = layers[2], activation = 'relu') %>%
    layer_dense(units = layers[3], activation = 'relu') %>%
    layer_dense(units = layers[4], activation = 'relu') %>%
    layer_dense(units = layers[5], activation = 'relu') %>%
    layer_dense(units = layers[6], activation = 'relu') %>%
    layer_dense(units = layers[7], activation = 'relu') %>%
    layer_dense(units = layers[8], activation = 'relu') %>%
    layer_dense(units = 1, activation = 'linear')
  }

  if (opt.alg == 'adadelta') {
    model %>% compile(
      loss = loss,
      optimizer = optimizer_adadelta(lr = learning.rate),
      metrics = c('mean_absolute_error'))
  } else if (opt.alg == 'adagrad') {
    model %>% compile(
      loss = loss,
      optimizer = optimizer_adagrad(lr = learning.rate),
      metrics = c('mean_absolute_error'))
  } else if (opt.alg == 'adam') {
    model %>% compile(
      loss = loss,
      optimizer = optimizer_adam(lr = learning.rate),
      metrics = c('mean_absolute_error'))
  } else if (opt.alg == 'adamax') {
    model %>% compile(
      loss = loss,
      optimizer = optimizer_adamax(lr = learning.rate),
      metrics = c('mean_absolute_error'))
  } else if (opt.alg == 'nadam') {
    model %>% compile(
      loss = loss,
      optimizer = optimizer_nadam(lr = learning.rate),
      metrics = c('mean_absolute_error'))
  } else if (opt.alg == 'rmsprop') {
    model %>% compile(
      loss = loss,
      optimizer = optimizer_rmsprop(lr = learning.rate),
      metrics = c('mean_absolute_error'))
  }

}
```

A.1.3.2 Fit

```
# fit.R
#
# William John Zywiec
# The George Washington University

Fit <- function(dataset, model, batch.size, epochs, val.split, remodel.dir, i) {

  # library(keras)
  # library(magrittr)

  if (missing(i)) {
    model %>% fit(
      dataset$training.df,
      dataset$training.data$keff,
      batch_size = batch.size,
      epochs = epochs,
      validation_split = val.split,
      verbose = FALSE)
  } else {
    checkpoint <- callback_model_checkpoint(paste0(remodel.dir, '/', i,
    ↪ '-{epoch:1d}.h5'), monitor = 'mean_absolute_error')
    model %>% fit(
      dataset$training.df,
      dataset$training.data$keff,
      batch_size = batch.size,
      epochs = epochs,
      validation_split = val.split,
      verbose = FALSE,
      callbacks = c(checkpoint))
  }

}
```

A.1.3.3 Plot

```
# plot.R
#
# William John Zywiec
# The George Washington University

Plot <- function(x, history) {

  library(ggplot2)
  library(magrittr)
  library(scales)

  # set theme
  new.theme <- theme_gray() + theme(axis.text = element_text(color = 'black', size = 11), text = element_text(color = 'black', family = 'serif', size = 11))
  theme_set(new.theme)

  if (missing('history')) {
    history <- read.csv(paste0(x, '.csv'), header = TRUE)
  } else {
    history <- data.frame(
      epoch = 1:length(history$metrics$val_loss),
      val.loss = history$metrics$val_loss,
      val.mae = history$metrics$val_mean_absolute_error,
      loss = history$metrics$loss,
      mae = history$metrics$mean_absolute_error)
    write.csv(history, file = paste0(x, '.csv'), row.names = FALSE)
  }

  ggplot(history, aes(x = epoch)) +
  geom_line(aes(y = val.mae, color = 'cross-validation data')) +
  geom_line(aes(y = mae, color = 'training data')) +
  geom_point(aes(x = which.min(history$mae), y = min(history$mae), color = 'training minimum')) +
  guides(color = guide_legend(override.aes = list(linetype = c(1, 1, NA), shape = c(NA, NA, 16)))) +
  scale_color_manual('', breaks = c('training data', 'cross-validation data', 'training minimum'), values = c('black', '#a9a9a9', 'red')) +
  scale_x_continuous() +
  scale_y_log10(breaks = c(2e-04, 2e-03, 2e-02, 2e-01), limits = c(2e-04, 2e-01)) +
  theme(
    legend.position = 'bottom',
    legend.spacing.x = unit(0.2, 'cm'),
    legend.text = element_text(size = 11),
    legend.title = element_blank()) +
  ylab('mean absolute error') +
  annotate(
    geom = 'text',
    x = which.min(history$mae),
    y = min(history$mae),
    vjust = 1.9,
    label = format(min(history$mae), digits = 3, scientific = TRUE),
    color = 'red',
    family = 'serif',
    size = 3.5)

  ggsave(paste0(x, '.png'), dpi = 1000, height = 4, width = 6.5) %>% suppressMessages()

}
```

A.1.3.4 Test

```
# test.R
#
# William John Zywiec
# The George Washington University

Test <- function(dataset, metamodel, mae, val.mae, test.dir) {

  library(keras)
  library(magrittr)

  setwd(test.dir)

  test.data <- dataset$test.data
  test.mae <- avg <- nm <- bfgs <- sa <- numeric()
  test.pred <- matrix(nrow = nrow(dataset$test.df), ncol = length(metamodel))
  nm.wt <- bfgs.wt <- sa.wt <- list()

  # build objective function
  Objective <- function(x) mean(abs(test.data$keff - rowSums(test.pred * x, na.rm =
   ↪ TRUE)))

  # minimize objective function
  for (i in 1:length(metamodel)) {

    test.pred[ , i] <- metamodel[[i]] %>% predict(dataset$test.df)
    test.mae[i] <- mean(abs(test.data$keff - test.pred[ , i]))

    avg[i] <- mean(abs(test.data$keff - rowMeans(test.pred, na.rm = TRUE)))

    nm.wt[[i]] <- optim(rep(0, i), Objective, method = 'Nelder-Mead', lower = 0)
    bfgs.wt[[i]] <- optim(rep(0, i), Objective, method = 'BFGS', lower = 0)
    sa.wt[[i]] <- optim(rep(0, i), Objective, method = 'SANN', lower = 0)

    nm[i] <- mean(abs(test.data$keff - rowSums(test.pred * nm.wt[[i]][[1]], na.rm =
     ↪ TRUE)))
    bfgs[i] <- mean(abs(test.data$keff - rowSums(test.pred * bfgs.wt[[i]][[1]],
     ↪ na.rm = TRUE)))
    sa[i] <- mean(abs(test.data$keff - rowSums(test.pred * sa.wt[[i]][[1]], na.rm =
     ↪ TRUE)))

    if (i == 1) {
      progress.bar <- txtProgressBar(min = 0, max = length(metamodel), style = 3)
      setTxtProgressBar(progress.bar, i)
      if (i == length(metamodel)) {
        cat('\n\n', sep = '')
      }
    } else if (i == length(metamodel)) {
      setTxtProgressBar(progress.bar, i)
      cat('\n\n', sep = '')
    } else {
      setTxtProgressBar(progress.bar, i)
    }

  }

  test.results <- data.frame(avg = avg, nm = nm, bfgs = bfgs, sa = sa)
  write.csv(test.results, file = 'test-results.csv', row.names = FALSE)

  cat('Mean Training MAE = ', mean(mae) %>% sprintf('%.6f', .), '\n', sep = '')
  cat('Mean Cross-Validation MAE = ', mean(val.mae) %>% sprintf('%.6f', .), '\n',
   ↪ sep = '')
  cat('Mean Test MAE = ', mean(test.mae) %>% sprintf('%.6f', .), '\n\n', sep = '')
  cat('Ensemble Test MAE = ', avg[length(metamodel)] %>% sprintf('%.9f', .), '\n',
   ↪ sep = '')
  cat('Ensemble Test MAE = ', nm[length(metamodel)] %>% sprintf('%.9f', .), '
   ↪ (Nelder-Mead)\n', sep = '')
```

```
cat('Ensemble Test MAE = ', bfgs[length(metamodel)] %>% sprintf('%.9f', .), '
 ↪ (BFGS)\n', sep = '')
cat('Ensemble Test MAE = ', sa[length(metamodel)] %>% sprintf('%.9f', .), '
 ↪ (SA)\n', sep = '')

test.min <- min(c(avg[which.min(avg)], nm[which.min(nm)], bfgs[which.min(bfgs)],
 ↪ sa[which.min(sa)]))

if (test.min == nm[which.min(nm)]) {
  wt <- nm.wt[[which.min(nm)]]
} else if (test.min == bfgs[which.min(bfgs)]) {
  wt <- bfgs.wt[[which.min(bfgs)]]
} else if (test.min == sa[which.min(sa)]) {
  wt <- sa.wt[[which.min(sa)]]
} else {
  wt <- 0
}

if (length(wt[[1]]) < length(metamodel) && wt[[1]][1] != 0) {
  cat('-\nTest MAE reaches a local minimum with ', length(wt[[1]]), ' neural
   ↪ networks\n\n', sep = '')
  cat('Ensemble Test MAE = ', avg[length(wt[[1]])] %>% sprintf('%.9f', .), '\n',
   ↪ sep = '')
  cat('Ensemble Test MAE = ', nm[length(wt[[1]])] %>% sprintf('%.9f', .), '
   ↪ (Nelder-Mead)\n', sep = '')
  cat('Ensemble Test MAE = ', bfgs[length(wt[[1]])] %>% sprintf('%.9f', .), '
   ↪ (BFGS)\n', sep = '')
  cat('Ensemble Test MAE = ', sa[length(wt[[1]])] %>% sprintf('%.9f', .), '
   ↪ (SA)\n', sep = '')
}

training.data <- dataset$training.data

# adjust predicted keff values
Adjust <- function(x, dataset, training.data, test.data, test.pred, nm.wt,
 ↪ bfgs.wt, sa.wt) {

  training.pred <- matrix(nrow = nrow(dataset$training.df), ncol = x)

  if (x == 1) {

    training.pred[ , 1] <- metamodel[[1]] %>% predict(dataset$training.df)

    training.data$avg <- training.pred[ , 1]
    training.data$nm <- training.pred[ , 1] * nm.wt[[x]][[1]]
    training.data$bfgs <- training.pred[ , 1] * bfgs.wt[[x]][[1]]
    training.data$sa <- training.pred[ , 1] * sa.wt[[x]][[1]]

    test.data$avg <- test.pred[ , 1]
    test.data$nm <- test.pred[ , 1] * nm.wt[[x]][[1]]
    test.data$bfgs <- test.pred[ , 1] * bfgs.wt[[x]][[1]]
    test.data$sa <- test.pred[ , 1] * sa.wt[[x]][[1]]

  } else {

    for (i in 1:x) {
      training.pred[ , i] <- metamodel[[i]] %>% predict(dataset$training.df)
    }

    training.data$avg <- rowMeans(training.pred[ , 1:x])
    training.data$nm <- rowSums(training.pred[ , 1:x] * nm.wt[[x]][[1]])
    training.data$bfgs <- rowSums(training.pred[ , 1:x] * bfgs.wt[[x]][[1]])
    training.data$sa <- rowSums(training.pred[ , 1:x] * sa.wt[[x]][[1]])

    test.data$avg <- rowMeans(test.pred[ , 1:x])
    test.data$nm <- rowSums(test.pred[ , 1:x] * nm.wt[[x]][[1]])
    test.data$bfgs <- rowSums(test.pred[ , 1:x] * bfgs.wt[[x]][[1]])
    test.data$sa <- rowSums(test.pred[ , 1:x] * sa.wt[[x]][[1]])

  }

```

```
122     write.csv(training.data, file = 'training-data.csv', row.names = FALSE)
123     write.csv(test.data, file = 'test-data.csv', row.names = FALSE)
124
125   }
126
127   Adjust(length(wt[[1]]), dataset, training.data, test.data, test.pred, nm.wt,
        ↪ bfgs.wt, sa.wt)
128
129   return(wt)
130
131 }
```

A.1.4 Bayesian Network

```
# bn.R
#
# William John Zywiec
# The George Washington University

BN <- function(dist, dist.dir) {

  library(bnlearn)

  setwd(dist.dir)

  # build graph
  nodes <- c('op', 'ctrl', 'mass', 'form', 'mod', 'rad', 'ref', 'thk')
  dag <- empty.graph(nodes = nodes)

  for (i in 2:length(nodes)) {
    dag <- set.arc(dag, 'op', nodes[i])
    if (i > 2) dag <- set.arc(dag, 'ctrl', nodes[i])
  }

  # build conditional probability tables
  op <- c('large sample', 'machining', 'metallurgy', 'small sample', 'solution',
   ↪ 'waste')
  ctrl <- c('A', 'B', 'C', 'D', 'E', 'M', 'P')
  mass <- seq(0, 4000, 1)
  form <- c('alpha', 'puo2')
  mod <- c('mgo', 'ch2', 'sepiolite', 'h2o', 'none')
  rad <- seq(0, 18, 0.25) * 2.54
  ref <- c('al', 'be', 'du', 'graphite', 'pb', 'mgo', 'ch2', 'ss304', 'h2o', 'none')
  thk <- seq(0, 6, 0.25) * 2.54

  op.cpt <- matrix(c(
    3.50e-01 ,  # large sample
    7.50e-02 ,  # machining
    1.25e-01 ,  # metallurgy
    3.00e-01 ,  # small sample
    1.00e-01 ,  # solution
    5.00e-02 ), # waste
    ncol = 1, dimnames = list(op, NULL))

  ctrl.cpt <- matrix(c(
    # A           B             C             D             E             M
     ↪             P
    5.71429e-01 , 2.85714e-02 , 2.85714e-02 , 1.71428e-01 , 8.57142e-02 ,
     ↪ 1.14286e-01 , 0           ,  # large sample
    1.53846e-01 , 0           , 0           , 0           , 0           ,
     ↪ 8.46154e-01 , 0           ,  # machining
    6.15385e-01 , 7.69231e-02 , 0           , 0           , 3.07692e-01 , 0
     ↪           , 0           ,  # metallurgy
    9.37500e-01 , 6.25000e-02 , 0           , 0           , 0           , 0
     ↪           , 0           ,  # small sample
    1           , 0           , 0           , 0           , 0           , 0
     ↪           , 0           ,  # solution
    5.00000e-01 , 0           , 0           , 0           , 0           , 0
     ↪           , 5.00000e-01 ), # waste
    nrow = 7, ncol = 6, dimnames = list(ctrl, op))

  mass.cpt <- array(unlist(readRDS(paste0('mass-', dist, '.RData'))), dim = c(4001,
   ↪ 7, 6), dimnames = list('mass' = mass, 'ctrl' = ctrl, 'op' = op))
  rad.cpt <- array(unlist(readRDS(paste0('rad-', dist, '.RData'))), dim = c(73, 7,
   ↪ 6), dimnames = list('rad' = rad, 'ctrl' = ctrl, 'op' = op))
  thk.cpt <- array(unlist(readRDS(paste0('thk-', dist, '.RData'))), dim = c(25, 7,
   ↪ 6), dimnames = list('thk' = thk, 'ctrl' = ctrl, 'op' = op))

  form.cpt <- array(c(
  # large sample
    9.50000E-01 , 5.00000e-02 ,  # A
```

```
    1            , 0            ,  # B
    1            , 0            ,  # C
    1            , 0            ,  # D
    1            , 0            ,  # E
    1            , 0            ,  # M
    1            , 0            ,  # P (NULL)
  # machining
    1            , 0            ,  # A
    1            , 0            ,  # B (NULL)
    1            , 0            ,  # C (NULL)
    1            , 0            ,  # D (NULL)
    1            , 0            ,  # E (NULL)
    1            , 0            ,  # M
    1            , 0            ,  # P (NULL)
  # metallurgy
    9.50000E-01  , 5.00000e-02  ,  # A
    1            , 0            ,  # B
    1            , 0            ,  # C (NULL)
    1            , 0            ,  # D (NULL)
    5.00000e-01  , 5.00000e-01  ,  # E
    1            , 0            ,  # M (NULL)
    1            , 0            ,  # P (NULL)
  # small sample
    1            , 0            ,  # A
    1            , 0            ,  # B
    1            , 0            ,  # C (NULL)
    1            , 0            ,  # D (NULL)
    1            , 0            ,  # E (NULL)
    1            , 0            ,  # M (NULL)
    1            , 0            ,  # P (NULL)
  # solution
    9.50000E-01  , 5.00000e-02  ,  # A
    1            , 0            ,  # B (NULL)
    1            , 0            ,  # C (NULL)
    1            , 0            ,  # D (NULL)
    1            , 0            ,  # E (NULL)
    1            , 0            ,  # M (NULL)
    1            , 0            ,  # P (NULL)
  # waste
    0            , 1            ,  # A
    1            , 0            ,  # B (NULL)
    1            , 0            ,  # C (NULL)
    1            , 0            ,  # D (NULL)
    1            , 0            ,  # E (NULL)
    1            , 0            ,  # M (NULL)
    0            , 1            ), # P
    dim = c(2, 7, 6), dimnames = list('form' = form, 'ctrl' = ctrl, 'op' = op))

  mod.cpt <- array(c(
  # large sample
    8.72522E-02  , 1.62535E-01  , 2.12734E-04  , 0            , 7.50000E-01  ,  # A
    8.72522E-02  , 1.62535E-01  , 2.12734E-04  , 0            , 7.50000E-01  ,  # B
    8.72522E-02  , 1.62535E-01  , 2.12734E-04  , 0            , 7.50000E-01  ,  # C
    8.72522E-02  , 1.62535E-01  , 2.12734E-04  , 0            , 7.50000E-01  ,  # D
    1.74504E-02  , 3.25070E-02  , 4.25468E-05  , 0            , 9.50000E-01  ,  # E
    8.72522E-02  , 1.62535E-01  , 2.12734E-04  , 0            , 7.50000E-01  ,  # M
    0            , 0            , 0            , 0            , 1            ,  # P (NULL)
  # machining
    6.49960E-03  , 2.42690E-01  , 8.10672E-04  , 0            , 7.50000E-01  ,  # A
    0            , 0            , 0            , 0            , 1            ,  # B (NULL)
    0            , 0            , 0            , 0            , 1            ,  # C (NULL)
    0            , 0            , 0            , 0            , 1            ,  # D (NULL)
    0            , 0            , 0            , 0            , 1            ,  # E (NULL)
    6.49960E-03  , 2.42690E-01  , 8.10672E-04  , 0            , 7.50000E-01  ,  # M
    0            , 0            , 0            , 0            , 1            ,  # P (NULL)
  # metallurgy
    6.31463E-02  , 1.86759E-01  , 9.47006E-05  , 0            , 7.50000E-01  ,  # A
    6.31463E-02  , 1.86759E-01  , 9.47006E-05  , 0            , 7.50000E-01  ,  # B
    0            , 0            , 0            , 0            , 1            ,  # C (NULL)
    0            , 0            , 0            , 0            , 1            ,  # D (NULL)
    1.26293E-02  , 3.73518E-02  , 1.89401E-05  , 0            , 9.50000E-01  ,  # E
```

```
    0           , 0           , 0           , 0           , 1           ,  # M (NULL)
    0           , 0           , 0           , 0           , 1           ,  # P (NULL)
  # small sample
    1.66387E-02 , 2.33079E-01 , 2.82013E-04 , 0           , 7.50000E-01 ,  # A
    1.66387E-02 , 2.33079E-01 , 2.82013E-04 , 0           , 7.50000E-01 ,  # B
    0           , 0           , 0           , 0           , 1           ,  # C (NULL)
    0           , 0           , 0           , 0           , 1           ,  # D (NULL)
    0           , 0           , 0           , 0           , 1           ,  # E (NULL)
    0           , 0           , 0           , 0           , 1           ,  # M (NULL)
    0           , 0           , 0           , 0           , 1           ,  # P (NULL)
  # solution
    3.46069E-04 , 2.48339E-01 , 1.31506E-03 , 0           , 7.50000E-01 ,  # A
    0           , 0           , 0           , 0           , 1           ,  # B (NULL)
    0           , 0           , 0           , 0           , 1           ,  # C (NULL)
    0           , 0           , 0           , 0           , 1           ,  # D (NULL)
    0           , 0           , 0           , 0           , 1           ,  # E (NULL)
    0           , 0           , 0           , 0           , 1           ,  # M (NULL)
    0           , 0           , 0           , 0           , 1           ,  # P (NULL)
  # waste
    7.84143E-02 , 1.66521E-01 , 5.06462E-03 , 0           , 7.50000E-01 ,  # A
    0           , 0           , 0           , 0           , 1           ,  # B (NULL)
    0           , 0           , 0           , 0           , 1           ,  # C (NULL)
    0           , 0           , 0           , 0           , 1           ,  # D (NULL)
    0           , 0           , 0           , 0           , 1           ,  # E (NULL)
    0           , 0           , 0           , 0           , 1           ,  # M (NULL)
    7.84143E-02 , 1.66521E-01 , 5.06462E-03 , 0           , 7.50000E-01 ), # P
    dim = c(5, 7, 6), dimnames = list('mod' = mod, 'ctrl' = ctrl, 'op' = op))

  ref.cpt <- array(c(
  # large sample
    2.51452E-02 , 1.62768E-02 , 1.62768E-02 , 1.96831E-04 , 1.62768E-02 ,
     ↪ 2.28734E-01 , 4.26090E-01 , 2.71004E-01 , 0           , 0           ,  # A
    2.51452E-02 , 1.62768E-02 , 1.62768E-02 , 1.96831E-04 , 1.62768E-02 ,
     ↪ 2.28734E-01 , 4.26090E-01 , 2.71004E-01 , 0           , 0           ,  # B
    1.25726E-03 , 8.13842E-04 , 8.13842E-04 , 9.84155E-06 , 8.13842E-04 ,
     ↪ 1.14367E-02 , 2.13045E-02 , 1.35502E-02 , 0           , 9.50000E-01 ,  # C
    2.51452E-02 , 1.62768E-02 , 1.62768E-02 , 1.96831E-04 , 1.62768E-02 ,
     ↪ 2.28734E-01 , 4.26090E-01 , 2.71004E-01 , 0           , 0           ,  # D
    2.51452E-02 , 1.62768E-02 , 1.62768E-02 , 1.96831E-04 , 1.62768E-02 ,
     ↪ 2.28734E-01 , 4.26090E-01 , 2.71004E-01 , 0           , 0           ,  # E
    2.51452E-02 , 1.62768E-02 , 1.62768E-02 , 1.96831E-04 , 1.62768E-02 ,
     ↪ 2.28734E-01 , 4.26090E-01 , 2.71004E-01 , 0           , 0           ,  # M
    0           , 0           , 0           , 0           , 0           , 0
     ↪           , 0           , 0           , 0           , 1           ,  # P
     ↪ (NULL)
  # machining
    6.56559E-02 , 1.85162E-02 , 1.85162E-02 , 4.17659E-05 , 1.85162E-02 ,
     ↪ 1.90870E-02 , 7.12693E-01 , 1.46974E-01 , 0           , 0           ,  # A
    0           , 0           , 0           , 0           , 0           , 0
     ↪           , 0           , 0           , 0           , 1           ,  # B
     ↪ (NULL)
    0           , 0           , 0           , 0           , 0           , 0
     ↪           , 0           , 0           , 0           , 1           ,  # C
     ↪ (NULL)
    0           , 0           , 0           , 0           , 0           , 0
     ↪           , 0           , 0           , 0           , 1           ,  # D
     ↪ (NULL)
    0           , 0           , 0           , 0           , 0           , 0
     ↪           , 0           , 0           , 0           , 1           ,  # E
     ↪ (NULL)
    6.56559E-02 , 1.85162E-02 , 1.85162E-02 , 4.17659E-05 , 1.85162E-02 ,
     ↪ 1.90870E-02 , 7.12693E-01 , 1.46974E-01 , 0           , 0           ,  # M
    0           , 0           , 0           , 0           , 0           , 0
     ↪           , 0           , 0           , 0           , 1           ,  # P
     ↪ (NULL)
  # metallurgy
    3.18113E-02 , 1.66920E-02 , 1.66920E-02 , 2.17439E-04 , 1.66920E-02 ,
     ↪ 1.44988E-01 , 4.28811E-01 , 3.44097E-01 , 0           , 0           ,  # A
    3.18113E-02 , 1.66920E-02 , 1.66920E-02 , 2.17439E-04 , 1.66920E-02 ,
     ↪ 1.44988E-01 , 4.28811E-01 , 3.44097E-01 , 0           , 0           ,  # B
```

```
    0           , 0           , 0           , 0           , 0           , 0
     ↪            , 0           , 0           , 0           , 1           ,  # C
     ↪ (NULL)
    0           , 0           , 0           , 0           , 0           , 0
     ↪            , 0           , 0           , 0           , 1           ,  # D
     ↪ (NULL)
    3.18113E-02 , 1.66920E-02 , 1.66920E-02 , 2.17439E-04 , 1.66920E-02 ,
     ↪ 1.44988E-01 , 4.28811E-01 , 3.44097E-01 , 0           , 0           ,  # E
    0           , 0           , 0           , 0           , 0           , 0
     ↪            , 0           , 0           , 0           , 1           ,  # M
     ↪ (NULL)
    0           , 0           , 0           , 0           , 0           , 0
     ↪            , 0           , 0           , 0           , 1           ,  # P
     ↪ (NULL)
  # small sample
    4.79521E-02 , 1.32551E-02 , 1.32551E-02 , 5.13268E-04 , 1.32551E-02 ,
     ↪ 5.14808E-02 , 7.21154E-01 , 1.39134E-01 , 0           , 0           ,  # A
    4.79521E-02 , 1.32551E-02 , 1.32551E-02 , 5.13268E-04 , 1.32551E-02 ,
     ↪ 5.14808E-02 , 7.21154E-01 , 1.39134E-01 , 0           , 0           ,  # B
    0           , 0           , 0           , 0           , 0           , 0
     ↪            , 0           , 0           , 0           , 1           ,  # C
     ↪ (NULL)
    0           , 0           , 0           , 0           , 0           , 0
     ↪            , 0           , 0           , 0           , 1           ,  # D
     ↪ (NULL)
    0           , 0           , 0           , 0           , 0           , 0
     ↪            , 0           , 0           , 0           , 1           ,  # E
     ↪ (NULL)
    0           , 0           , 0           , 0           , 0           , 0
     ↪            , 0           , 0           , 0           , 1           ,  # M
     ↪ (NULL)
    0           , 0           , 0           , 0           , 0           , 0
     ↪            , 0           , 0           , 0           , 1           ,  # P
     ↪ (NULL)
  # solution
    8.46767E-02 , 4.01848E-02 , 4.01848E-02 , 0           , 4.01848E-02 ,
     ↪ 9.40852E-04 , 6.75155E-01 , 1.18673E-01 , 0           , 0           ,  # A
    0           , 0           , 0           , 0           , 0           , 0
     ↪            , 0           , 0           , 0           , 1           ,  # B
     ↪ (NULL)
    0           , 0           , 0           , 0           , 0           , 0
     ↪            , 0           , 0           , 0           , 1           ,  # C
     ↪ (NULL)
    0           , 0           , 0           , 0           , 0           , 0
     ↪            , 0           , 0           , 0           , 1           ,  # D
     ↪ (NULL)
    0           , 0           , 0           , 0           , 0           , 0
     ↪            , 0           , 0           , 0           , 1           ,  # E
     ↪ (NULL)
    0           , 0           , 0           , 0           , 0           , 0
     ↪            , 0           , 0           , 0           , 1           ,  # M
     ↪ (NULL)
    0           , 0           , 0           , 0           , 0           , 0
     ↪            , 0           , 0           , 0           , 1           ,  # P
     ↪ (NULL)
  # waste
    1.59817E-02 , 5.38644E-02 , 5.38644E-02 , 2.53678E-04 , 5.38644E-02 ,
     ↪ 2.27803E-01 , 4.83765E-01 , 1.10604E-01 , 0           , 0           ,  # A
    0           , 0           , 0           , 0           , 0           , 0
     ↪            , 0           , 0           , 0           , 1           ,  # B
     ↪ (NULL)
    0           , 0           , 0           , 0           , 0           , 0
     ↪            , 0           , 0           , 0           , 1           ,  # C
     ↪ (NULL)
    0           , 0           , 0           , 0           , 0           , 0
     ↪            , 0           , 0           , 0           , 1           ,  # D
     ↪ (NULL)
    0           , 0           , 0           , 0           , 0           , 0
     ↪            , 0           , 0           , 0           , 1           ,  # E
     ↪ (NULL)
```

```
    0            , 0            , 0            , 0            , 0            , 0
     ↪            , 0            , 0            , 0            , 1            ,  # M
     ↪ (NULL)
    1.59817E-02 , 5.38644E-02 , 5.38644E-02 , 2.53678E-04 , 5.38644E-02 ,
     ↪ 2.27803E-01 , 4.83765E-01 , 1.10604E-01 , 0           , 0           ), # P
    dim = c(10, 7, 6), dimnames = list('ref' = ref, 'ctrl' = ctrl, 'op' = op))

  bn <- list(
    op = op.cpt,
    ctrl = ctrl.cpt,
    mass = mass.cpt,
    form = form.cpt,
    mod = mod.cpt,
    rad = rad.cpt,
    ref = ref.cpt,
    thk = thk.cpt)

  bn <- custom.fit(dag, dist = bn)

  return(bn)

}
```

A.1.5 Risk

```
# risk.R
#
# William John Zywiec
# The George Washington University

Risk <- function(bn, dataset, metamodel, risk.pool, sample.size, source.dir,
 ↪ test.dir) {

  library(dplyr)

  # load function
  source(paste0(source.dir, '/sample.R'))

  risk.dir <- paste0(test.dir, '/risk/', dist, '-', sample.size)
  dir.create(risk.dir, recursive = TRUE, showWarnings = FALSE)

  setwd(risk.dir)

  # adjust risk pool and sample size (12 GB RAM)
  if (sample.size > 5e+08) {
    risk.pool <- risk.pool * sample.size / 5e+08
    sample.size <- 5e+08
  }

  if (file.exists('risk.csv')) {

    bn.data <- readRDS('bn-data.RData')
    risk <- read.csv('risk.csv', fileEncoding = 'UTF-8-BOM')
    cat('Risk = ', format(mean(risk$risk), digits = 3, scientific = TRUE), '\n', sep
     ↪ = '')
    cat('Variance = ', format(var(risk$risk), digits = 3, scientific = TRUE), '\n',
     ↪ sep = '')

  } else {

    bn.data <- list()
    risk <- pooled.risk <- numeric()

    for (i in 1:risk.pool) {
      bn.data[[i]] <- Sample(bn, dataset, metamodel, sample.size)
      risk[i] <- length(bn.data[[i]]$keff[bn.data[[i]]$keff >= 0.95]) / sample.size
       ↪ # USL = 0.95
      if (i == 1) {
        progress.bar <- txtProgressBar(min = 0, max = risk.pool, style = 3)
        setTxtProgressBar(progress.bar, i)
        if (i == risk.pool) {
          cat('\n', sep = '')
        }
      } else if (i == risk.pool) {
        setTxtProgressBar(progress.bar, i)
        cat('\n', sep = '')
      } else {
        setTxtProgressBar(progress.bar, i)
      }
    }

    if (risk.pool > 100) {
      breaks <- seq((length(risk) / 100), length(risk), (length(risk) / 100))
      for (i in 1:100) pooled.risk[i] <- sum(risk[(breaks[i] - (length(risk) / 100 -
       ↪ 1)):breaks[i]])
      risk <- pooled.risk
    }

    saveRDS(bn.data, file = 'bn-data.RData')
    write.csv(as.data.frame(risk, col.names = 'risk'), file = 'risk.csv', row.names
     ↪ = FALSE)
```

```
61      cat('\nRisk = ', format(mean(risk), digits = 3, scientific = TRUE), '\n', sep =
        ↪ '')
62      cat('Variance = ', format(var(risk), digits = 3, scientific = TRUE), '\n', sep =
        ↪ '')
63
64    }
65
66    bn.data <- bind_rows(bn.data)
67
68    return(list(risk, bn.data))
69
70  }
```

A.1.5.1 Sample

```
# sample.R
#
# William John Zywiec
# The George Washington University

Sample <- function(bn, dataset, metamodel, sample.size) {

  library(bnlearn)
  library(magrittr)
  library(parallel)

  cluster <- makeCluster((detectCores() / 2), type = 'SOCK')

  # sample conditional probability tables
  bn.data <- cpdist(
    bn,
    nodes = c('mass', 'form', 'mod', 'rad', 'ref', 'thk'),
    evidence = (as.integer(mass) > 120) & (as.integer(rad) > 7) & mod == 'ch2',
    # evidence = TRUE,
    batch = sample.size,
    cluster = cluster,
    n = sample.size) %>% na.omit()

  stopCluster(cluster)

  bn.data[[1]] <- unlist(bn.data[[1]]) %>% as.character() %>% as.numeric() # mass
  bn.data[[2]] <- unlist(bn.data[[2]])                                     # form
  bn.data[[3]] <- unlist(bn.data[[3]])                                     # mod
  bn.data[[4]] <- unlist(bn.data[[4]]) %>% as.character() %>% as.numeric() # rad
  bn.data[[5]] <- unlist(bn.data[[5]])                                     # ref
  bn.data[[6]] <- unlist(bn.data[[6]]) %>% as.character() %>% as.numeric() # thk

  # set Pu density (g/cc)
  pu.density <- ifelse((bn.data$form == 'alpha'), 19.86, 11.5)

  # calculate vol (cc)
  vol <- 4/3 * pi * bn.data$rad^3

  # fix mod, vol (cc), and rad (cm)
  bn.data$mod[vol <= bn.data$mass / pu.density] <- 'none'
  vol[vol <= bn.data$mass / pu.density] <- bn.data$mass[vol <= bn.data$mass /
   ↪ pu.density] / pu.density[vol <= bn.data$mass / pu.density]
  bn.data$rad <- (3/4 * vol / pi)^(1/3)

  # fix ref and thk (cm)
  bn.data$ref[bn.data$thk == 0] <- 'none'
  bn.data$thk[bn.data$ref == 'none'] <- 0

  # set conc (g/cc)
  conc <- ifelse((vol == 0), 0, (bn.data$mass / vol))

  # set form, vol (cc), and conc (g/cc)
  bn.data$form <- ifelse((pu.density == 19.86), 'alpha', 'puo2')
  bn.data$vol <- vol
  bn.data$conc <- conc

  library(caret)

  # one-hot encode categorical variables
  dummy <- dummyVars(~ ., data = bn.data, sep = '')
  bn.df <- data.frame(predict(dummy, newdata = bn.data))

  bn.df <- bn.df[head(names(dataset$training.data), -2)]

  # scale data
  bn.df$mass <- scale(bn.df$mass, center = dataset$training.mean[1], scale =
   ↪ dataset$training.sd[1])
```

```
  bn.df$rad <- scale(bn.df$rad, center = dataset$training.mean[2], scale =
   ↪ dataset$training.sd[2])
  bn.df$thk <- scale(bn.df$thk, center = dataset$training.mean[3], scale =
   ↪ dataset$training.sd[3])
  bn.df$vol <- scale(bn.df$vol, center = dataset$training.mean[4], scale =
   ↪ dataset$training.sd[4])
  bn.df$conc <- scale(bn.df$conc, center = dataset$training.mean[5], scale =
   ↪ dataset$training.sd[5])

  # convert data frame to matrix (Keras requirement)
  bn.df <- as.matrix(bn.df)

  library(keras)

  # predict keff values
  bn.data$keff <- metamodel[[1]][[1]] %>% predict(bn.df)

  bn.df <- cbind(bn.df, bn.data$keff) %>% subset(bn.data$keff > 0.9)
  bn.df <- bn.df[ , -ncol(bn.df)]

  bn.data <- subset(bn.data, keff > 0.9)

  if (typeof(metamodel[[2]]) == 'list') {
    keff <- matrix(nrow = nrow(bn.df), ncol = length(metamodel[[2]][[1]]))
    for (i in 1:length(metamodel[[2]][[1]])) keff[ , i] <- metamodel[[1]][[i]] %>%
     ↪ predict(bn.df)
    bn.data$keff <- rowSums(keff * metamodel[[2]][[1]])
  } else {
    keff <- matrix(nrow = nrow(bn.df), ncol = length(metamodel[[1]]))
    for (i in 1:length(metamodel[[1]])) keff[ , i] <- metamodel[[1]][[i]] %>%
     ↪ predict(bn.df)
    bn.data$keff <- rowMeans(keff)
  }

  return(bn.data)

}
```

Appendix B: Truncated Probability Distributions and Q-Q Plots

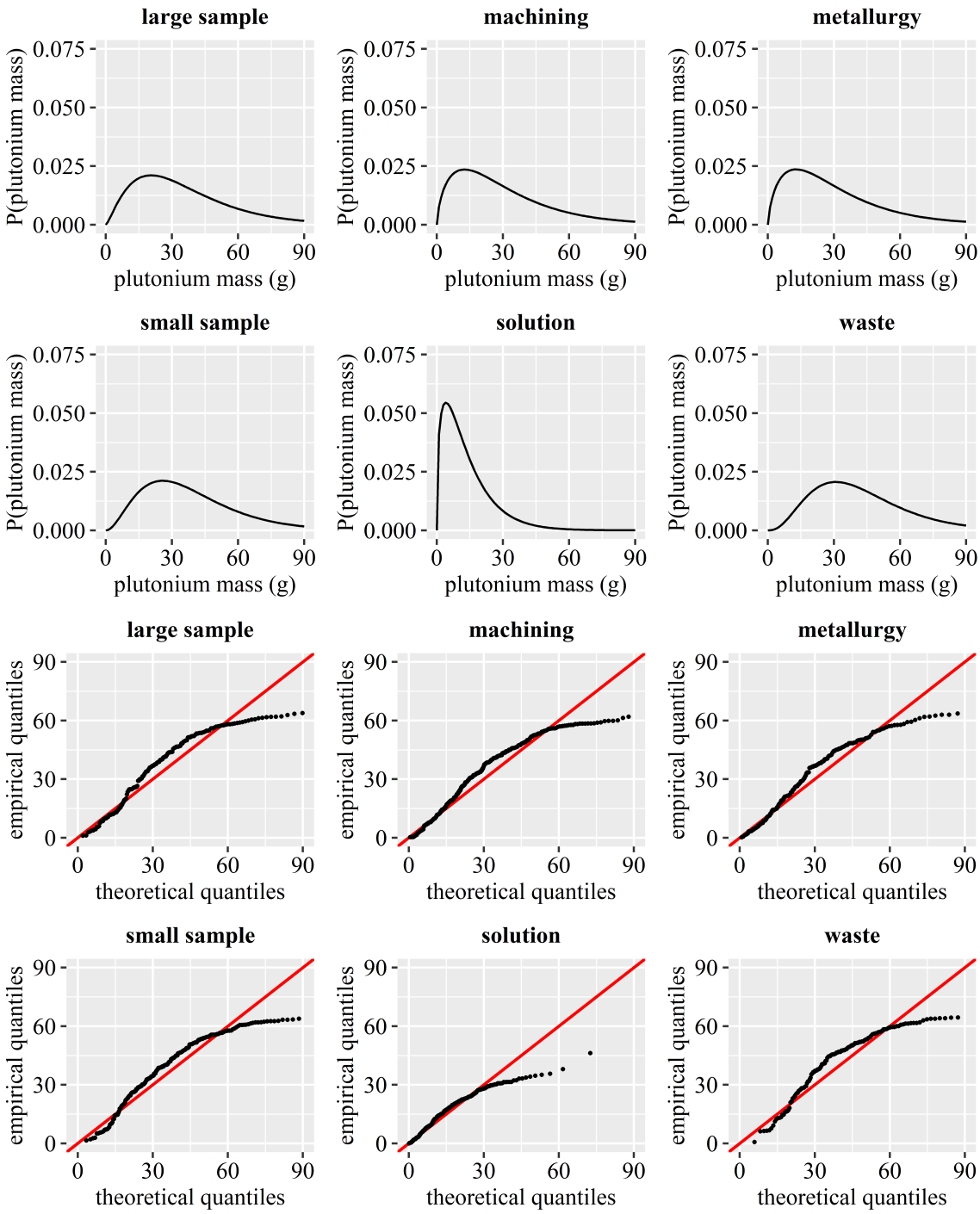

Figure B.1: Truncated gamma distributions and Q-Q plots for the 65-g mass limit

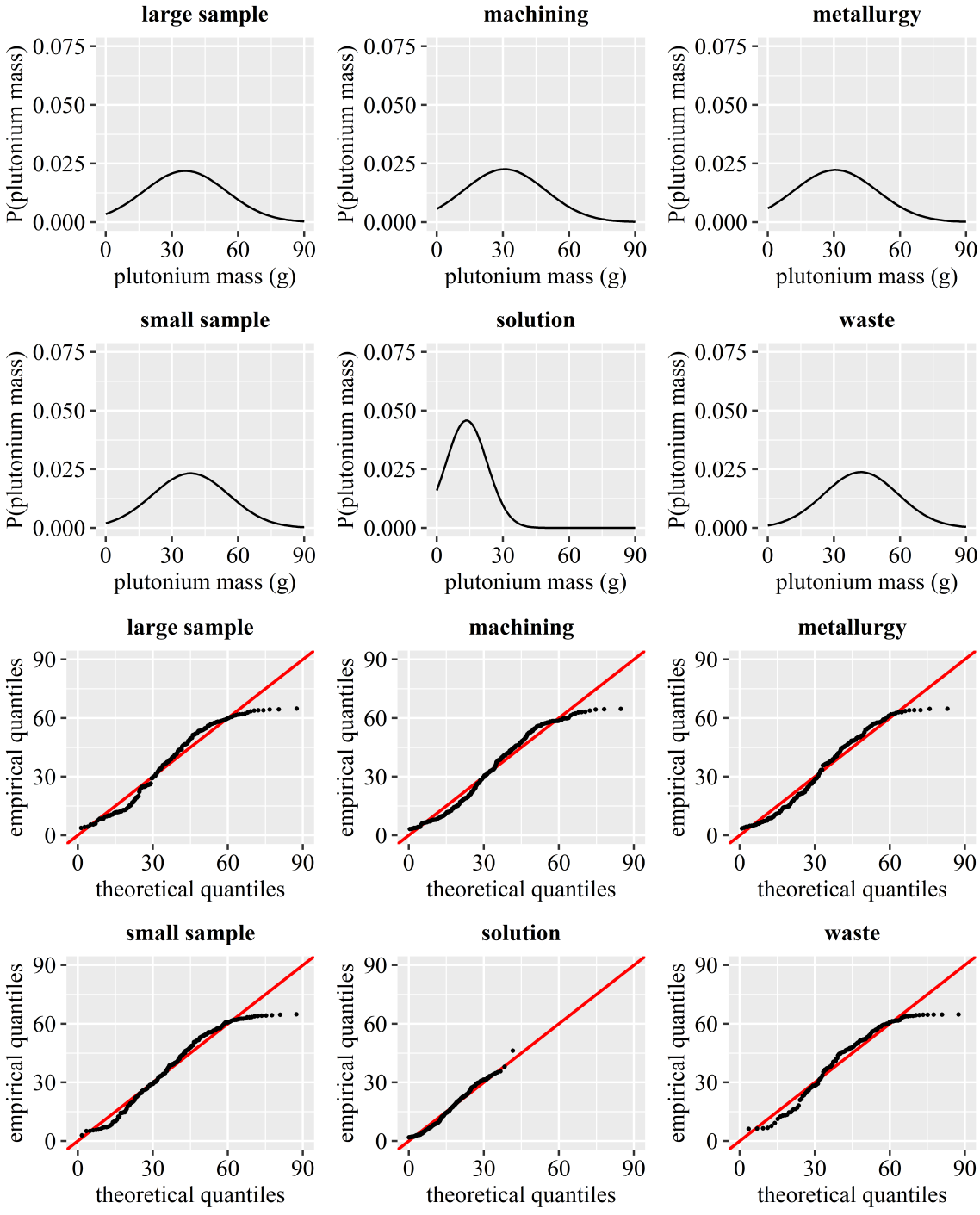

Figure B.2: Truncated normal distributions and Q-Q plots for the 65-g mass limit

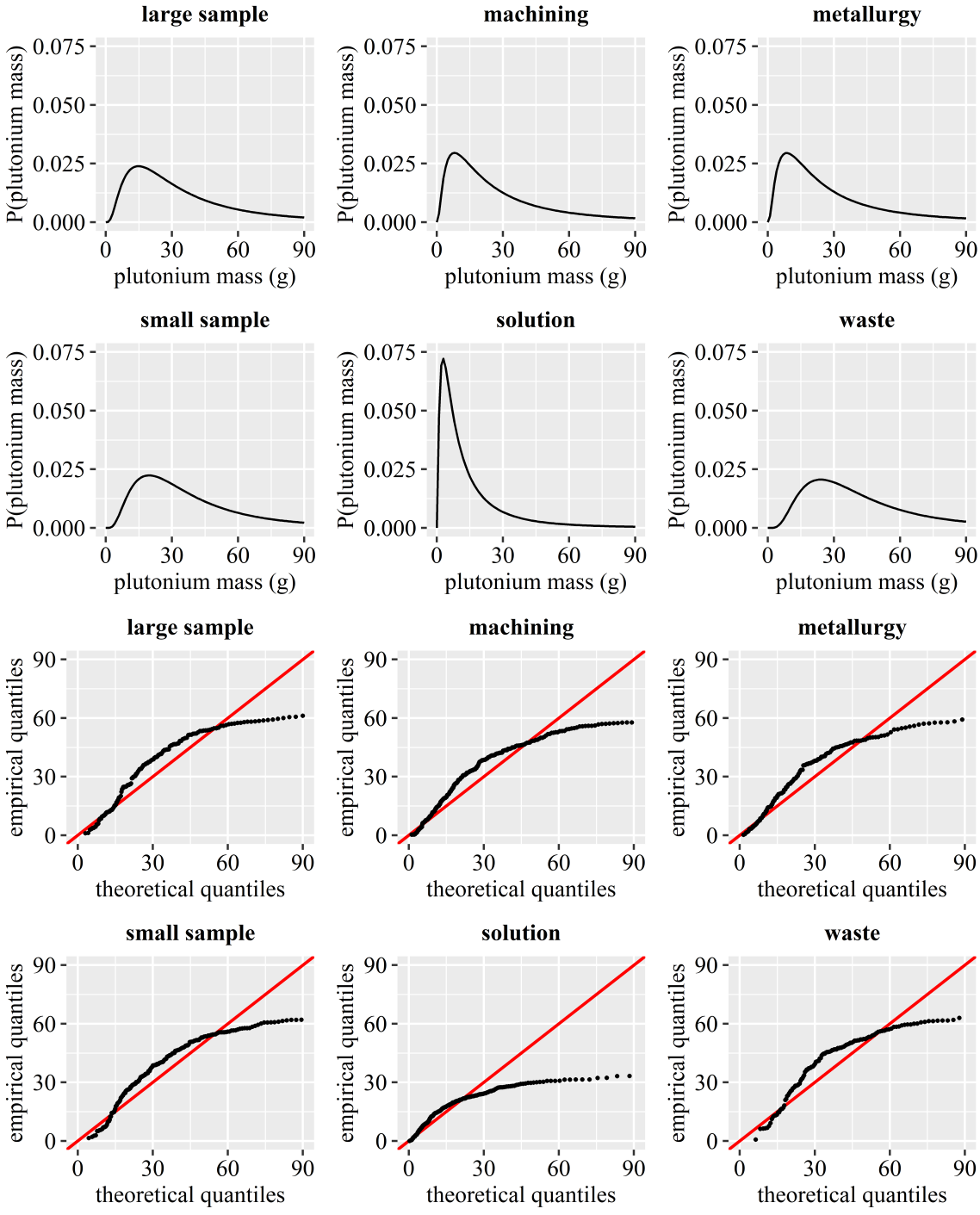

Figure B.3: Truncated log-normal distributions and Q-Q plots for the 65-g mass limit

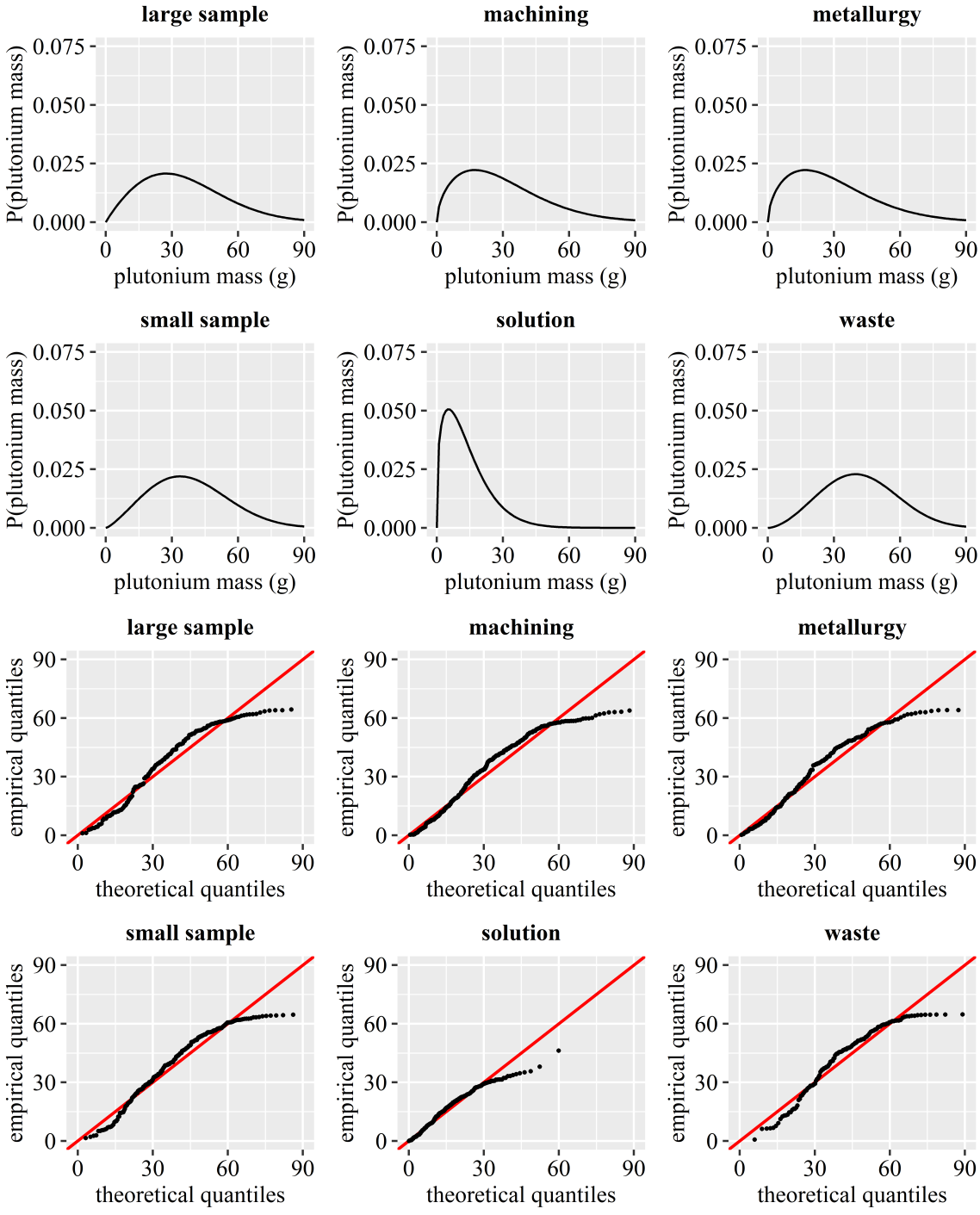

Figure B.4: Truncated Weibull distributions and Q-Q plots for the 65-g mass limit

Appendix C: Last Batch Bias

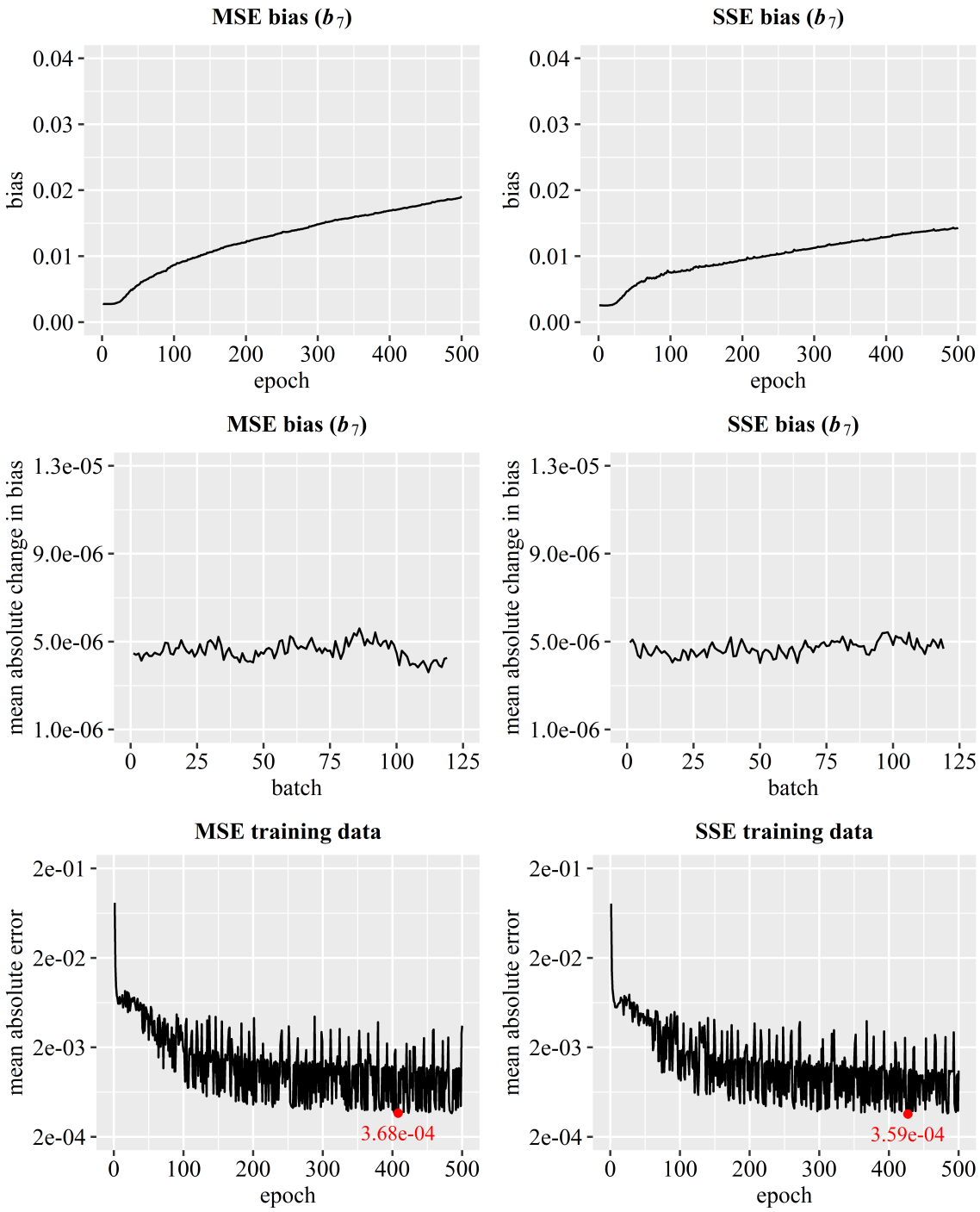

Figure C.1: Neural networks trained on 987,134 samples (a mod n = 4,094)

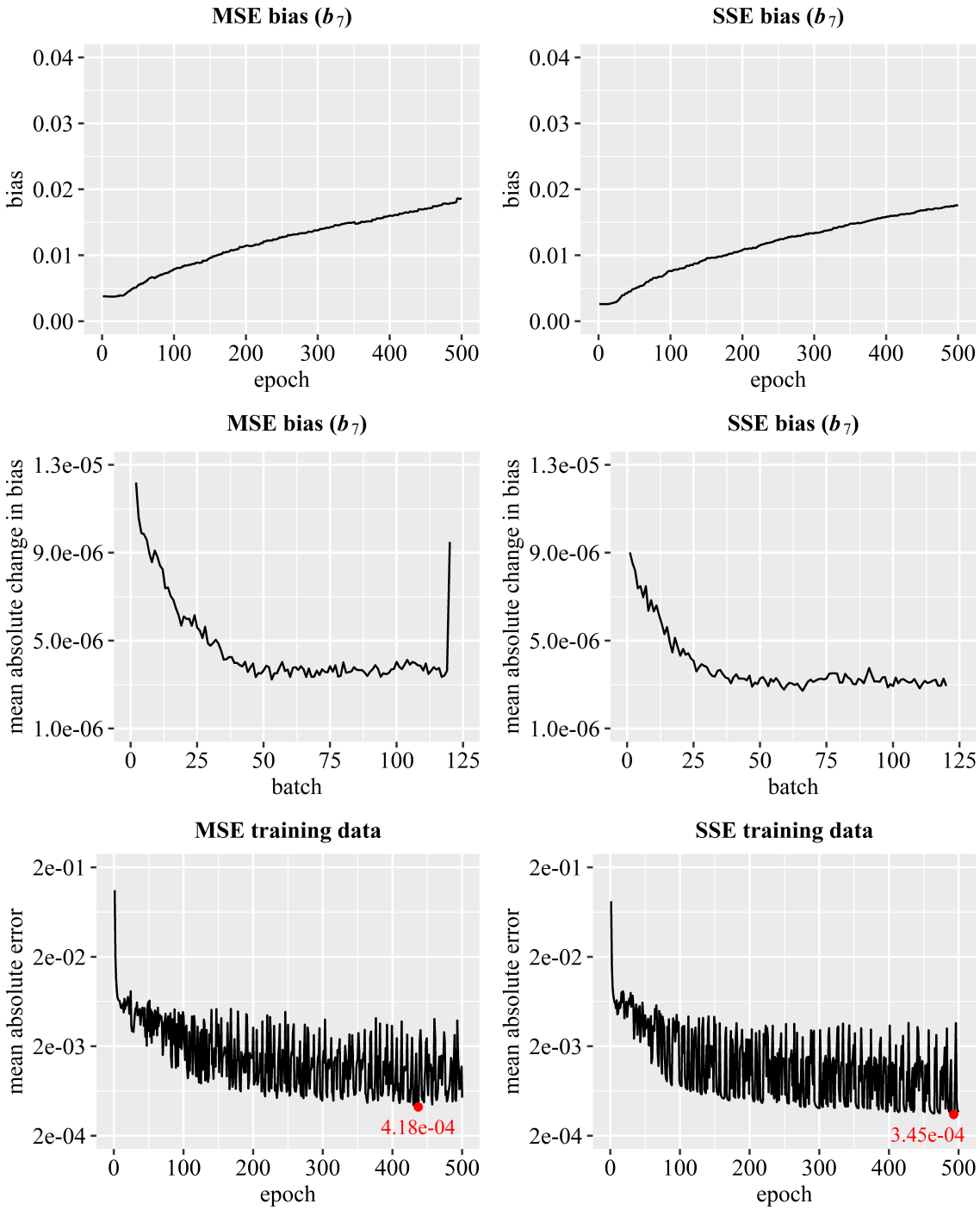

Figure C.2: Neural networks trained on 985,088 samples (a mod n = 2,048)

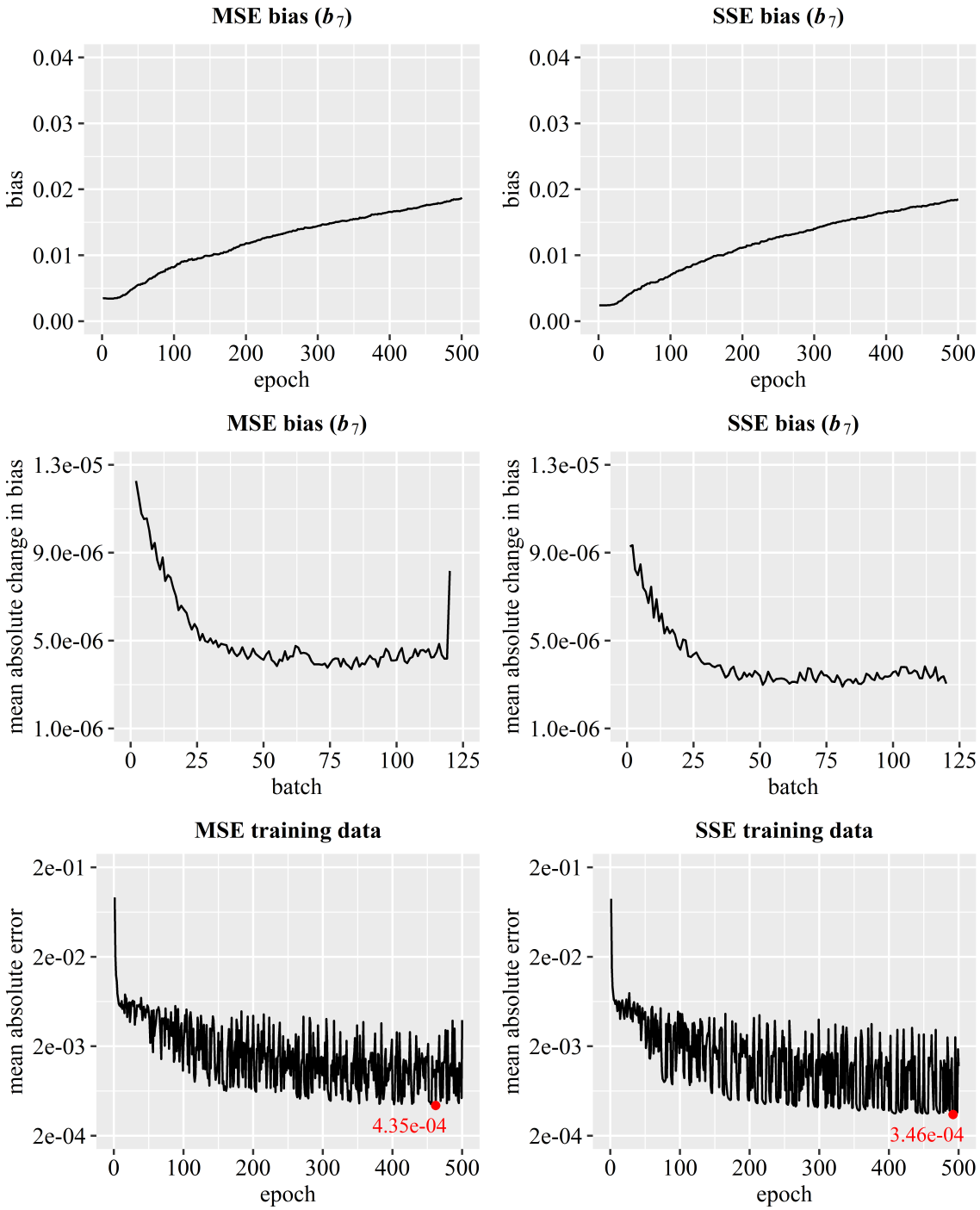

Figure C.3: Neural networks trained on 984,064 samples (a mod n = 1,024)

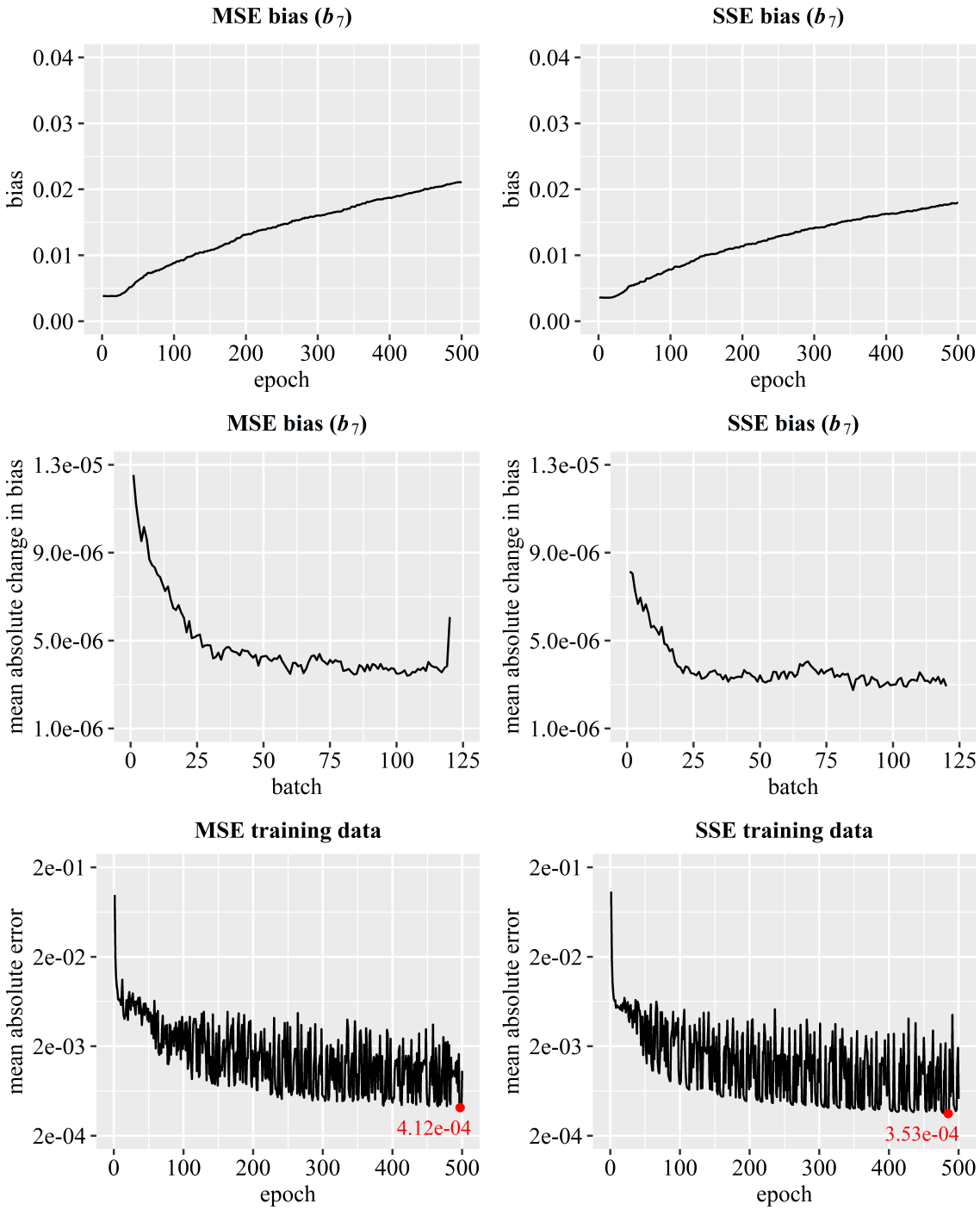

Figure C.4: Neural networks trained on 983,552 samples (a mod n = 512)

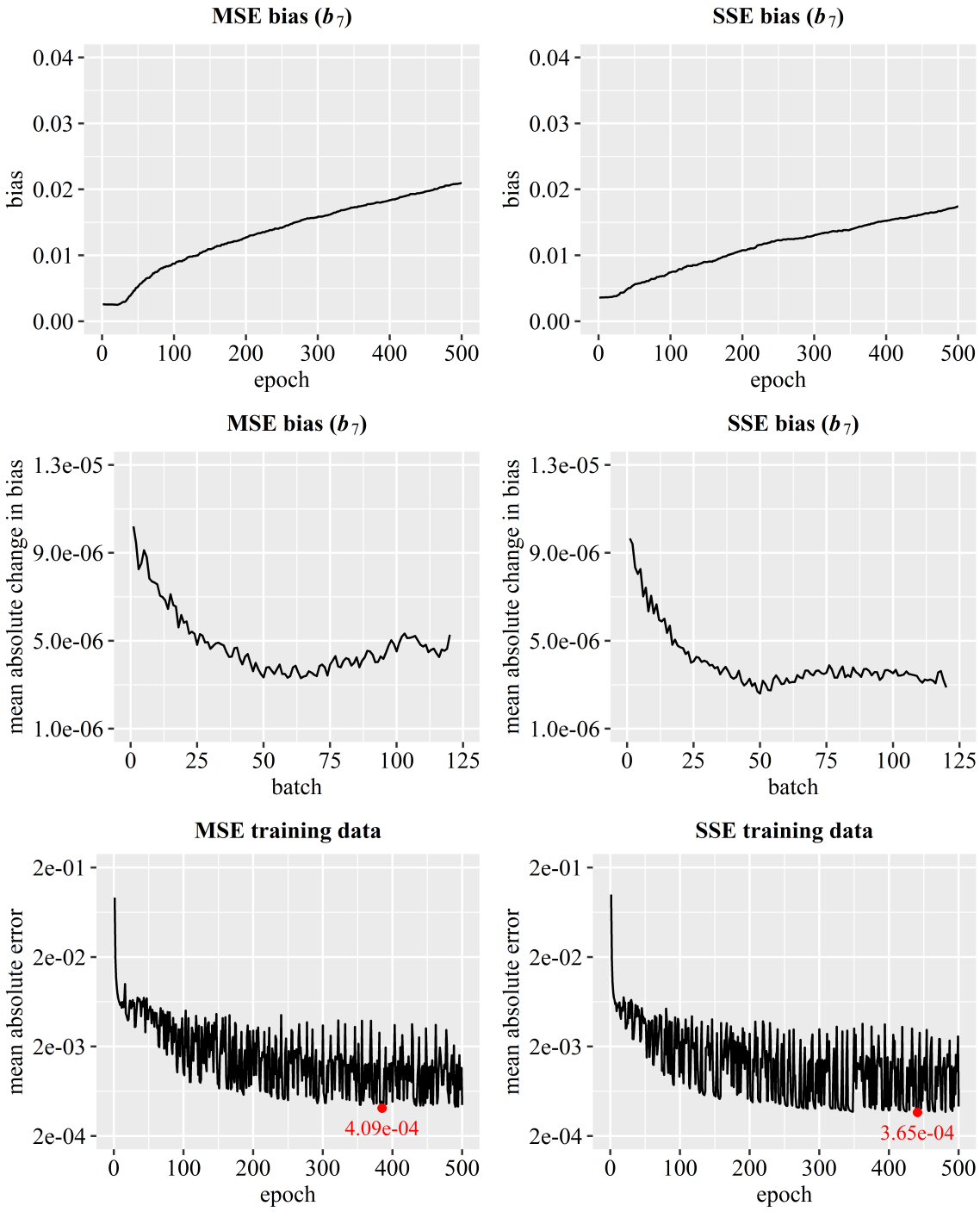

Figure C.5: Neural networks trained on 983,296 samples (a mod $n = 256$)

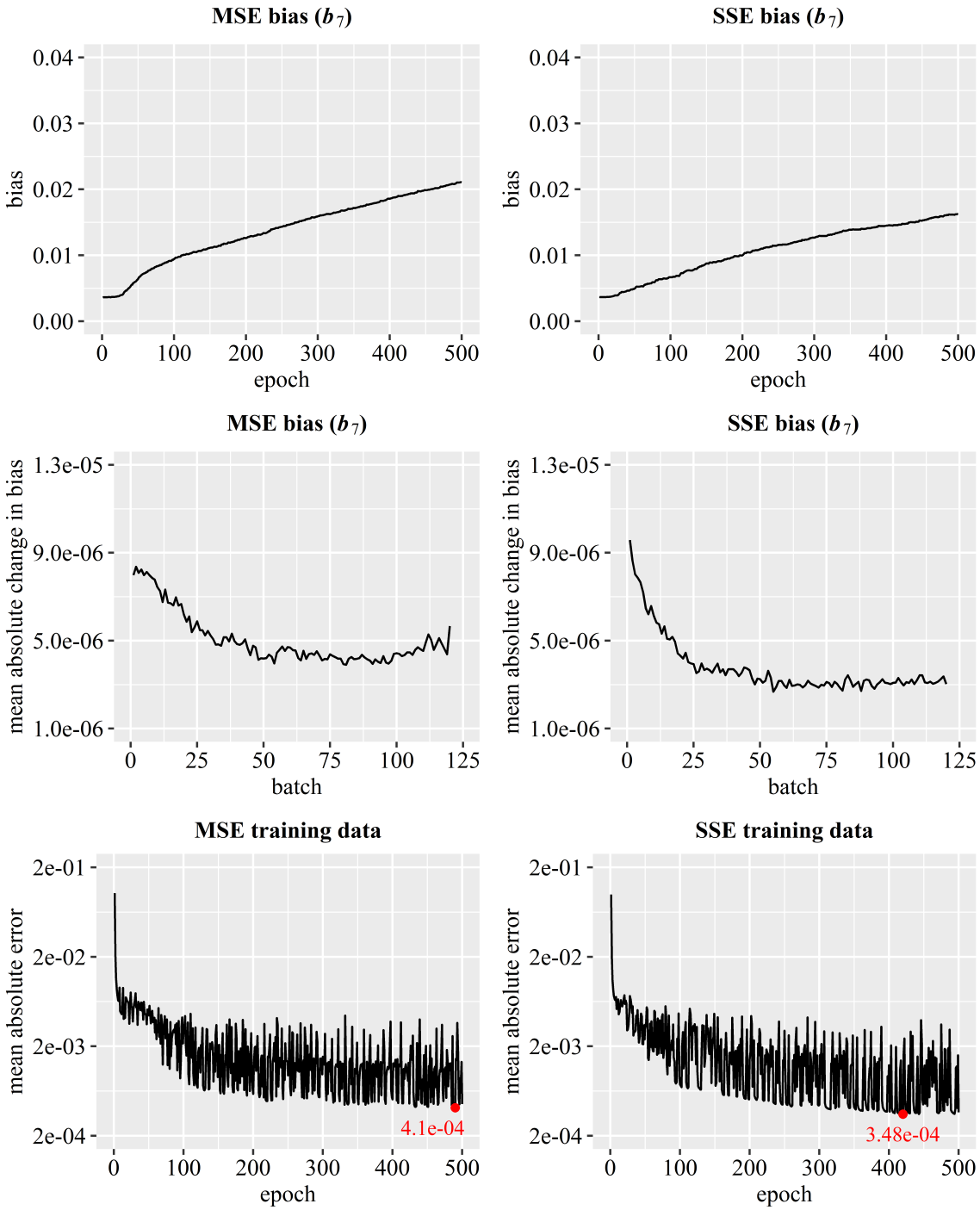

Figure C.6: Neural networks trained on 983,168 samples (a mod n = 128)

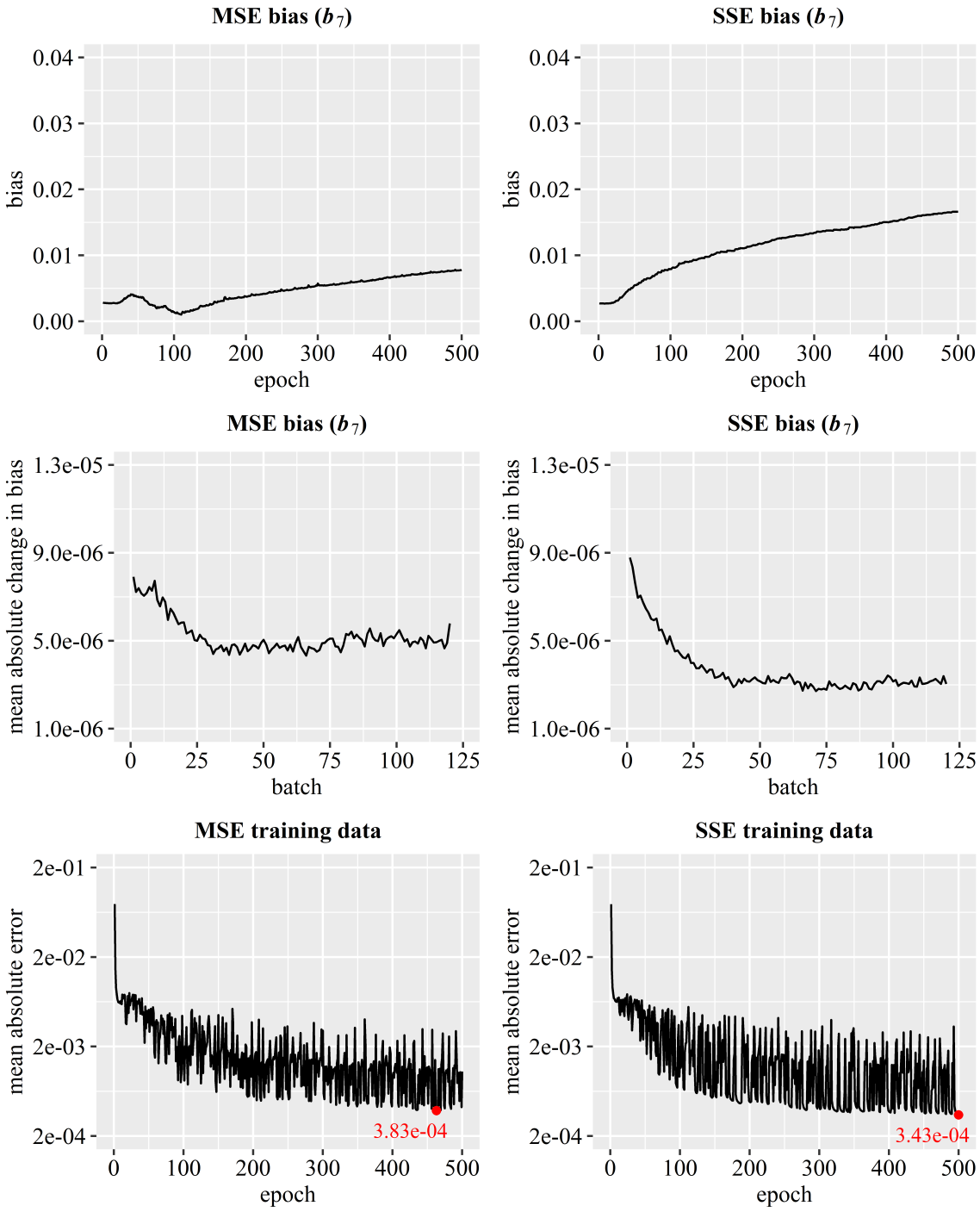

Figure C.7: Neural networks trained on 983,104 samples (a mod n = 64)

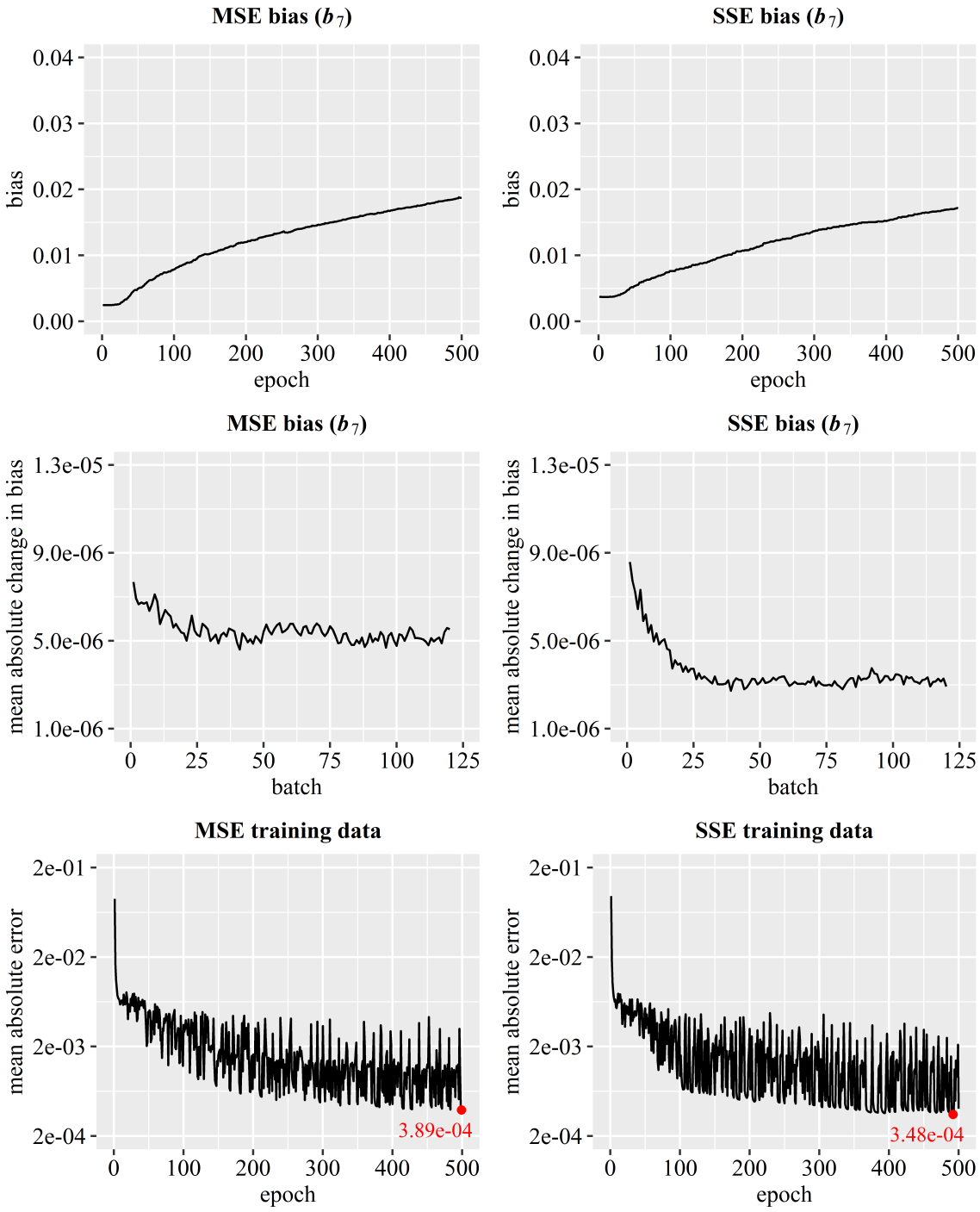

Figure C.8: Neural networks trained on 983,072 samples (a mod n = 32)

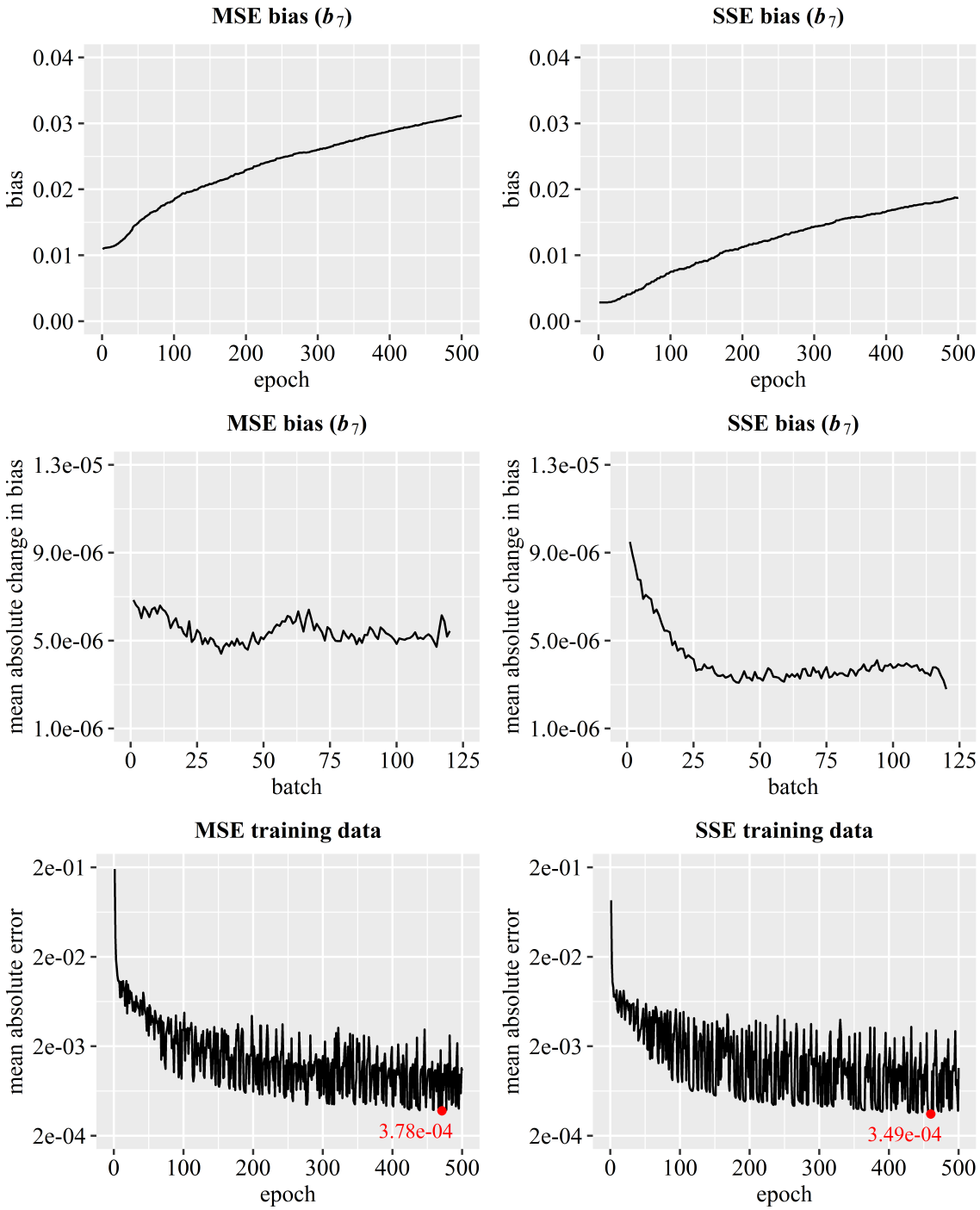

Figure C.9: Neural networks trained on 983,056 samples (a mod n = 16)

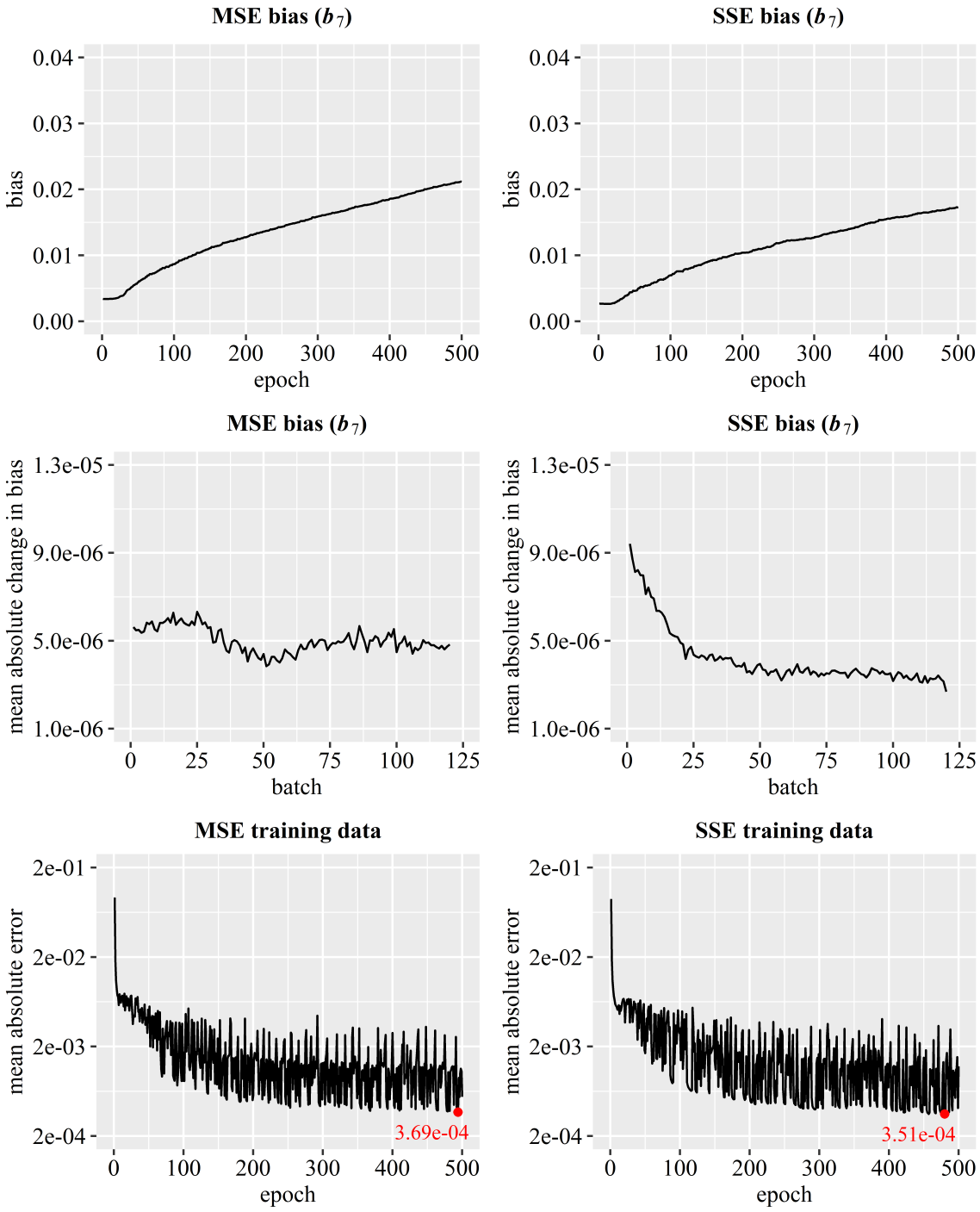

Figure C.10: Neural networks trained on 983,048 samples (a mod $n = 8$)

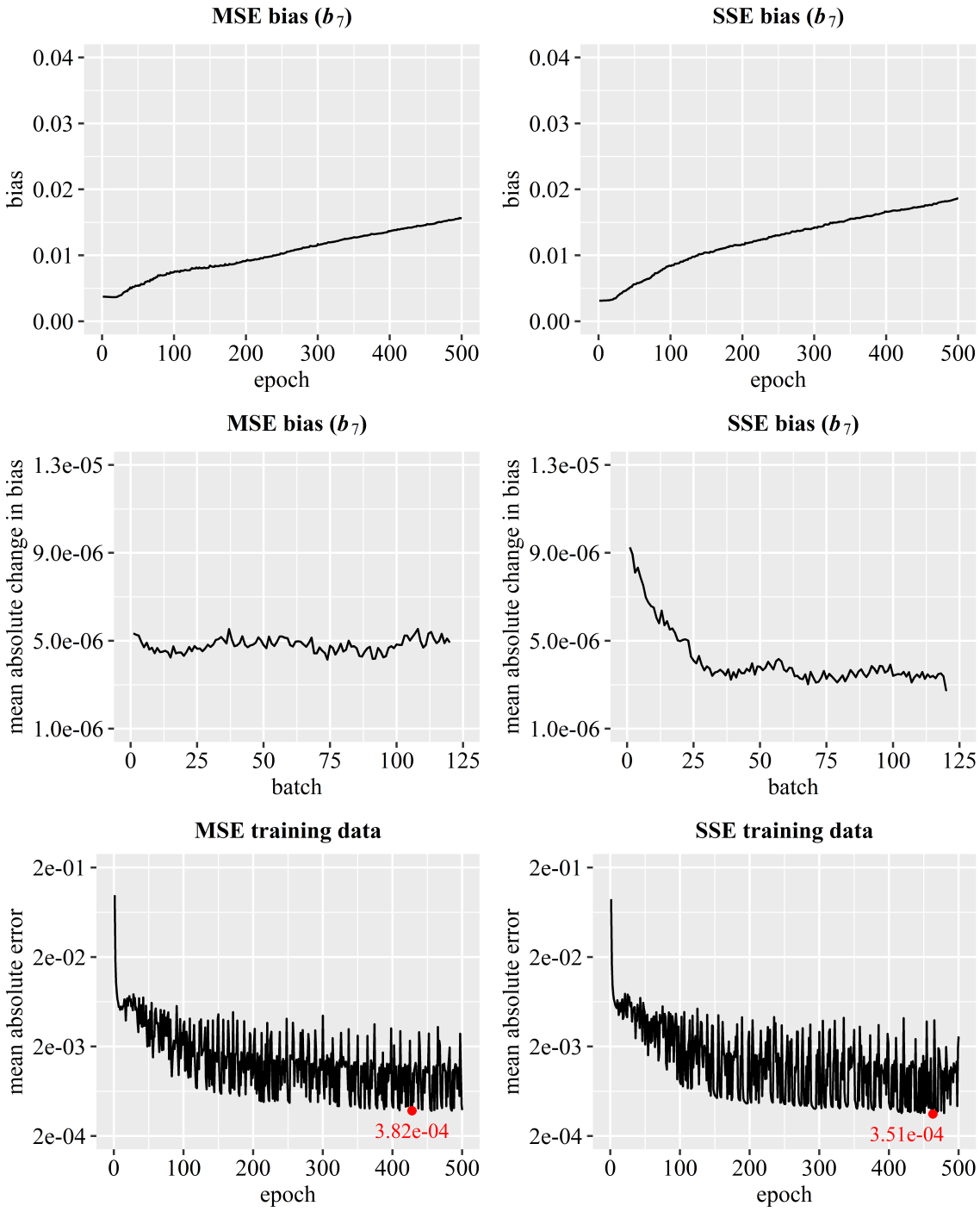

Figure C.11: Neural networks trained on 983,044 samples (a mod $n = 4$)

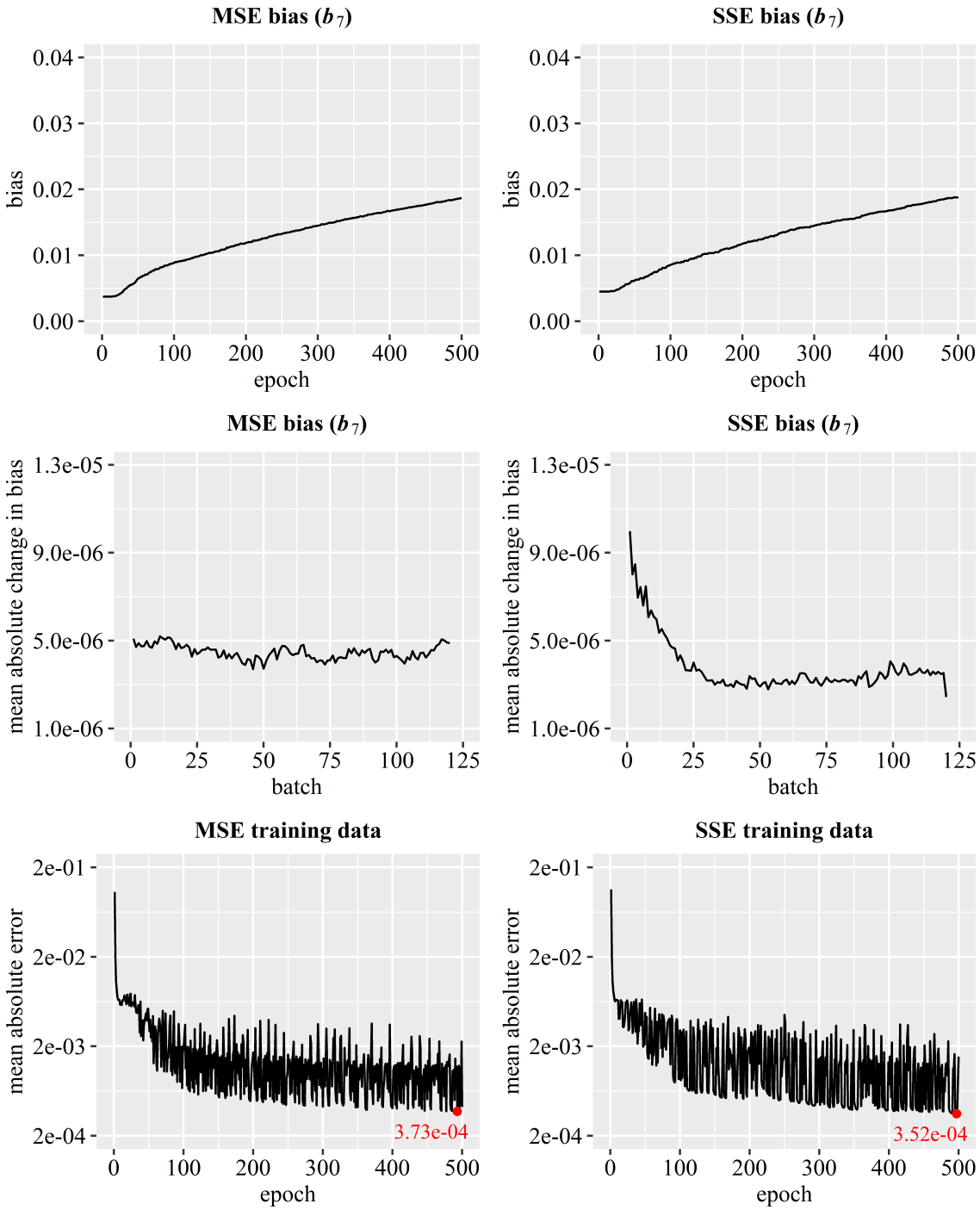

Figure C.12: Neural networks trained on 983,042 samples (a mod $n = 2$)

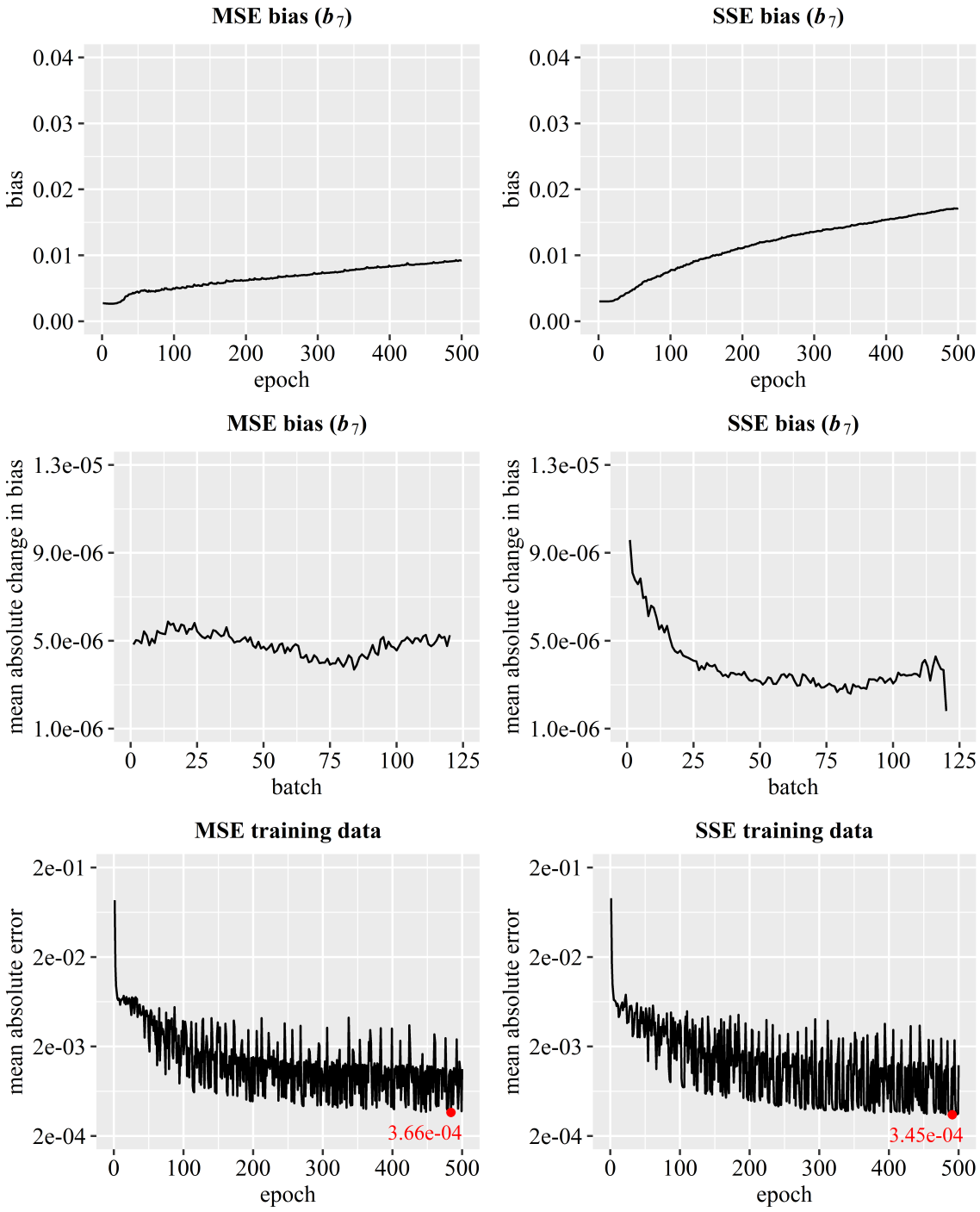

Figure C.13: Neural networks trained on 983,041 samples (a mod n = 1)

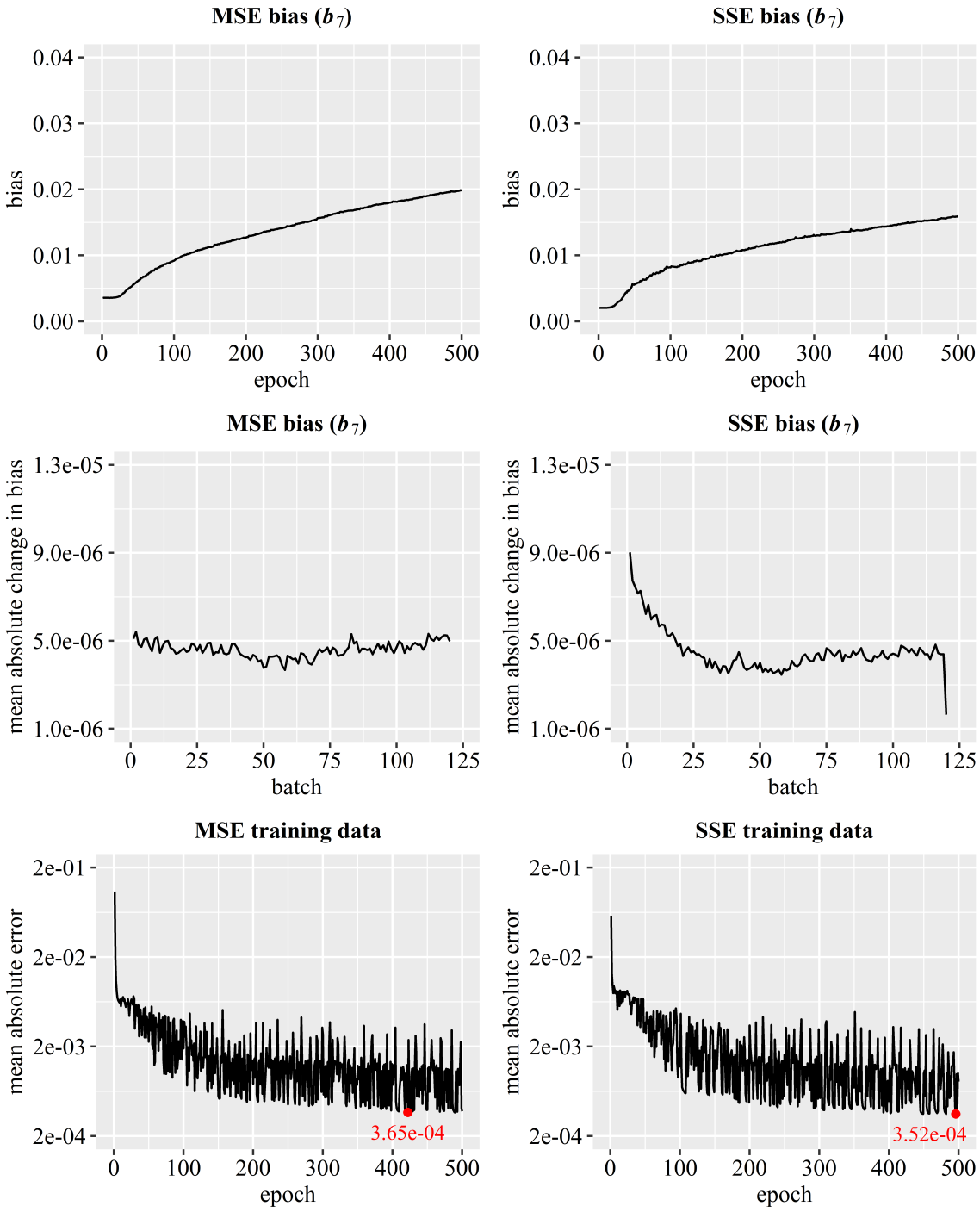

Figure C.14: Neural networks trained on 983,040 samples ($a \bmod n = 0$)

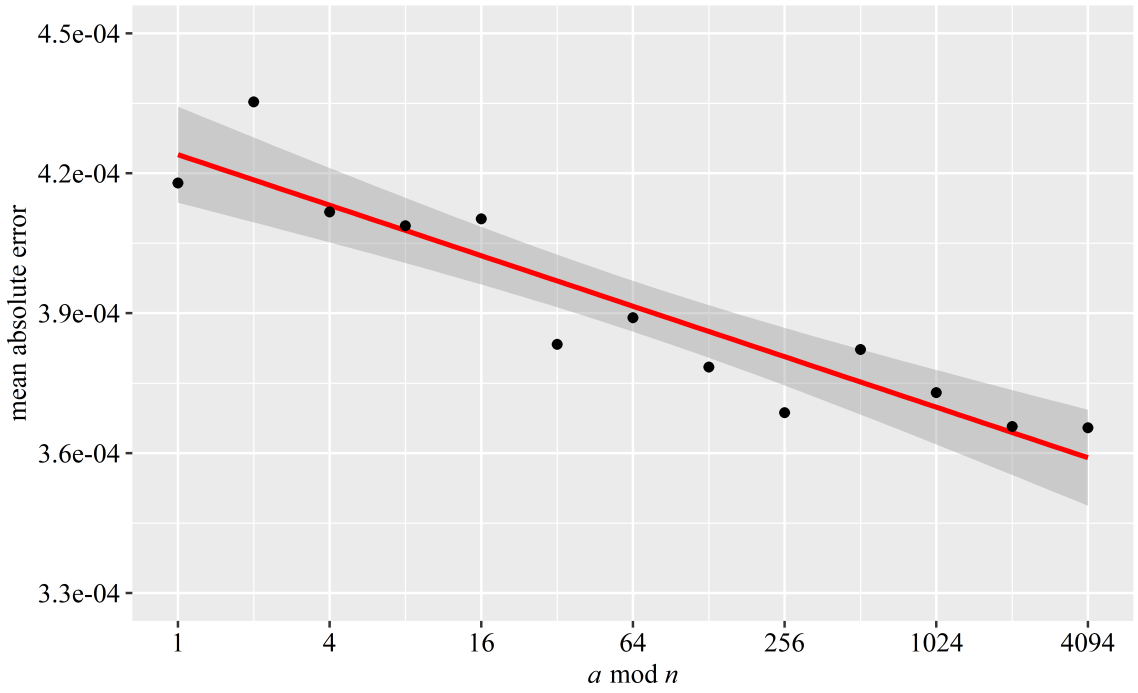

Figure C.15: Neural networks trained for 500 epochs using the MSE loss function

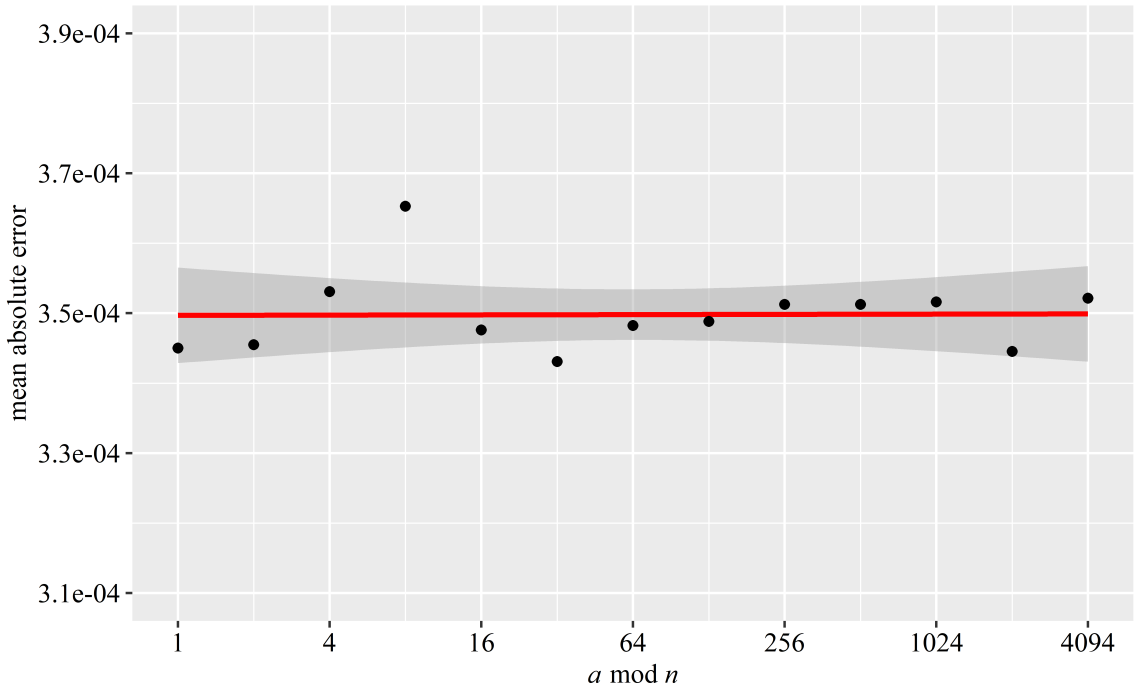

Figure C.16: Neural networks trained for 500 epochs using the SSE loss function

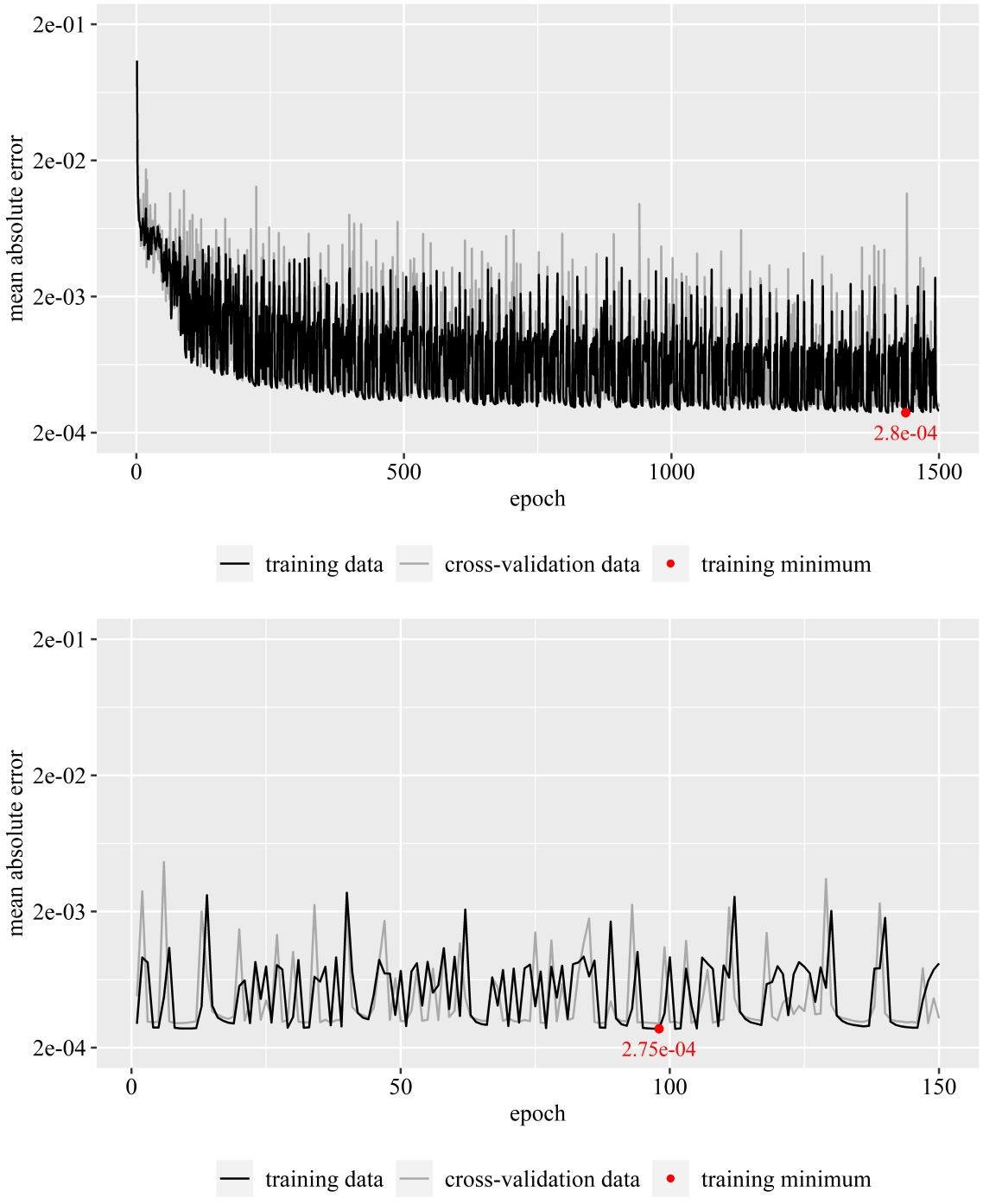

Figure D.1: Neural network trained using the SSE loss function (1 of 20)

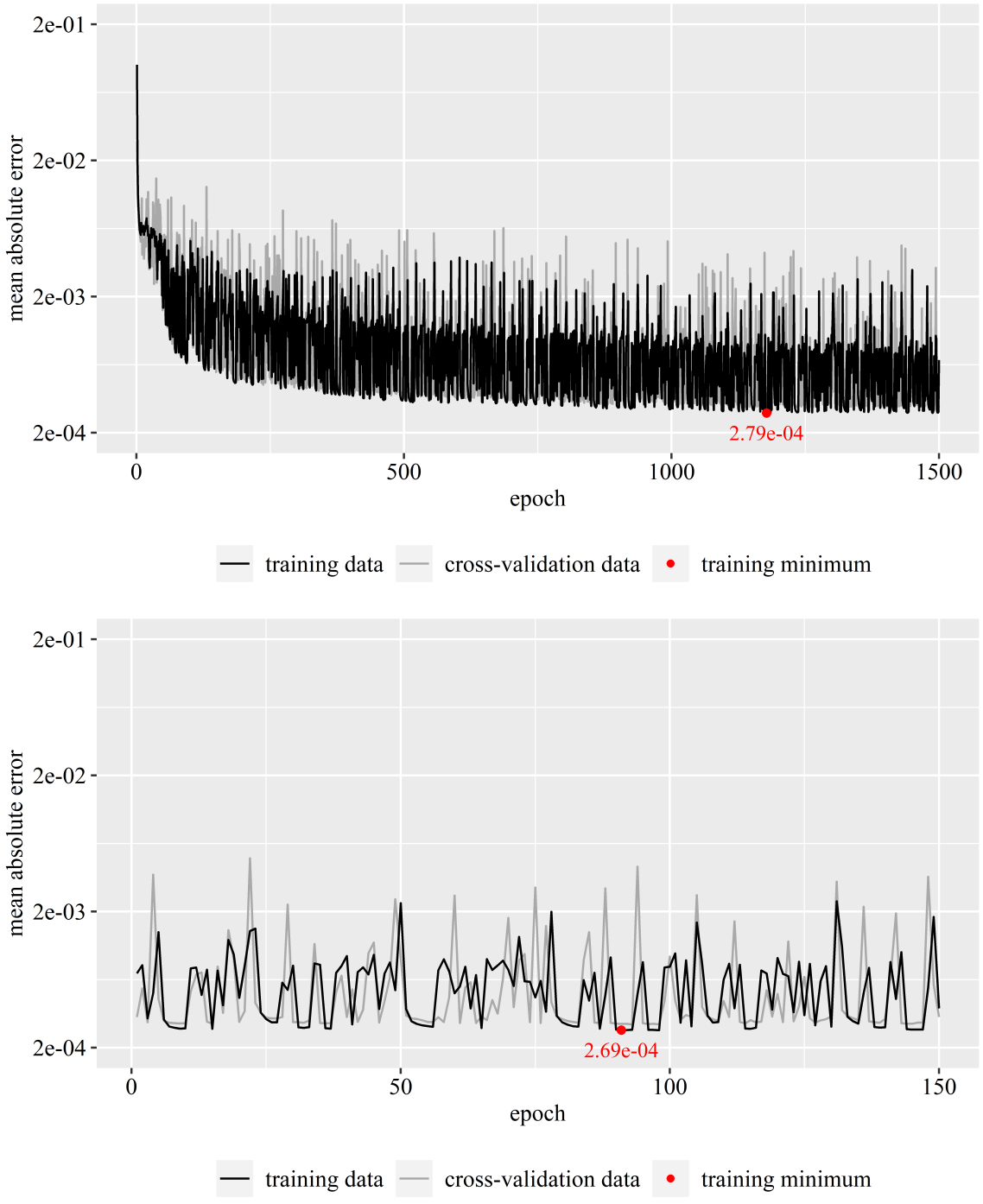

Figure D.2: Neural network trained using the SSE loss function (2 of 20)

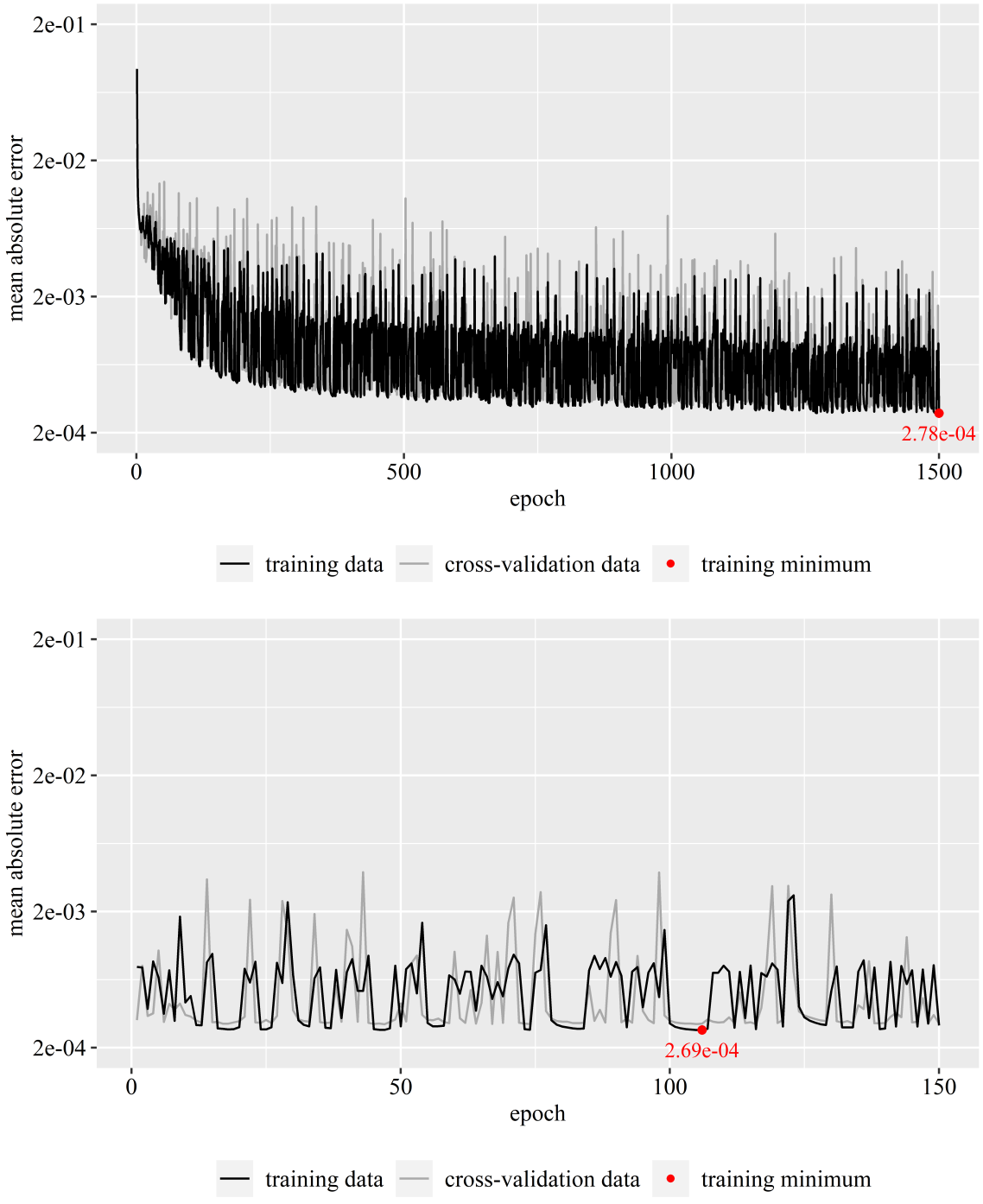

Figure D.3: Neural network trained using the SSE loss function (3 of 20)

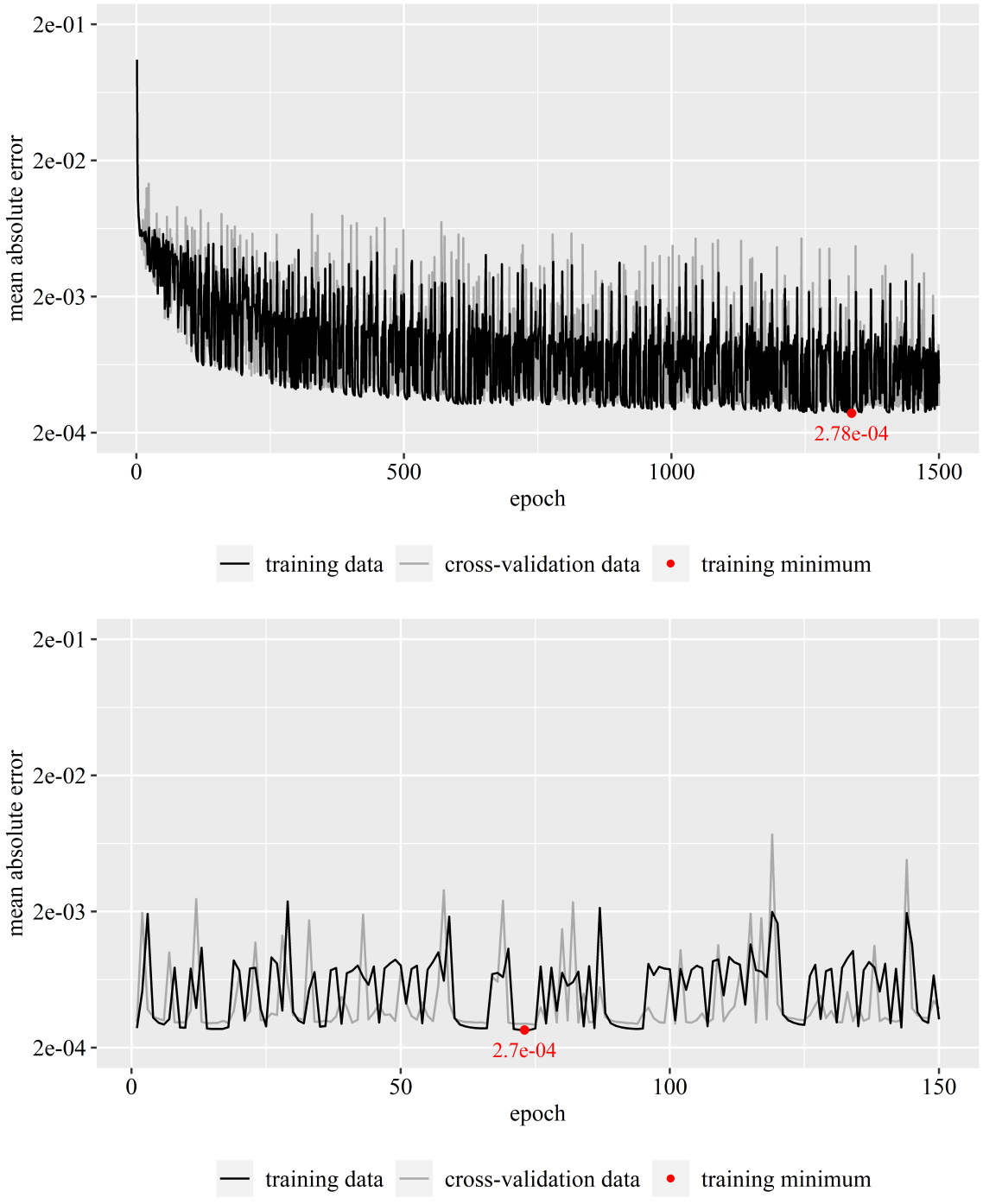

Figure D.4: Neural network trained using the SSE loss function (4 of 20)

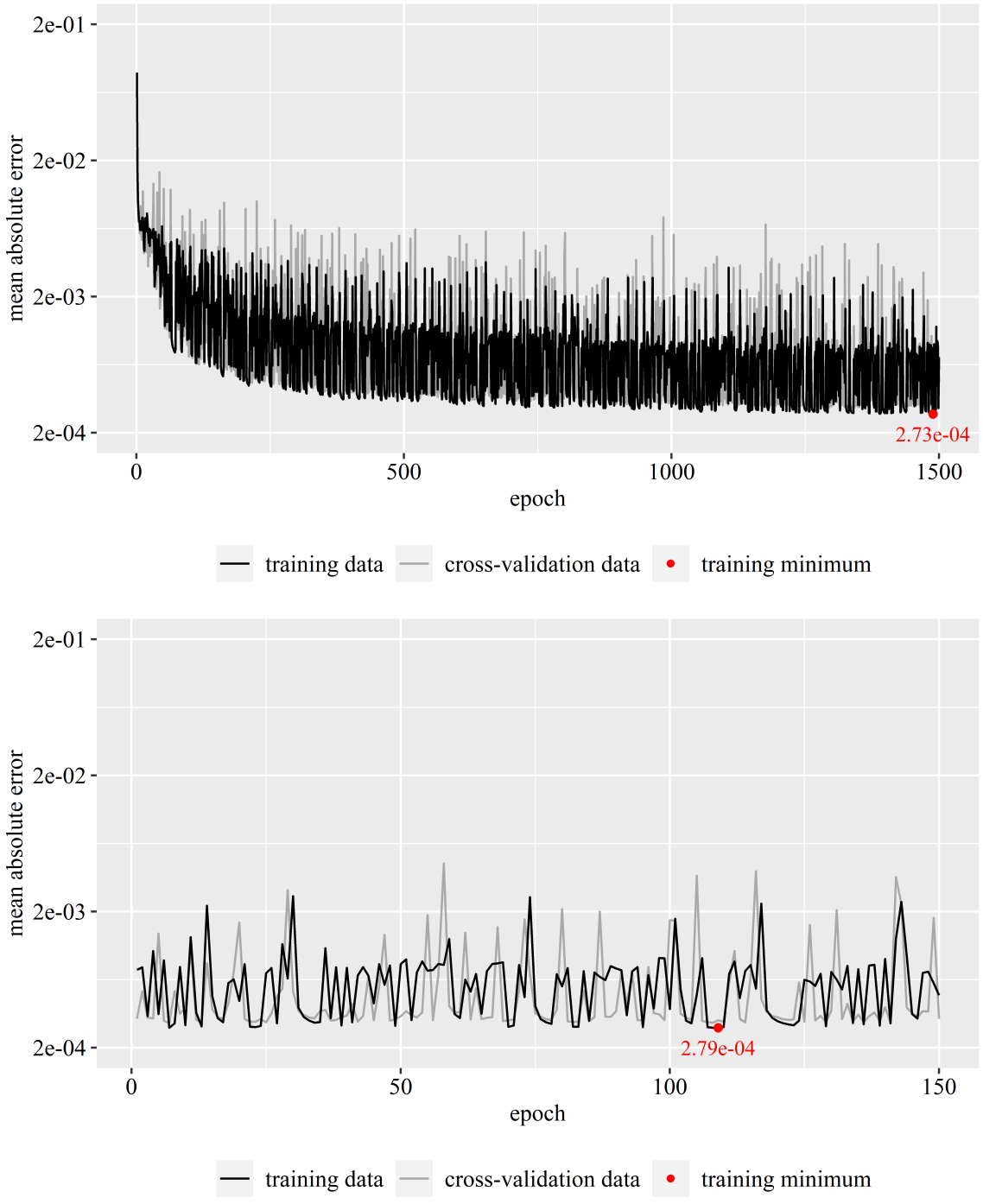

Figure D.5: Neural network trained using the SSE loss function (5 of 20)

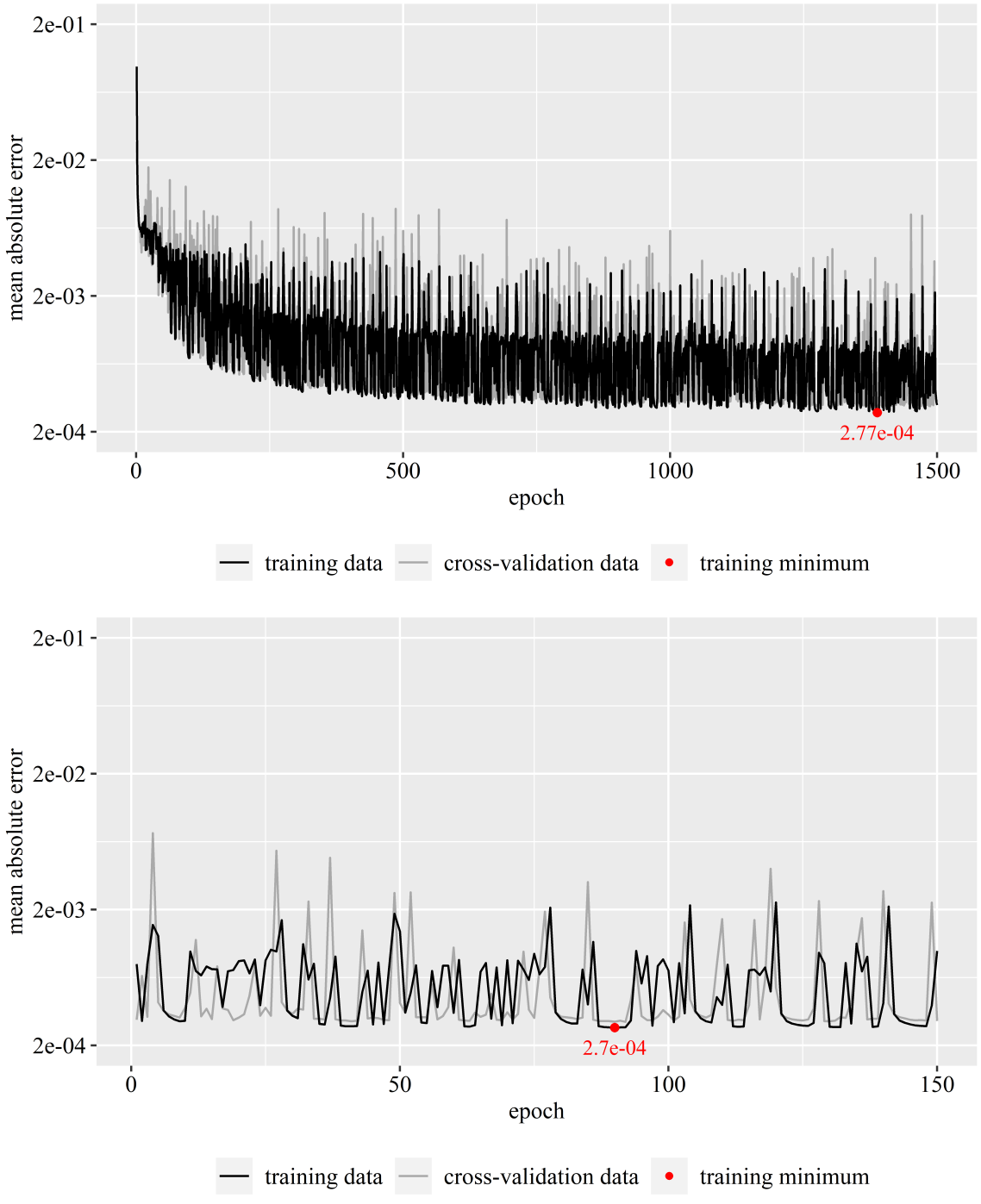

Figure D.6: Neural network trained using the SSE loss function (6 of 20)

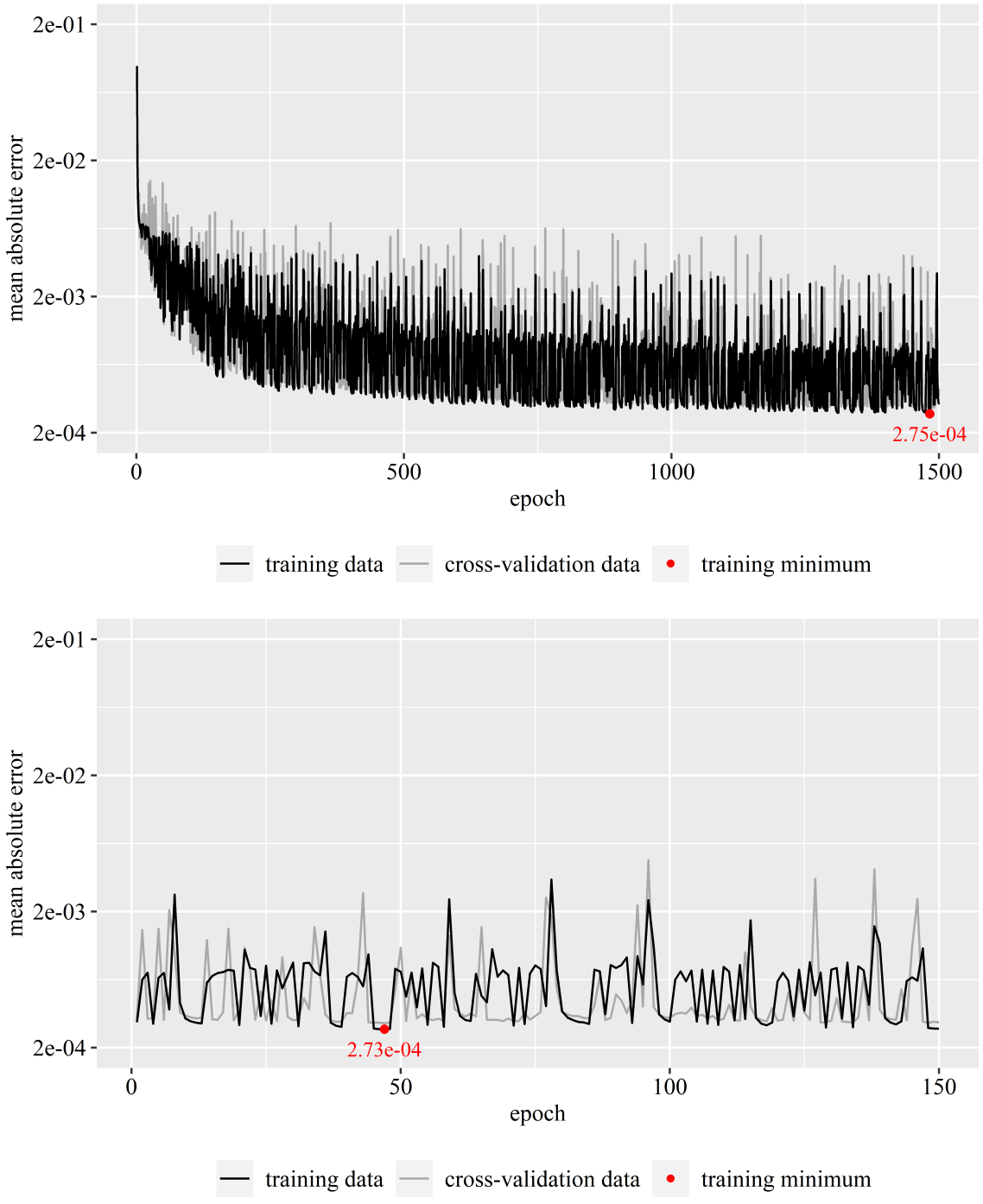

Figure D.7: Neural network trained using the SSE loss function (7 of 20)

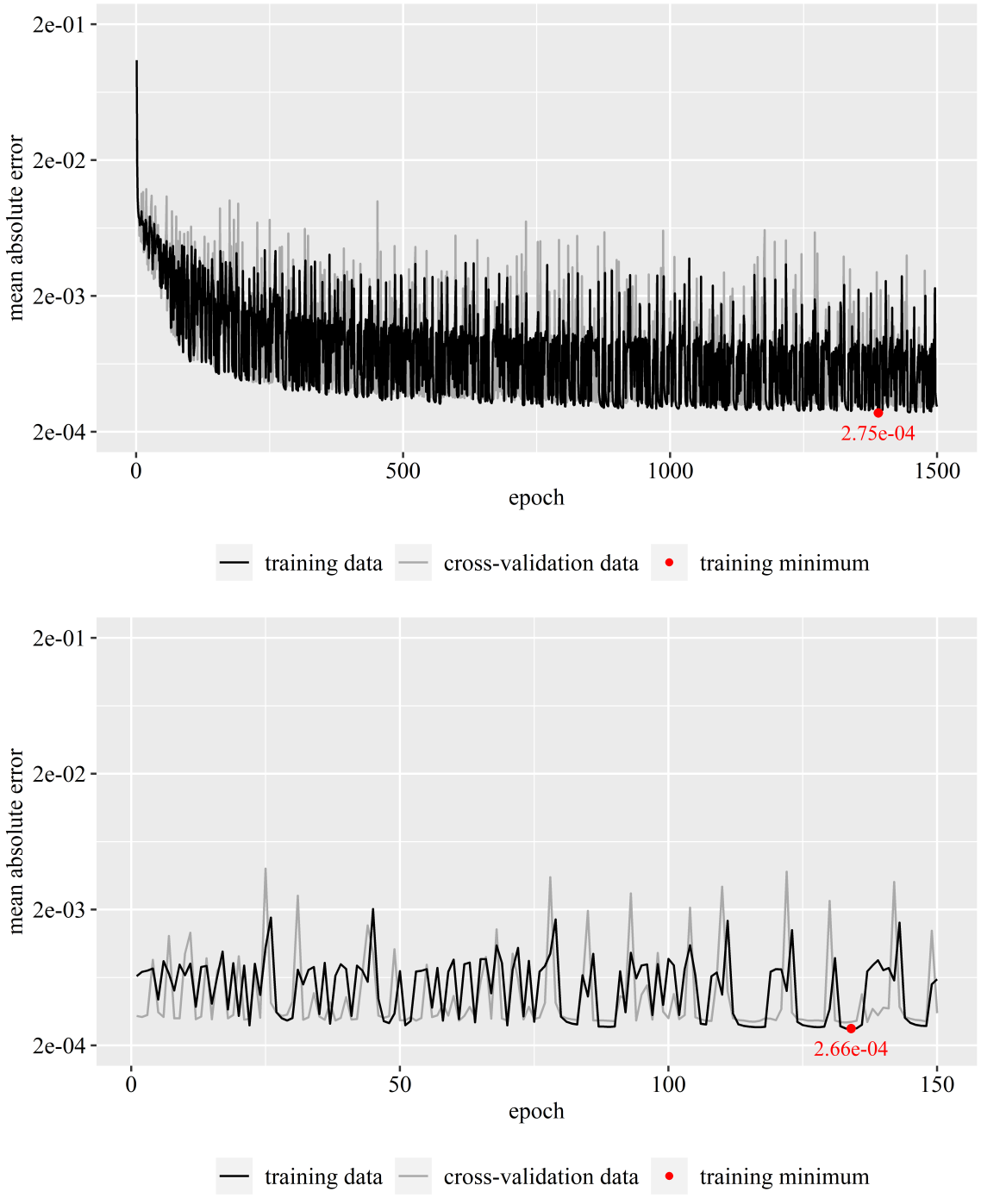

Figure D.8: Neural network trained using the SSE loss function (8 of 20)

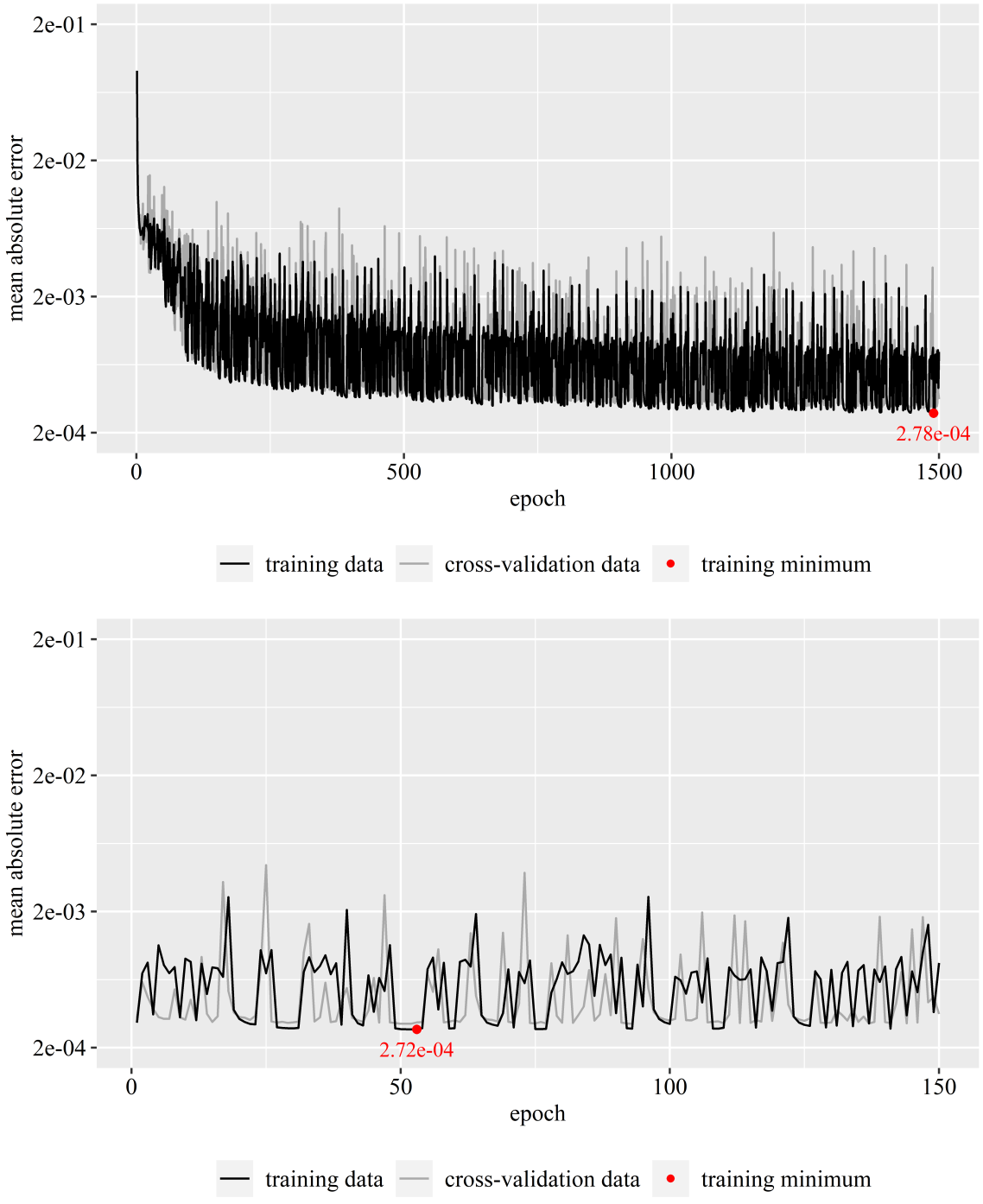

Figure D.9: Neural network trained using the SSE loss function (9 of 20)

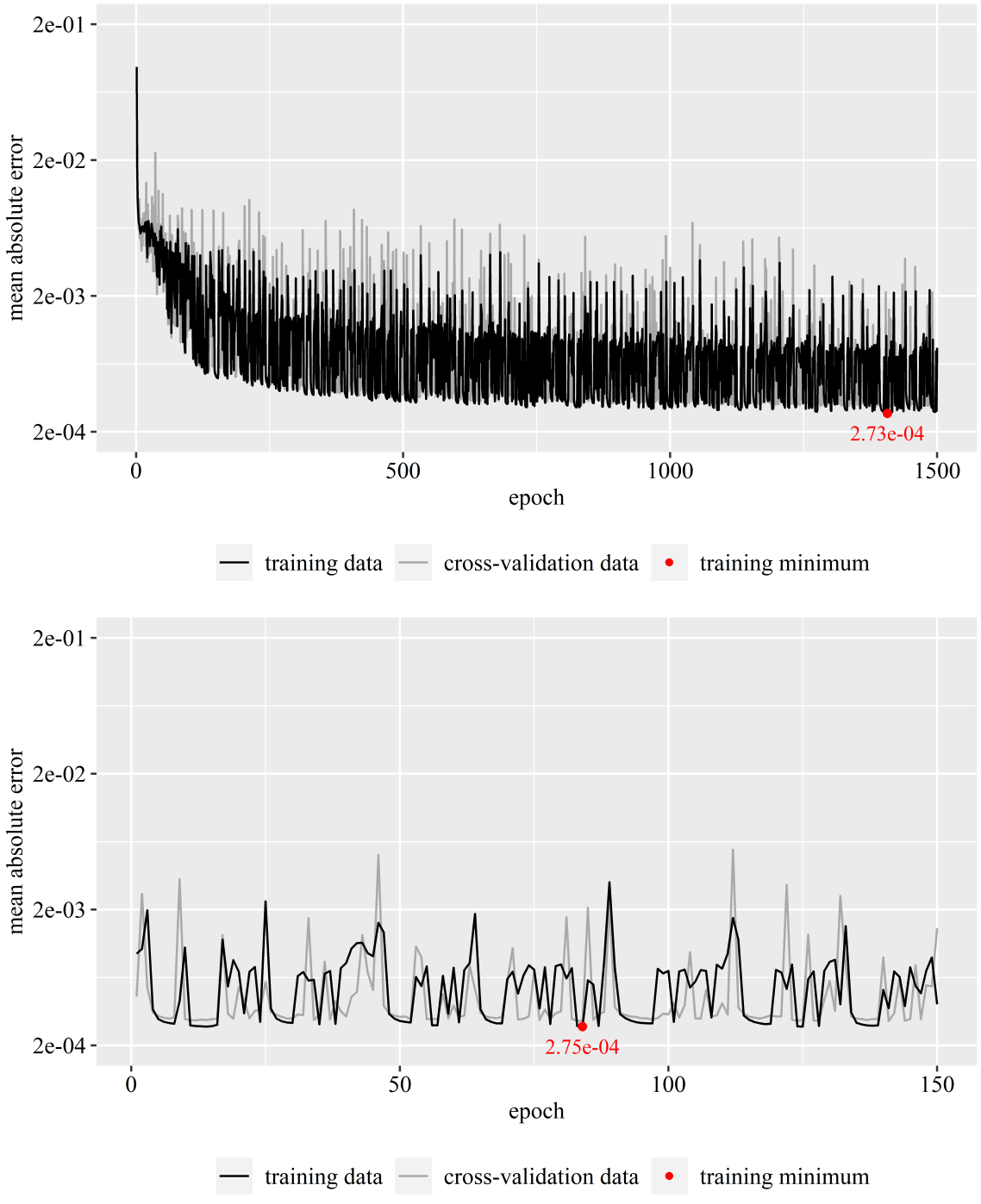

Figure D.10: Neural network trained using the SSE loss function (10 of 20)

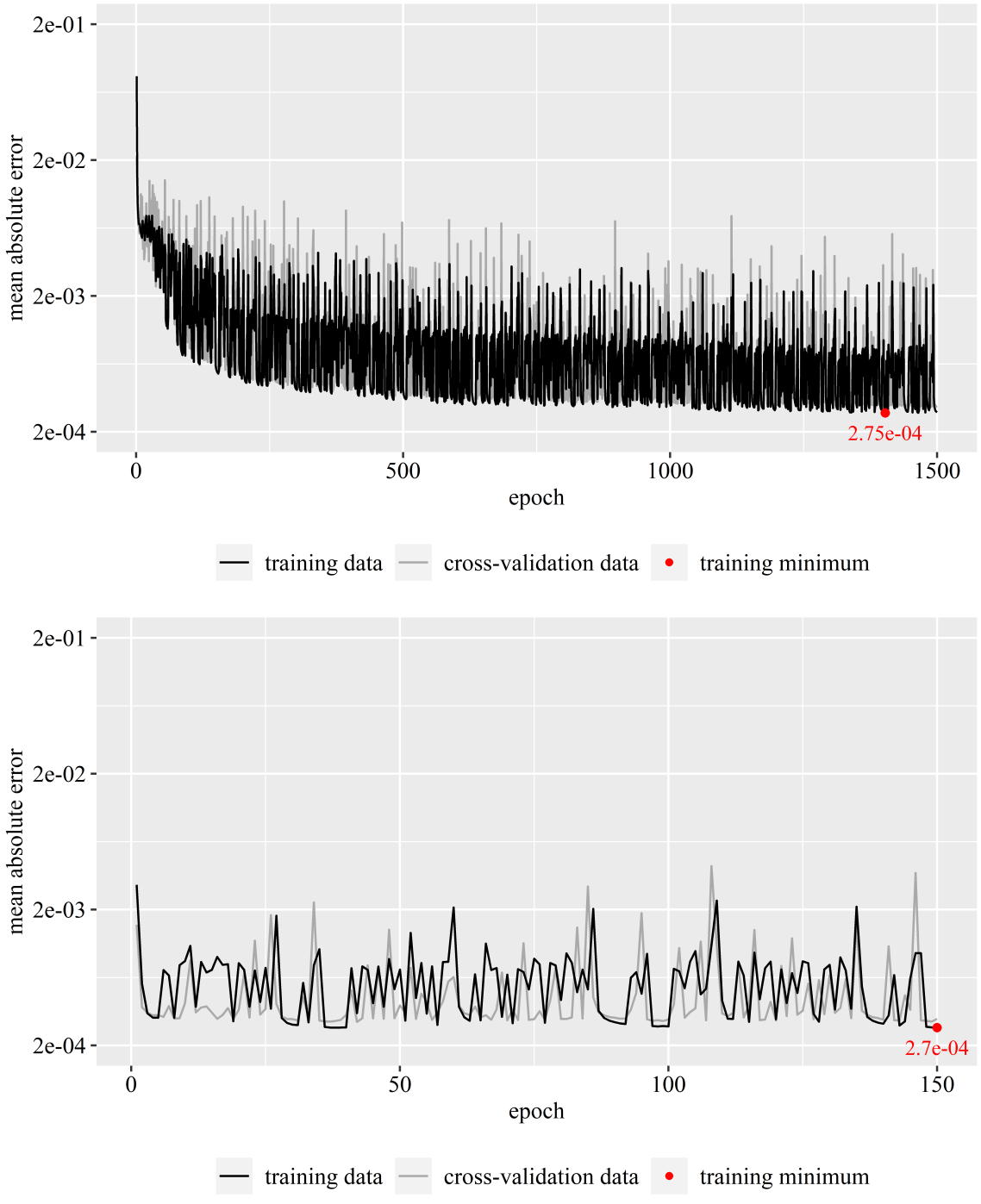

Figure D.11: Neural network trained using the SSE loss function (11 of 20)

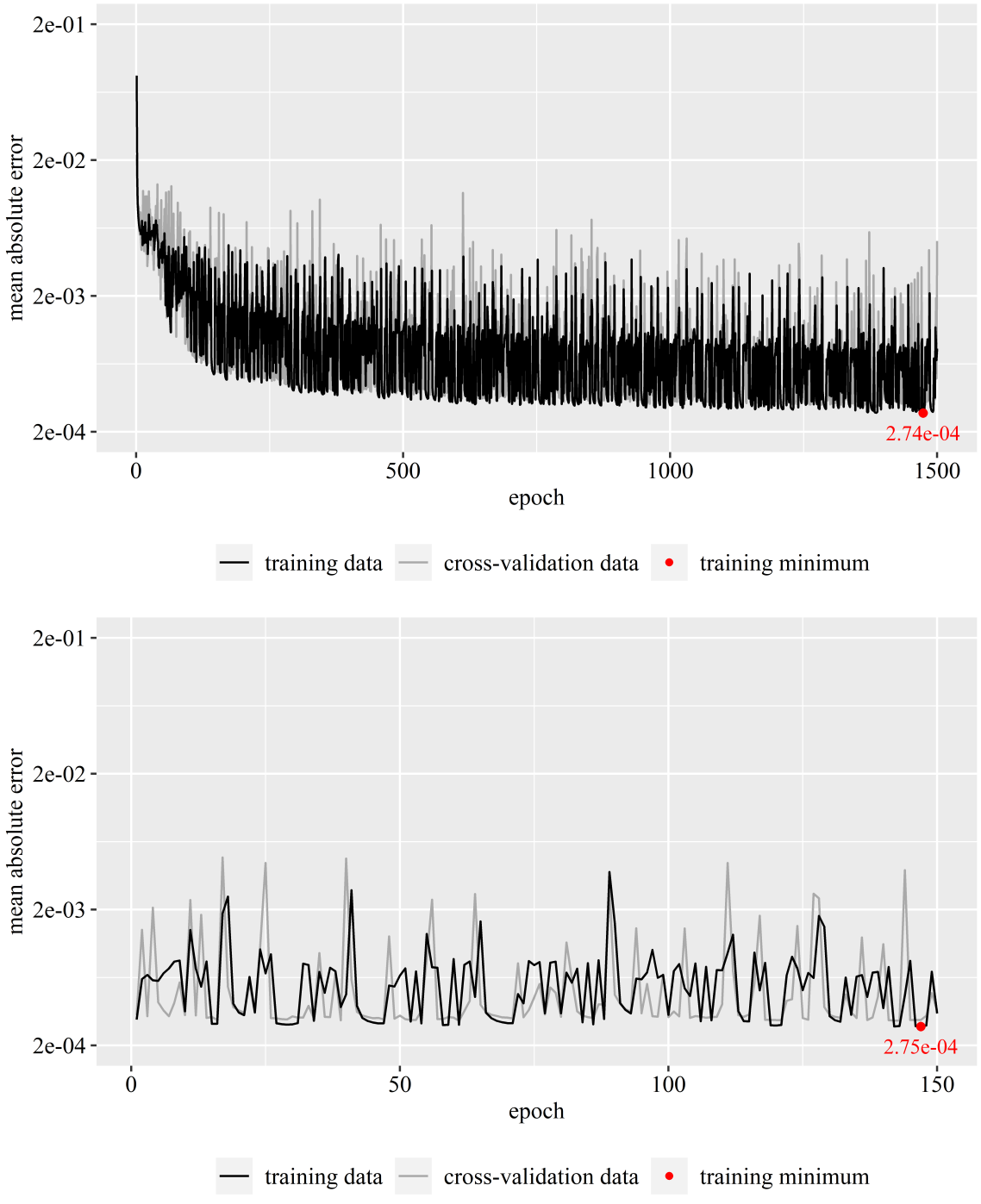

Figure D.12: Neural network trained using the SSE loss function (12 of 20)

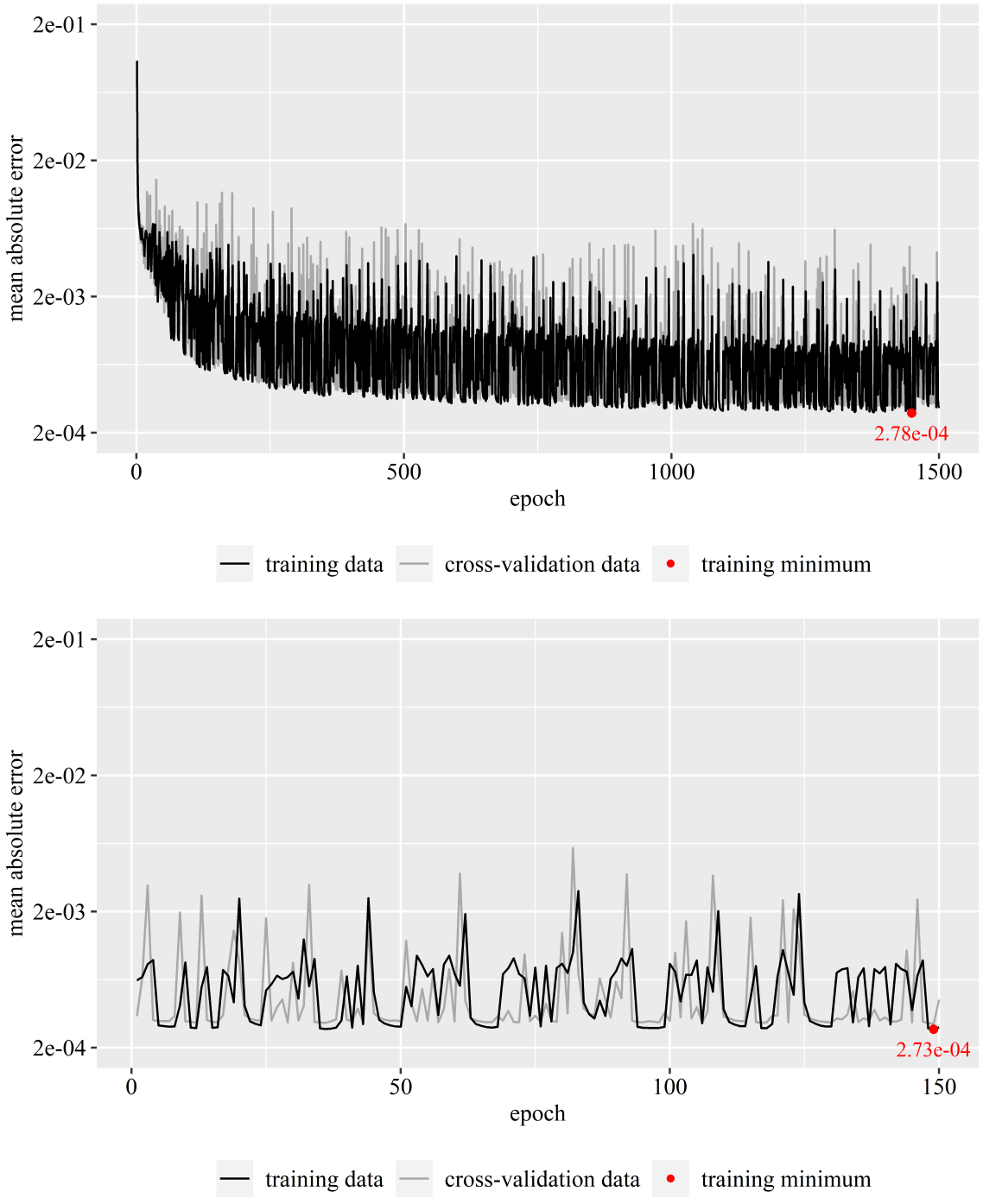

Figure D.13: Neural network trained using the SSE loss function (13 of 20)

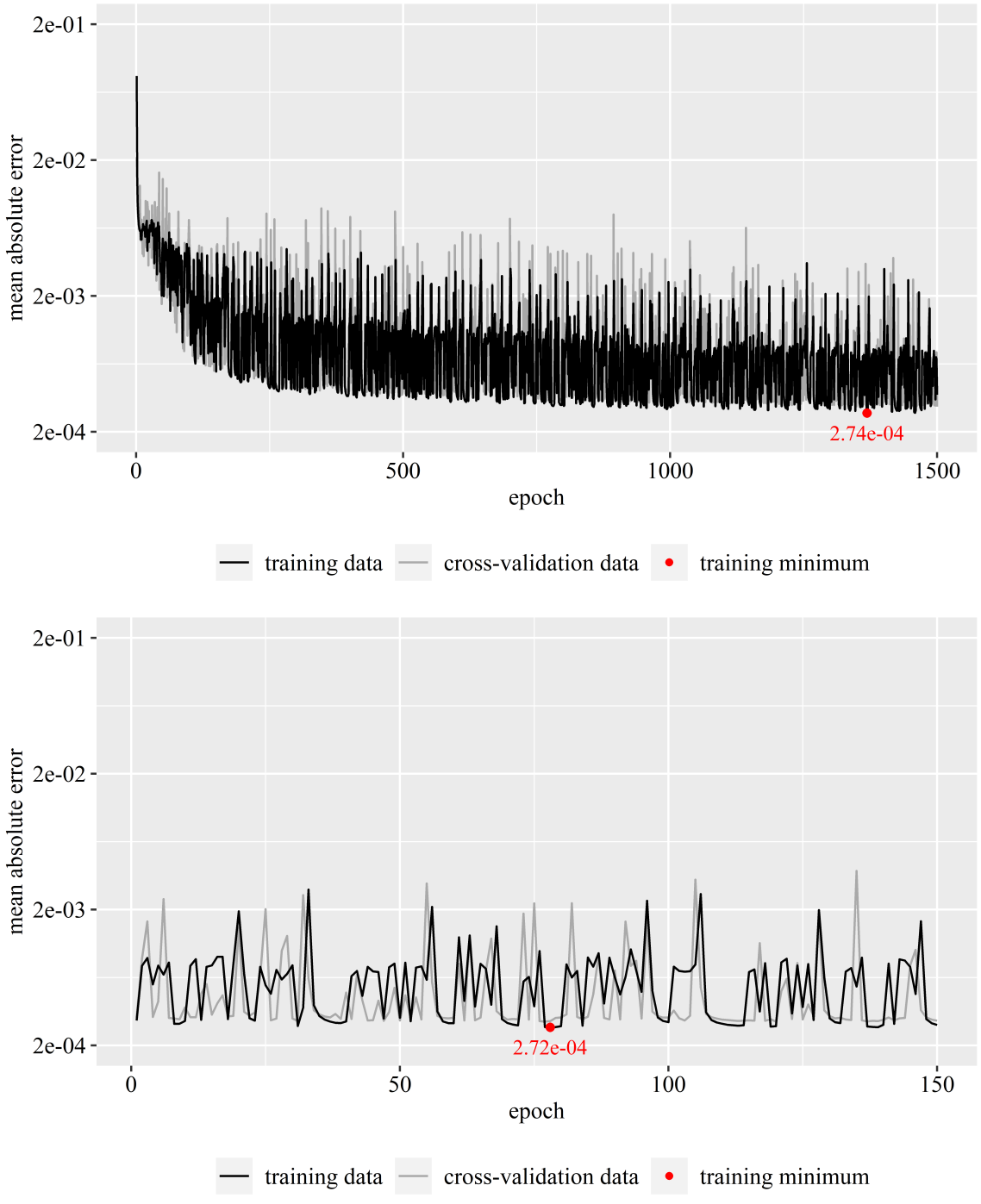

Figure D.14: Neural network trained using the SSE loss function (14 of 20)

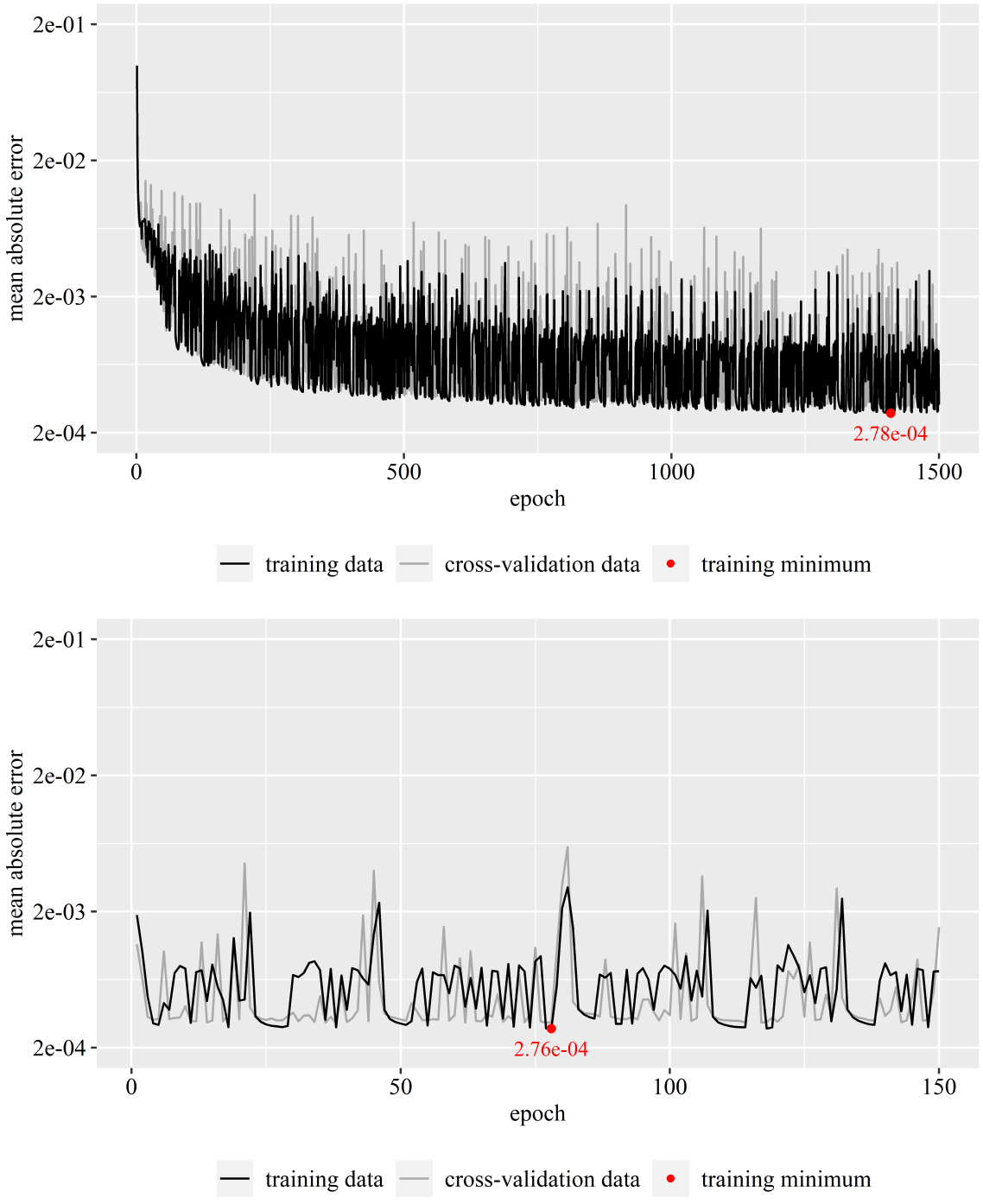

Figure D.15: Neural network trained using the SSE loss function (15 of 20)

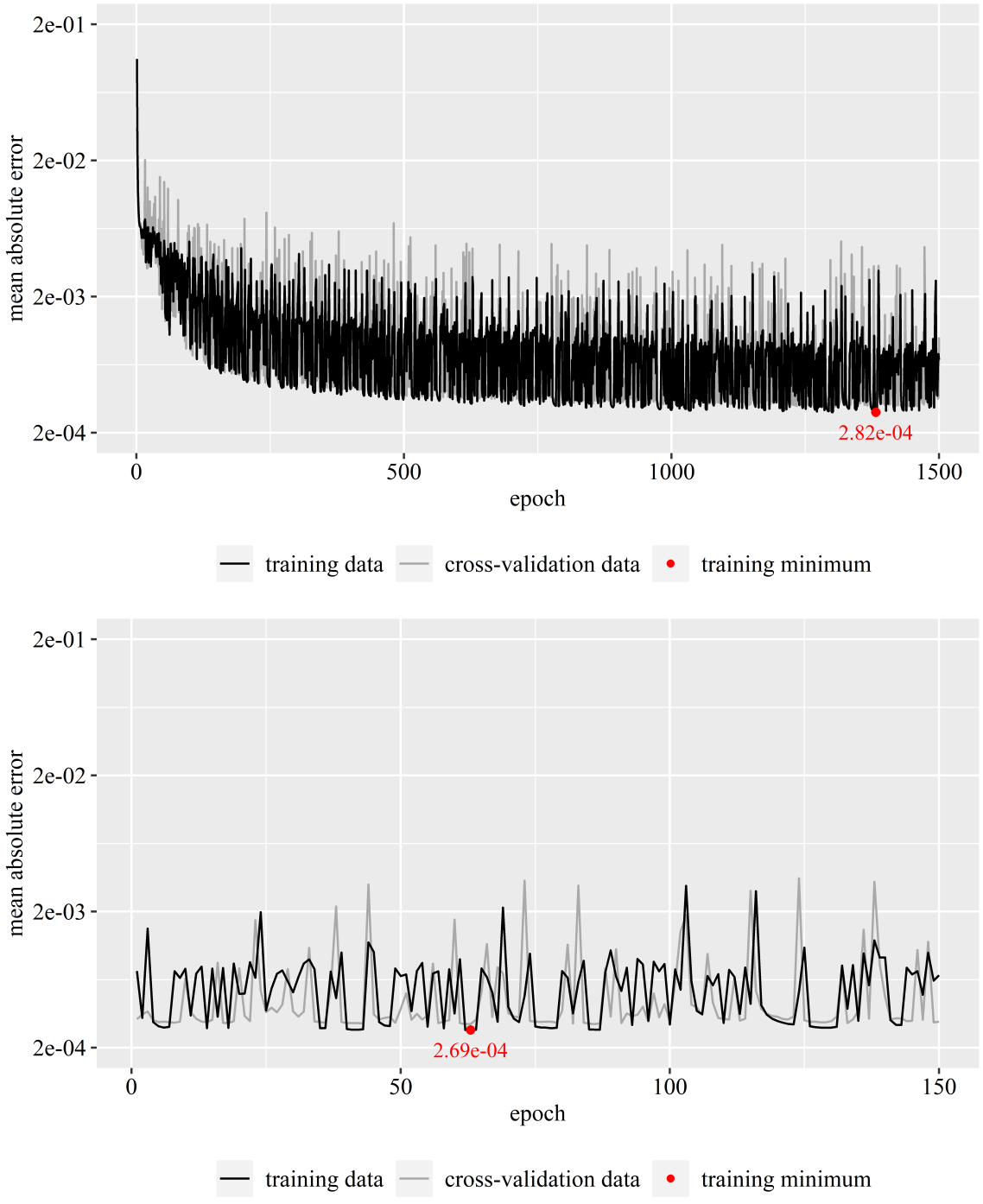

Figure D.16: Neural network trained using the SSE loss function (16 of 20)

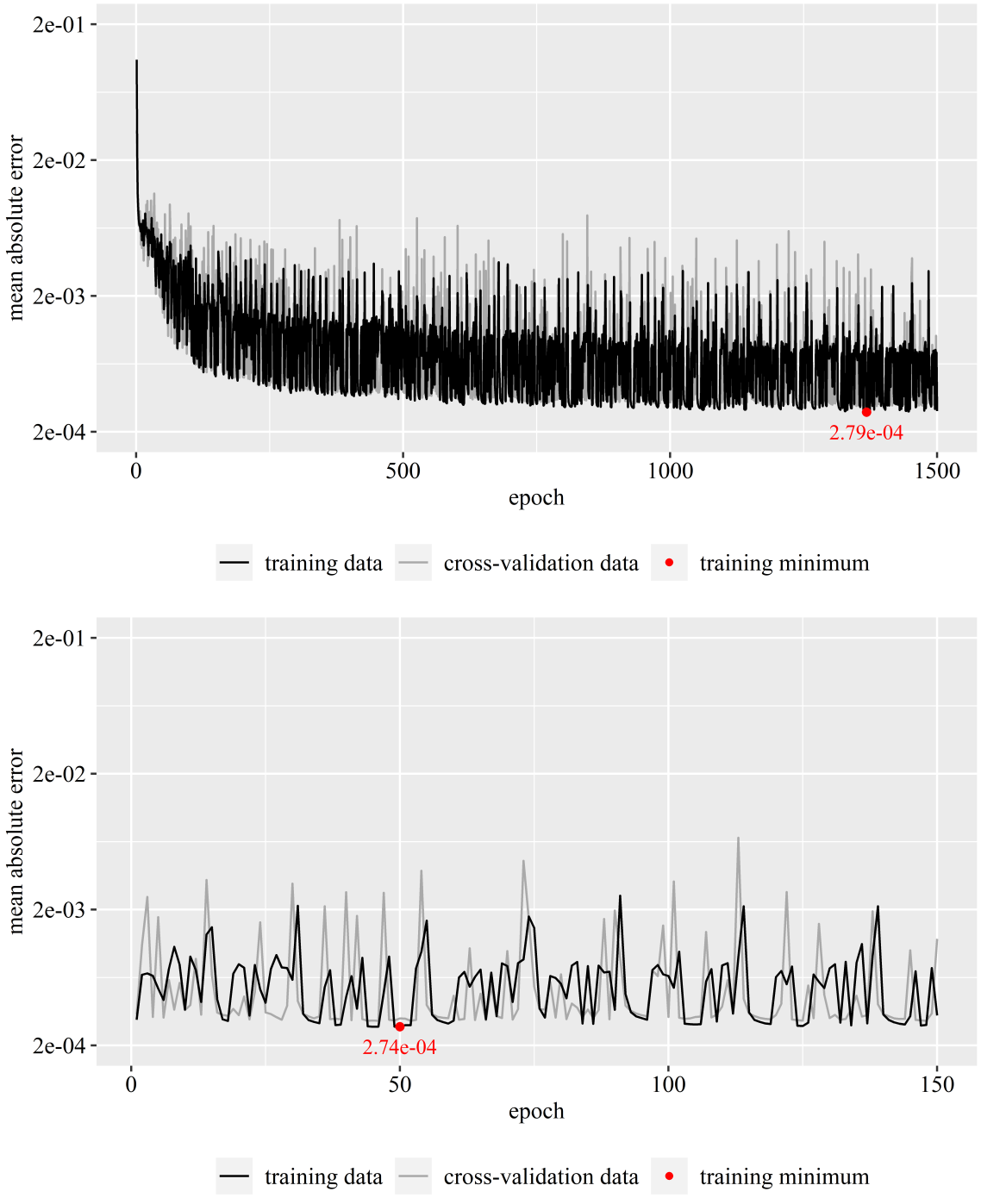

Figure D.17: Neural network trained using the SSE loss function (17 of 20)

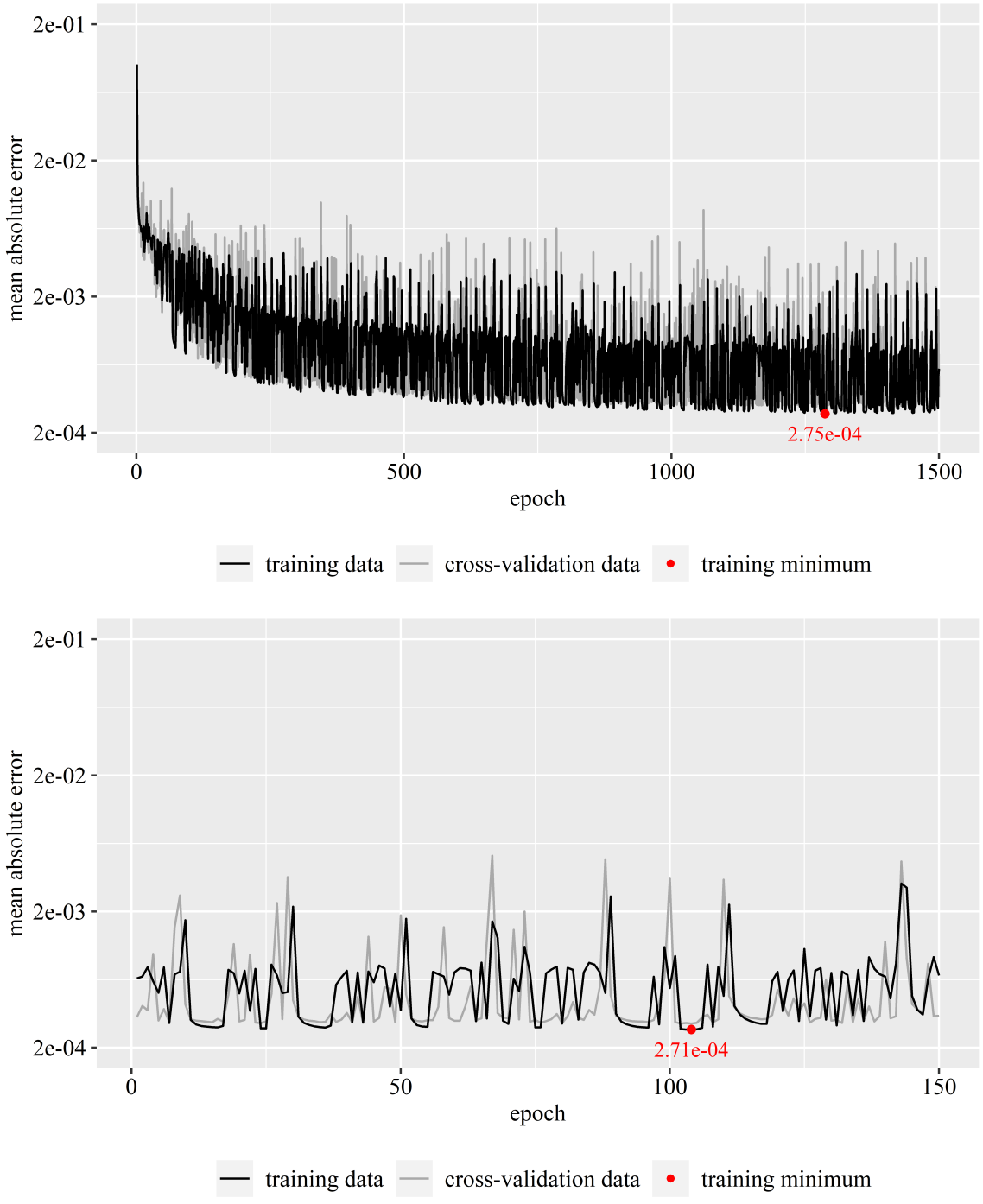

Figure D.18: Neural network trained using the SSE loss function (18 of 20)

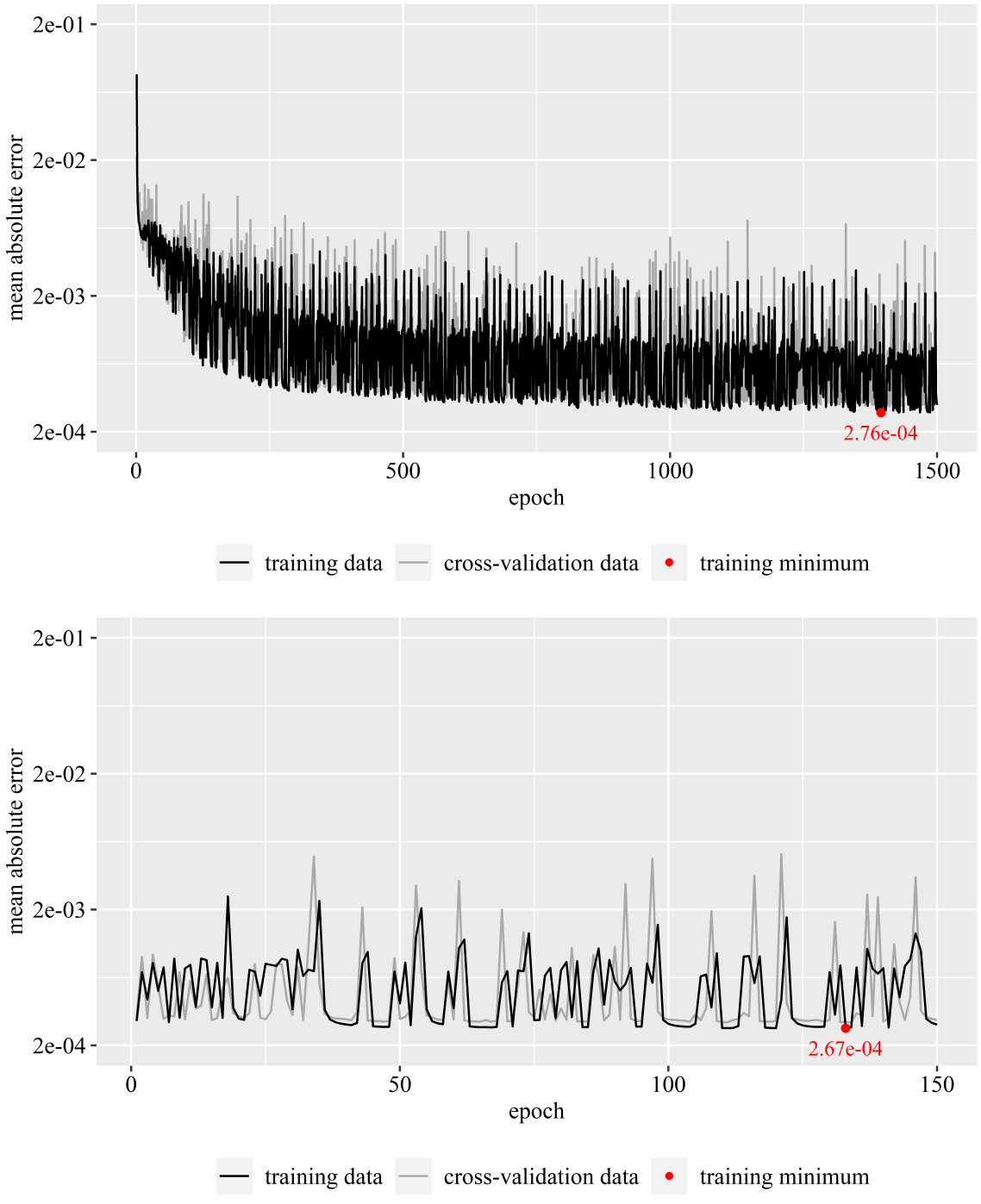

Figure D.19: Neural network trained using the SSE loss function (19 of 20)

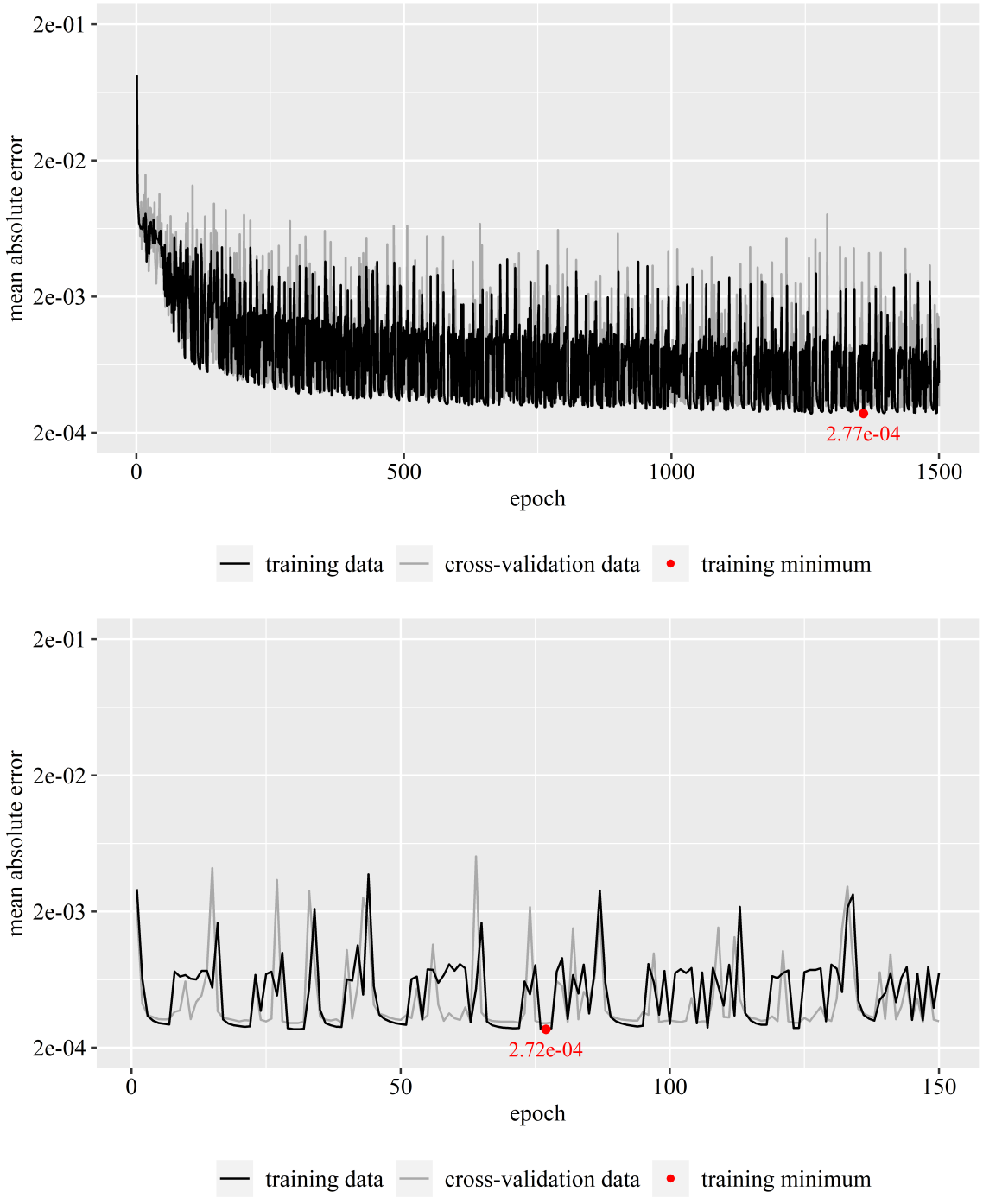

Figure D.20: Neural network trained using the SSE loss function (20 of 20)