Robustness Measure for Deep Neural Networks

Kielhöfer, Lionel L

Robustness Measure for Deep Neural Networks

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE
in
PHYSICS

Author :	Lionel Kielhöfer
Student ID :	s1978462
Supervisor :	Dr. J.N. van Rijn
	A. Bosman MSc.
	Dr. V. Dunjko
	Prof. dr. H.H. Hoos
Second corrector :	

Leiden, The Netherlands, May 24, 2024

Robustness Measure for Deep Neural Networks

Lionel Kielhöfer

Huygens-Kamerlingh Onnes Laboratory, Leiden University
P.O. Box 9500, 2300 RA Leiden, The Netherlands

May 24, 2024

Abstract

Neural networks are susceptible to minor distortions in their input, which can lead to errors they would not otherwise make. This susceptibility, termed as the network's robustness, is a crucial aspect to evaluate. While several methods exist for measuring robustness, they usually suffer from interpretability issues and do not provide a statistical guarantee. In this work, we propose a novel robustness measure that addresses these shortcomings by modeling the robustness as a probability distribution and measuring its 0.05 quantile. Additionally, previous work suggests the potential modeling of robustness through a log-normal distribution. To evaluate this hypothesis and its computational benefits, we introduce an estimator that assumes the distribution is log-normal. A comparison with the standard parameter-free estimator reveals significantly improved computational efficiency with the parametrized approach. However, the log-normal assumption requires further research. The assumption is too strong and needs to be relaxed before the parametrized estimator can reliably be utilized.

Chapter 1

Introduction

Measuring the robustness of a neural network is crucial in situations where the input to the network can deviate slightly from what the network was trained on. These deviations can occur naturally, such as by noisy input data, or maliciously, by attackers intentionally trying to induce errors in the network [1, 2]. Measuring the robustness is especially important for safety-critical systems [3], where a misclassification could result in harm. In these situations, a network that is less sensitive to these distortions might be preferred over an accurate one.

The robustness of a neural network is commonly measured on a per-instance basis. Multiple input images are selected, and the network's robustness against perturbations on these images is examined. For each image, all distortions within a chosen distance are considered. If the network correctly classifies these distortions, it is deemed locally robust on the respective image [4–8]. Once the local robustness is measured across multiple images, various robustness measures can be employed to gauge the network's overall robustness. The most common measure is the *robust accuracy*, calculated as the percentage of images on which the network demonstrates local robustness with respect to a given ϵ value [2, 6, 9–16]. However, this measure is affected by the choice of the maximum distance threshold ϵ for being locally robust, and does not provide a statistical guarantee. Choosing a suitable threshold for robust accuracy is highly task-specific, interpreting the robust accuracy thus requires a deep understanding of the task under evaluation. Moreover, local robustness is binary: it indicates whether the network is locally robust on a given image for the chosen distance threshold or not. Therefore, it may not provide information about the extent of the network's robustness on the image.

Bosman et al. [8] introduce the concept of the critical epsilon of a net-

work on an image, representing the largest distortion for which the network maintains local robustness on that image. This provides additional insight into the network's robustness on an image, as it indicates the threshold at which the network attains local robustness on that image. They use this concept to analyze how the critical epsilon is distributed for various neural networks trained on the MNIST dataset [17]. These distributions, termed *robustness distributions*, entail significant computational costs for their computation. Based on their findings, the authors note that these distributions exhibit similarities to a log-normal distribution.

In this thesis, we build upon the work of Bosman et al. [8]. Firstly, we introduce a novel robustness measure that relies on the robustness distribution. Specifically, our measure involves estimating a small quantile of the robustness distribution and constructing a confidence interval for it. Compared to the robust accuracy, our measure eliminates the need for a deep understanding of the task under evaluation, thereby enabling interpretaion and comparison of robustness between different networks even by those without expertise in the specific task. Additionally, by leveraging the robustness distribution instead of relying on local robustness, our measure provides a more comprehensive insight into the network's robustness, accompanied by a statistical guarantee.

Secondly, we analyse the effectiveness and computational advantages of the hypothesis proposed by Bosman et al. [8] that robustness distributions adhere to a log-normal distribution. To do this, we employ two distinct estimators. The first estimator is a Bayesian estimator that assumes robustness distributions adhere to a log-normal distribution, while the second operates without any assumption, being a standard parameter-free estimator. We utilize the outcome generated by the parameter-free estimator as our ground truth for comparison.

Our analysis indicates that, while the log-normal assumption represents progress, it proves overly restrictive as we find that the underlying distribution is not log-normal, but close to it. The Bayesian estimator's results align with the ground truth approximately 80% of the time. However, for an estimator to be considered reliable, it should consistently yield accurate results. Moreover, the Bayesian estimator's outcomes exhibit considerable deviations, rendering it inadequate for drawing definitive conclusions based on a single measurement. Nonetheless, it offers a notable advantage in terms of computational efficiency, requiring only 30 images. In contrast, the ground truth estimation, which makes no assumptions, demands approximately 1000 images, significantly increasing the computational burden. We are confident that with further research into relaxing the log-normal assumption, it is possible to develop an efficient and accu-

rate estimator.

This thesis continues as follows: In chapter 2 we highlight relevant related work that has been performed in local and global robustness and some background information on these topics. In chapter 3 we introduce our measure and show the details of both the estimators that are used in the experiments. The results of these experiments are shown and discussed in chapter 4. We finish with a conclusion in chapter 5.

Chapter 2

Background and Related Work

Background

To determine the local robustness or critical epsilon of a neural network on a given image, one uses Neural Network Verification (NNV) algorithms. Neural Network Verification (NNV) provides a formal method of ascertaining whether a neural network adheres to a desired property between its input and output. In our case, this property would be the local robustness of a network on a given image. This is used to calculate the critical epsilon and, consequently, the robustness distribution. NNV algorithms can either be complete or incomplete. With an incomplete algorithm, there is no guarantee that the desired property holds for the network or not. A complete algorithm on the other hand will provide that guarantee and is thus more desirable in most cases. In this thesis, we exclusively utilize results obtained from complete verification methods.

In NNV a property is defined by a mathematical formula that establishes a relationship between the inputs of a network and its outputs. The objective of an NNV algorithm is therefore to verify whether a given network adheres to this specified formula. To this extent, NNV algorithms often use satisfiability modulo theories (SMT) solvers [16, 18]. A NNV property can also be formulated as a mixed-integer programming problem, thus MIP solvers have also been used for NNV algorithms [6, 7]. In either formulation, solving the problem is computationally demanding, requiring a lot of compute time and memory. To address this challenge, current state-of-the-art networks employ branch and bound based algorithms [19]. These algorithms effectively partition the solution space into smaller regions, which are computationally less expensive to solve. By doing so, regions that don't contain the solution can be discarded early in the

process, thus conserving computational resources.

The local robustness property of a neural network for an image can formally be written as follows:

Definition 1 (local robustness)**.** Consider a neural network $f : \mathbb{R}^n \to \mathbb{N}$ that classifies images with n pixels, an image $x_0 \in \mathbb{R}^n$, and a distortion ϵ. We say that the network is locally robust, or ϵ-robust, on x_0 if $\forall x \in \mathbb{R}^n : ||x - x_0||_\infty < \epsilon$ we have that $f(x) = f(x_0)$ (the same classification).

The idea of the local robustness is to examine all possible distortions that are at most epsilon away from your image and see if the network classifies these distortions the same.

As previously mentioned, Bosman et al. [8] use the local robustness to define a property that encompasses the robustness of a network, on an image, better:

Definition 2 (critical robustness)**.** Consider a neural network $f : \mathbb{R}^n \to \mathbb{N}$ that classifies images with n pixels, and an image $x_0 \in \mathbb{R}^n$. The critical robustness, or critical epsilon, of the network on that image is the distortion ϵ^* such that the network is locally robust on x_0 for all $\epsilon \leq \epsilon^*$, but not locally robust for all $\epsilon > \epsilon^*$.

The critical robustness of a network on an image represents the maximum distortion for which the network retains local robustness on that image.

Related Work

A robustness measure seeks to determine a universal value of robustness for a given neural network. Typically, this is achieved by computing the local robustness of a network across a set of images. The most commonly utilized robustness measure, to our knowledge, is the *robust accuracy* [2, 6, 9–16]. This measure is also known by various other names, including astuteness [9], adversarial error rate [10], adversarial accuracy [6], and certified accuracy [14]. Additionally, it is equivalent to 1 minus the adversarial frequency [13]. The robust accuracy is calculated as the percentage of images on which the network demonstrates local robustness.

Although not a robustness measure, the average critical epsilon is sometimes used as an indicator of the robustness of a neural network [2, 6, 20–23]. We refer to this as the *average minimum distortion* [2], but it is also known as the average verified bound [20] and the mean minimum adversarial distortion [6].

Bastani et al. [13] introduce the adversarial severity. This closely resembles the average minimum distortion; however, it only considers critical epsilon values that fall below a chosen threshold. This measure aims to quantify the severity of distortion when the network fails to maintain local robustness on an image.

In their work, Katz et al. [16] discuss the concept of global adversarial robustness. They consider a network globally adversarial robust if it maintains local robustness on every possible image. They note that they were only able to prove this property for small networks, due to the domain of possible images being very large.

Chapter 3

Method

We build upon the work of Bosman et al. [8] and introduce our measure, which we call the σ-robustness:

Definition 3 (σ-robustness)**.** Consider a quantile $0 < \sigma < 1$. The σ-robustness of the network is the function $M_\sigma : \{\mathbb{R}^n \to \mathbb{N}\} \to \mathbb{R}_{>0}$ that maps a neural network $f : \mathbb{R}^n \to \mathbb{N}$, that classifies images with n pixels, as follows:

$$M_\sigma(f) = \epsilon \text{ such that } \mathbb{P}(\epsilon^* \leq \epsilon) = \sigma$$

Here, ϵ^* represents the critical epsilon, and the probability distribution is over all possible critical epsilons. The probability of a specific critical epsilon ϵ^* is determined by how likely the network has that critical epsilon on an arbitrary image.

The σ-robustness denotes the maximum distortion that the network maintains local robustness on any arbitrary image, provided that we disregard the $(\sigma * 100)\%$ least critical robust images. The value of σ represents the permitted error and is chosen to be very small. In our experiments, it is set to 0.05.

One of the key observations made by Bosman et al. [8] is the resemblance of robustness distributions to log-normal distributions. If this is indeed true, it presents an opportunity for exploitation. Adopting a naive approach, we will assume that robustness distributions are log-normal distributions. We will see later that this assumption is too strong. Nonetheless, it allows us to gauge the computational time saved by making this assumption. Additionally, we need to assess the accuracy of the assumption. To accomplish this, we also measure the σ-robustness in the standard

manner without any assumptions, serving as our ground truth and providing a baseline for computational cost.

Measurements of the σ-robustness are conducted using estimators. To leverage the log-normal assumption, we will employ a Bayesian estimator. We will use the standard parameter-free estimator to estimate the ground truth, thereby avoiding any assumptions about the distribution. For each measurement of the σ-robustness, we will construct 95% confidence intervals. These intervals allow us to compare different measurements, as non-overlapping intervals indicate distinct measurements. *

Bayesian Estimator

We utilize Bayesian statistics to implement the assumption that a robustness distribution follows a log-normal distribution. Formally, this is done by employing a surrogate model S_θ that is parameterized by certain parameters θ which are unknown to us. The surrogate model incorporates our beliefs about the distribution while allowing for some flexibility through its parameters. In our case, the surrogate model corresponds to the log-normal distribution. The log-normal distribution is characterized by two parameters, μ and σ. Therefore, we define $\theta := (\mu, \sigma)$. Mathematically, the surrogate model is employed as follows:

$$\mathbb{P}(\epsilon^* \leq \epsilon \mid \theta) = \int_0^\epsilon S_\theta(x)dx \tag{3.1}$$

Where we specify the dependence on the parameters θ on the left side.

Once we gather some data D, we can gain insight into our parameters θ. Formally, with the data D, we create a probability distribution over possible model parameters as follows:

$$\mathbb{P}(\theta \mid D) \propto \mathbb{P}(D \mid \theta)\mathbb{P}(\theta) \tag{3.2}$$

Where we have used Bayes theorem in the last step.

One can use the prior $\mathbb{P}(\theta)$ to encode beliefs about the parameters θ. In our case, where we lack concise information about the parameters, we opt for a uniform prior.

The distribution over model parameters also induces a distribution over the σ-robustness as follows:

$$\mathbb{P}(\epsilon \mid D) = \int_\theta \mathbb{P}(\epsilon \cap \theta \mid D)d\theta = \int_\theta \mathbb{P}(\epsilon \mid \theta, D)\mathbb{P}(\theta \mid D)d\theta \tag{3.3}$$

*Full details here: `https://github.com/quantagenmtg/Robustness-Metric`

Where $\epsilon = M_\sigma(f)$ is the σ-robustness of a network f.

We can remove the dependence on D in $\mathbb{P}(\epsilon \mid \theta, D)$ as it is not relevant to the value of ϵ once the parameters are set. We also observe that $\mathbb{P}(\epsilon \mid \theta) = 1$ when $\int_0^\epsilon S_\theta(x)dx = \sigma$ and 0 otherwise. We can thus rewrite equation 3.3 as follows:

$$\mathbb{P}(\epsilon \mid D) = \int_{\theta:\int_0^\epsilon S_\theta(x)dx=\sigma} \mathbb{P}(\theta \mid D)d\theta \propto \int_{\theta:\int_0^\epsilon S_\theta(x)dx=\sigma} \mathbb{P}(D \mid \theta)\mathbb{P}(\theta)d\theta \quad (3.4)$$

Where we have used Bayes theorem in the last step just as described in equation 3.2. We can neglect the constant of proportionality since, in practice, we can normalize the probabilities after calculating them.

This integral is too complex to solve directly. In practice, we will rely on Monte Carlo integration. Further details can be found in the Appendix.

Our dataset consists of critical epsilon bounds, as it is infeasible to obtain the exact critical epsilon. We denote the dataset as:

$$D = \{(\epsilon_1^L, \epsilon_1^U), \ldots, (\epsilon_n^L, \epsilon_n^U)\} \quad (3.5)$$

Where ϵ_i^L represents the lower bound of the critical epsilon of the network on image i, and ϵ_i^U represents the upper bound. Since the data points are i.i.d. we see that the following holds:

$$\mathbb{P}(D \mid \theta) = \prod_{i=1}^{n} \int_{\epsilon_i^L}^{\epsilon_i^U} S_\theta(x)dx \quad (3.6)$$

Algorithm

Equation 3.4 allows us to construct a distribution over possible values for the σ-robustness $M_\sigma(f) = \epsilon$. From this distribution, we build a 95% confidence interval. Algorithm 1 outlines the process for calculating the distribution over all potential values of ϵ. Since it's not feasible to numerically compute the probability for every ϵ, we instead calculate it for a finite number of bins. At each step, we collect a new data point and update the distribution. We then remake the bins for the area with nonzero probability. We halt the process of adding data points either when the distribution attains an uncertainty level γ or when it converges to zero probability for all ϵ. The latter usually happens due to the restrictive nature of the log-normal assumption. In our experiments, the critical epsilon values have an uncertainty of 0.002; hence, we set γ to that value. Figure 3.1 shows an example of the resulting distribution and the ground truth 95% confidence interval.

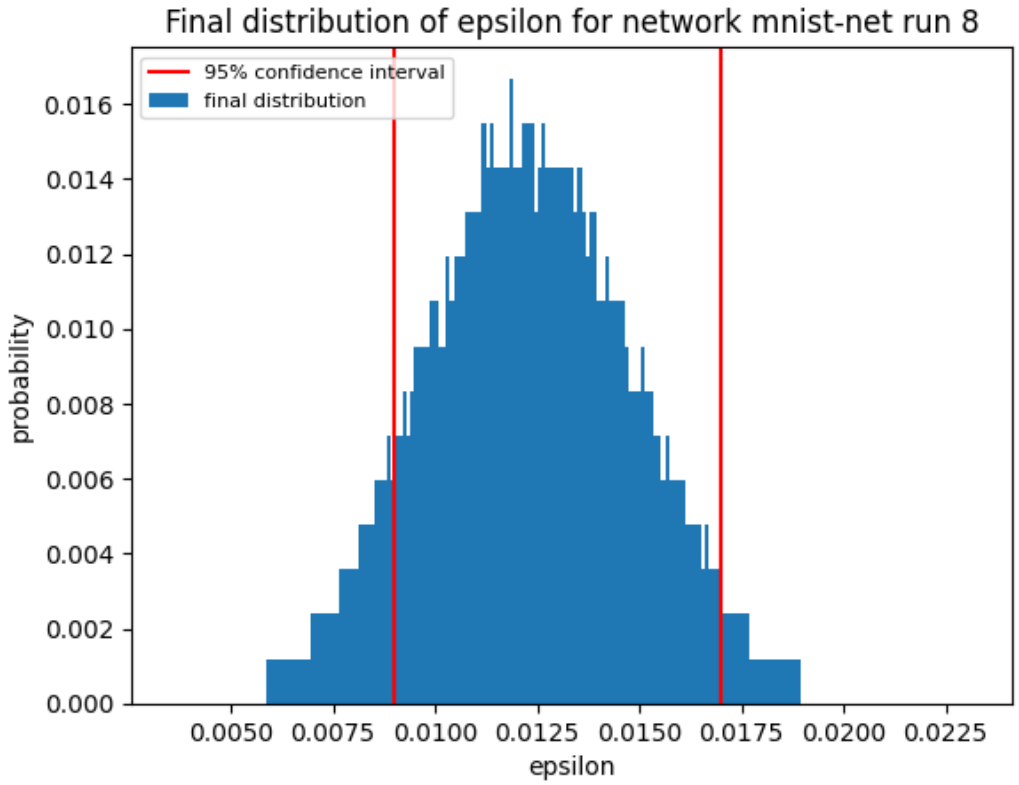

Figure 3.1: *Blue is the distribution given by algorithm 1. The vertical red lines indicate the ground truth 95% confidence interval.*

Algorithm 1 Distribution over σ-robustness

Input: Number of bins n, Smallest edge ϵ_{min}, Largest edge ϵ_{max}, γ, σ
Output: ϵ_i for $i \in \{1, \ldots, n+1\}$ and $\mathbb{P}(\epsilon_i < \epsilon \leq \epsilon_{i+1} \mid D)$ for $i \in \{1, \ldots, n\}$

$D \leftarrow \emptyset$ ▷ Initialize dataset
while $\epsilon_{max} - \epsilon_{min} > 2\gamma$ **do**
  $\epsilon_i \leftarrow \epsilon_{min} + (i-1)\frac{\epsilon_{max} - \epsilon_{min}}{n} \quad \forall i \in \{1, \ldots, n+1\}$
  $D \leftarrow D \cup \{(\epsilon^L, \epsilon^U)\}$ ▷ Gather a new data point
  Calculate $\mathbb{P}(\epsilon_i < \epsilon \leq \epsilon_{i+1} \mid D) \quad \forall i \in \{1, \ldots, n\}$ ▷ See Appendix
  if $\mathbb{P}(\epsilon_i < \epsilon \leq \epsilon_{i+1} \mid D) = 0$ for $i \in \{1, \ldots, n\}$ **then**
    $D \leftarrow D \setminus \{(\epsilon^L, \epsilon^U)\}$ ▷ Remove last data point
    Break loop
  end if
  Take $l \leftarrow \min\{i : \mathbb{P}(\epsilon_i < \epsilon \leq \epsilon_{i+1} \mid D) \neq 0\}$
  Take $u \leftarrow \max\{i : \mathbb{P}(\epsilon_{i-1} < \epsilon \leq \epsilon_i \mid D) \neq 0\}$
  $\epsilon_{min} \leftarrow \epsilon_l, \epsilon_{max} \leftarrow \epsilon_u$ ▷ Update domain where nonzero probability
end while

The first 3 **inputs** of the algorithm represent the initial area of the distribution which the algorithm investigates. In our experiments, we use $a_{min} = 0$, $a_{max} = 0.4$, and around 200 bins. The **input** γ depicts the uncertainty we want to reach. The **input** σ depicts the quantile we are estimating. The algorithm **outputs** bins with their corresponding probabilities. In **line 1** we initialize the dataset. In **line 2-12** we perform a while loop that gathers data until we have reached the uncertainty γ or we have reached zero probability (**lines 6-9**). In **line 3** we update the bin edges such that we have n bins of equal size. In **line 4** we insert a new data point into our dataset. In **line 5** we update the distribution with the new bins and the new data point. In **line 10 - 12** we reduce the area that the algorithm investigates. Everything below a_{min} and above a_{max} has a probability of zero.

Parameter-free estimator

For the parameter-free estimator we will follow Meeker et al. [24]. We build a 95% confidence interval and use it as the ground truth to evaluate the Bayesian estimator.

In our experiments we are given samples $X_1, ..., X_n$ of critical epsilons bounds. To simplify matters, we will assume that these are not bounds but values. Mathematically this will not make a difference, but it will simplify a lot of equations. With these samples we can now make order statistics $X_{(1)} \leq X_{(2)} \leq, \cdots, \leq X_{(n)}$.

Our goal is to find an appropriate interval $[X_{(l)}, X_{(u)}]$ such that we form a 95% confidence interval $\mathbb{P}(X_{(l)} \leq \epsilon \leq X_{(u)}) \geq 0.95$. For $X_{(l)} \leq \epsilon$ we need at least l out of the n samples to be less or equal to ϵ. Since for any sample i we have that $\mathbb{P}(X_i \leq \epsilon) = \sigma$ it follows that $\mathbb{P}(X_{(l)} \leq \epsilon) = \mathbb{P}(B(n, \sigma) \geq l) = 1 - \mathbb{P}(B(n, \sigma) \leq l - 1)$ where $B(n, p)$ is the binomial distribution for n trials and probability p of success for one trial. Following this logic we get:

$$\mathbb{P}(X_{(l)} \leq a \leq X_{(u)}) = \mathbb{P}(X_{(l)} \leq a) - \mathbb{P}(X_{(u)} \leq a) = \\ \mathbb{P}(B(n, \sigma) \leq u - 1) - \mathbb{P}(B(n, \sigma) \leq l - 1) \quad (3.7)$$

There are many choices of l and u that are possible, we pick them such that we get a two sided confidence interval. This is given by $\mathbb{P}(X_{(l)} \geq a) = \mathbb{P}(X_{(u)} \leq a) = 0.025$. The same is done when constructing the confidence interval with the Bayesian estimator.

Version of May 24, 2024– Created May 24, 2024 - 17:41

To go back to the original formulation, where the samples are bounds, we just take the interval $[X_{(l)} - 0.002, X_{(u)}]$ instead.

Version of May 24, 2024– Created May 24, 2024 - 17:41

Chapter 4

Results

In our experiments, we sought to assess the Bayesian estimator based on three distinct criteria. Firstly, we wanted to evaluate its accuracy by examining its overlaps with the ground truth. Secondly, recognizing that the log-normal assumption might be overly restrictive, we aimed to determine whether the estimator, in its current state, is suitable for comparing the robustness of networks. Thirdly, we aimed to evaluate the total runtime required by the estimator, considering that measuring the critical epsilon of an image is computationally demanding.

To perform these assessments we take multiple measurements using the Bayesian estimator. As mentioned, algorithm 1 will terminate after utilizing approximately 30 instances. Altering the instances it considers will yield different measurements for the Bayesian estimator. In our experiments, we do this 100 times.

To obtain the ground truth we use all instances available to us for the parameter-free estimator. This amounts to around 1000 instances. We thus have 100 different measurements with the Bayesian estimator and 1 ground truth measurement with the parameter-free estimator per network.

Experimental setup

For our experiments, we utilized critical epsilon values obtained from three neural networks pre-trained on the MNIST dataset [17]. This dataset comprises a vast collection of handwritten digits along with their corresponding labels, with 60,000 instances allocated for training and 10,000 for testing. From this pool, Bosman et al. [8] selected the first 1000 test

and 1000 train instances to measure the critical robustness across the designated networks. Notably, only instances correctly classified by the networks were considered, resulting in potentially varied image distributions per network. The networks are: mnist-net, mnist-net 256x4 and mnist relu 4 1024.

Critical robustness for the network on each image was assessed through a parallelized binary search across epsilon values ranging from 0.001 to 0.4 in increments of 0.002. At each epsilon, Bosman et al. [8] verified if the network remained epsilon-robust on the image until determining an interval of size 0.002 wherein the critical epsilon must lie.

Consequently, we obtained critical epsilon bounds and runtimes for each of the three aforementioned neural networks on approximately 1000 test instances. These are used by the estimators.

Furthermore, Bosman et al. [8] analyzed nine additional networks pretrained on the MNIST dataset. However, critical robustness for these networks was measured on only the first 100 test and 100 train instances. Although included in Tables 4.2, and 4.3, we refrain from drawing conclusions based on these networks due to the unreliable ground truth derived from only 100 instances. The additional networks are: mnist-net 256x2, mnist-net 256x6, mnist nn, mnist relu 3 100, mnist relu 3 50, mnist relu 6 100, mnist relu 6 200, mnist relu 9 100, mnist relu 9 200.

Accuracy of Bayesian estimator

To assess the accuracy of the Bayesian estimator, we examine the number of intervals out of the 100 that overlap with the ground truth. If overlap occurs, we analyze the assigned probability to the overlap and check what ratio of the interval is part of the overlap. This can be seen in table 4.1. Out of the 100 intervals, 95 overlap with the ground truth for mnist-net, 81 for mnist-net 256x4, and 86 for mnist relu 4 1024. If an adequate surrogate model is used there should always be overlap with the ground truth. With our current surrogate model, this overlap only occurs around 80% of the time for these networks. Although this is not 100%, it does indicate that a log-normal surrogate model is a step in the right direction, as most of the intervals produced overlap with the ground truth. The table also highlights that the probability assigned to the overlap falls short of the desired level, peaking at 0.65 for mnist-net but being lower for the other two networks. Furthermore, the ratio of overlap with the ground truth is below 0.5 for all three networks, suggesting that the measured intervals predominantly lie outside the ground truth rather than within it.

Version of May 24, 2024– Created May 24, 2024 - 17:41

Network	Frequency of overlap	Average probability of overlap	Average size ratio of overlap
mnist-net	95	0.65	0.48
mnist-net 256x4	81	0.34	0.22
mnist relu 4 1024	86	0.42	0.28

Table 4.1: *Shown is the frequency, average probability and average size of overlap for the Bayesian estimator measurements with the ground truth. The frequency is given by the number of intervals measured with the Bayesian estimator that overlap with the ground truth (out of the 100) for each network. The average probability of the overlap is computed by summing up the probabilities of the generated distribution using algorithm 1 within the overlap region. The size ratio of overlap is determined by dividing the size of the overlap by the size of the interval. The ground truth was calculated using* 1000 *instances with a parameter-free estimator.*

Precision of Bayesian estimator

Since the surrogate model used is not perfect we want to see the distribution of confidence intervals produced by the Bayesian estimator. This gives us an idea of how useful the estimator is for comparing networks. Table 4.2 shows some statistics over the intervals We also illustrate this distribution for the 3 networks that have 1000 instances using a histogram. We take the lower bounds and upper bounds of each interval, these will be the bin edges. The frequency for each bin is given by the number of intervals that contain the epsilon values inside the bin. Figure 4.1 shows this histogram for each of the 3 networks. It is clear from this figure that mnist-net 256x4 is more robust than the other two networks as its ground truth confidence interval is strictly greater than that of the other two networks. With the Bayesian estimator we cannot always make this conclusion. We can see this by looking at the lowest and highest interval in table 4.2. It is possible that, when using the Bayesian estimator, confidence intervals such that mnist-net 256x4 has a smaller interval than the other two networks. This would lead to the conclusion that mnist-net 256x4 is less robust than the other two networks, which would be false.

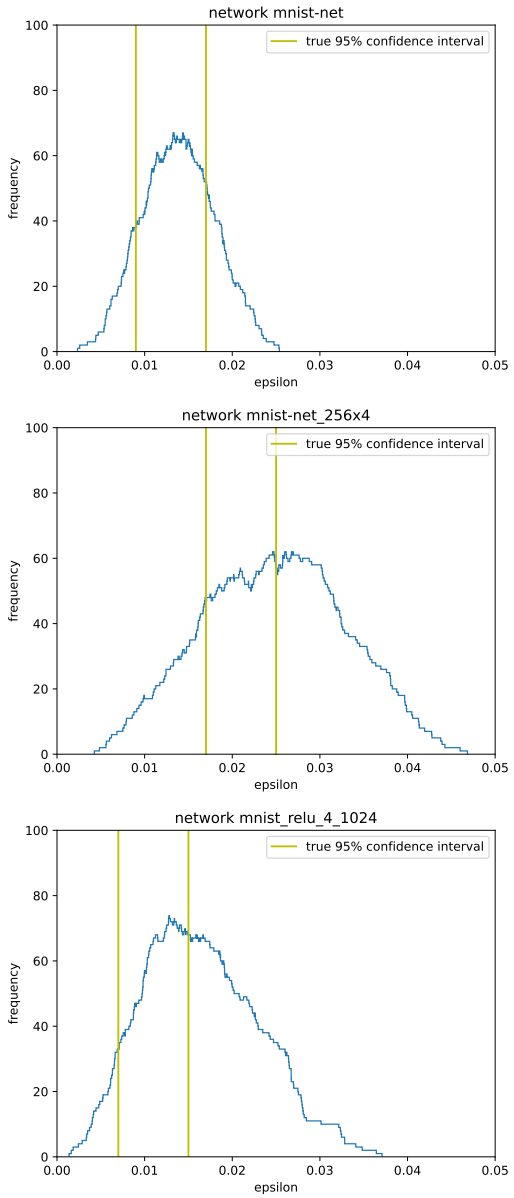

Figure 4.1: *Each figure is for a different network. The blue curve is a histogram that shows the amount of times the epsilon values in a given bin were in one of the 100 different intervals generated using the Bayesian estimator. The yellow vertical lines show the true two sided 95% confidence interval. This was calculated using the parameter-free estimator.*

Network	Average lower bound	Average upper bound	True 95% confidence interval	Lowest interval	Highest interval	Minimum size	Maximum size	Average size	Ground truth size
mnist-net	0.0102	0.0173	(0.0090, 0.0170)	(0.0023, 0.0077)	(0.0151, 0.0254)	0.0001	0.0109	0.0072	0.0080
mnist-net 256x4	0.0172	0.0306	(0.0170, 0.0250)	(0.0043, 0.0100)	(0.0258, 0.0468)	0.0002	0.0210	0.0134	0.0080
mnist relu 4 1024	0.0097	0.0218	(0.0070, 0.0150)	(0.0014, 0.0077)	(0.0213, 0.0371)	0.0004	0.0203	0.0121	0.0080
mnist-net 256x2	0.0126	0.0184	(0.0070, 0.0210)	(0.0073, 0.0149)	(0.0179, 0.0261)	0.0001	0.0089	0.0058	0.0140
mnist-net 256x6	0.0181	0.0304	(0.0050, 0.0330)	(0.0046, 0.0178)	(0.0390, 0.0549)	0.0019	0.0192	0.0123	0.0280
mnist nn	0.0040	0.0082	(0.0010, 0.0110)	(0.0015, 0.0046)	(0.0083, 0.0164)	0.0003	0.0081	0.0042	0.0100
mnist relu 3 100	0.0142	0.0253	(0.0050, 0.0310)	(0.0024, 0.0091)	(0.0304, 0.0438)	0.0001	0.0169	0.0111	0.0260
mnist relu 3 50	0.0166	0.0264	(0.0070, 0.0270)	(0.0082, 0.0190)	(0.0242, 0.0366)	0.0005	0.0138	0.0098	0.0200
mnist relu 6 100	0.0169	0.0293	(0.0090, 0.0270)	(0.0077, 0.0191)	(0.0251, 0.0418)	0.0019	0.0184	0.0123	0.0180
mnist relu 6 200	0.0176	0.0310	(0.0050, 0.0330)	(0.0066, 0.0193)	(0.0326, 0.0516)	0.0002	0.0190	0.0134	0.0280
mnist relu 9 100	0.0119	0.0220	(0.0090, 0.0270)	(0.0024, 0.0094)	(0.0193, 0.0358)	0.0001	0.0176	0.0102	0.0180
mnist relu 9 200	0.0174	0.0303	(0.0090, 0.0330)	(0.0068, 0.0220)	(0.0281, 0.0470)	0.0002	0.0188	0.0129	0.0240

Table 4.2: *Statistics for the Bayesian estimator. The ground truth is calculated using the parameter-free interval. The lowest and highest intervals contain the lowest and highest value of epsilon respectively. The size of an interval is given by subtracting its lower bound from its upper bound. For the networks in bold, their ground truth was calculated using* 1000 *instances. For the remaining networks, it was calculated using only* 100 *instances.*

23

Runtime of Bayesian estimator

Table 4.3 presents statistics for the runtimes of the Bayesian estimator, it also includes the runtime of the parameter-free estimator. Additionally, to visualize the runtime distribution, we plot a cumulative density plot of the runtimes of the Bayesian estimator in Figure 4.2. The runtime is given by the total time it took to calculate the critical epsilon for the network on each instance used for the measurement. For the ground truth all instances are used, the runtime is thus the total time required to compute the critical epsilon for the network on every instance in the dataset. The table indicates a significant speedup achieved by the Bayesian estimator compared to the parameter-free estimator, typically around 100 times faster. The figure shows that the runtime distributions for mnist-net 256x4 and mnist relu 4 1024 are very similar. By looking at the steepest parts of the cumulative density plot we can see that for mnist-net quite a lot of runtimes are at the lower tail of its distributions. for the other two networks this is not the case as they do not have a lot of runtimes at either tales of their distribution.

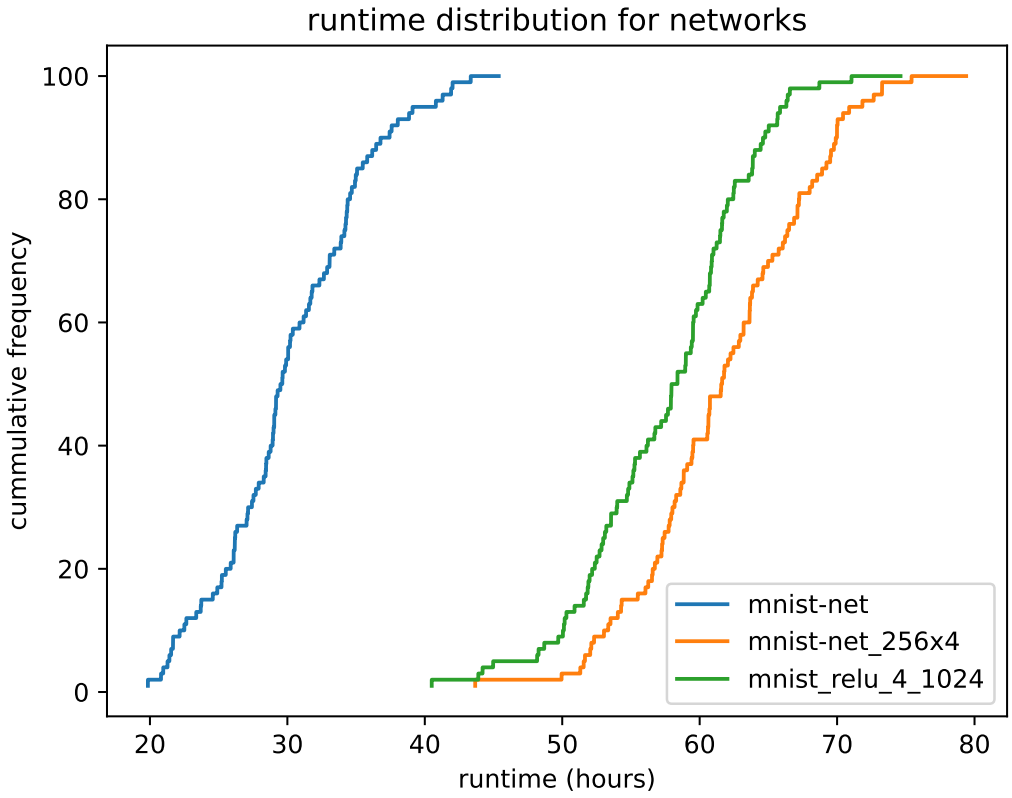

Figure 4.2: *Cumulative frequency of runtime for the Bayesian estimator on three networks. Each curve is a histogram showing the cumulative runtime of the* 100 *different intervals generated using the Bayesian estimator. The cumulative frequency for a bin gives the amount of intervals that had less than or equal to the runtimes in the bin. The runtime for an interval is given by the total runtime of each instances used to create the interval. The runtime of the instance comes from measuring its critical robustness.*

Network	Minimum runtime (hours)	Maximum runtime (hours)	Average runtime (hours)	Ground truth runtime (hours)
mnist-net	19.85	45.38	30.32	4049.16
mnist-net 256x4	43.67	79.39	61.94	9631.47
mnist relu 4 1024	40.50	74.62	57.73	10791.52
mnist-net 256x2	2.68	14.58	8.09	432.95
mnist-net 256x6	49.53	80.41	62.12	1716.84
mnist nn	3.80	19.55	11.52	360.63
mnist relu 3 100	38.02	64.08	49.29	1155.38
mnist relu 3 50	12.29	30.64	21.81	827.41
mnist relu 6 100	44.55	66.67	53.70	1184.82
mnist relu 6 200	45.34	67.69	56.10	1219.33
mnist relu 9 100	39.76	65.84	51.86	1153.48
mnist relu 9 200	51.18	77.18	62.74	1737.23

Table 4.3: *Runtime statistics for the Bayesian estimator are given in the first 3 columns. The runtime for the parameter-free estimator is given in the last column. For the networks in bold, their ground truth was calculated using* 1000 *instances. For the remaining networks, it was calculated using only* 100 *instances.*

Chapter 5

Conclusion

In this thesis we have introduced a measure that gives a guarantee for the robustness of a network and is easily interpretable. To use this measure, we build upon previous work on robustness distributions and introduce a Bayesian estimator that assumes robustness distributions are log-normal. To evaluate this approach, we compared it with a standard parameter-free estimator, which served as our ground truth.

Our experiments revealed that while the log-normal assumption provided computational efficiency for the Bayesian estimator, it was overly restrictive, leading to deviations from the ground truth in about 20% of cases. We find that, robustness distributions are not perfectly log-normal. The assumption thus restricts the estimator too much by not giving it the freedom to explore different types of distributions. However, the Bayesian estimator demonstrated significant speed advantages, converging more than 100 times faster than the parameter-free estimator.

Further research needs to be invested into a proper surrogate model for the Bayesian estimator, we find that using a log-normal is a step in the right direction but too restrictive. A model that is similar to a log-normal but not as strict, such as a mixture model, should be investigated. Additionally, other estimators should also be explored, such as a standard frequentist estimator that assumes a normal distribution after the data is transformed.

Bibliography

[1] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[2] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[3] Kyle D. Julian, Mykel J. Kochenderfer, and Michael P. Owen. Deep neural network compression for aircraft collision avoidance systems. *CoRR*, abs/1810.04240, 2018. URL `http://arxiv.org/abs/1810.04240`.

[4] Rudy Bunel, Ilker Turkaslan, Philip H. S. Torr, Pushmeet Kohli, and Pawan Kumar Mudigonda. A unified view of piecewise linear neural network verification. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 4795–4804, 2018.

[5] Alessandro De Palma, Rudy Bunel, Alban Desmaison, Krishnamurthy Dvijotham, Pushmeet Kohli, Philip H. S. Torr, and M. Pawan Kumar. Improved branch and bound for neural network verification via lagrangian decomposition. *CoRR*, 2021.

[6] Vincent Tjeng, Kai Yuanqing Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[7] Elena Botoeva, Panagiotis Kouvaros, Jan Kronqvist, Alessio Lomuscio, and Ruth Misener. Efficient verification of relu-based neural networks via dependency analysis. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 3291–3299. AAAI Press, 2020.

[8] Annelot W. Bosman, Holger H. Hoos, and Jan N. van Rijn. A preliminary study of critical robustness distributions in neural network verification. *6th Workshop on Formal Methods for ML-Enabled Autonomous Systems (FoMLAS)*, 2023.

[9] Yao-Yuan Yang, Cyrus Rashtchian, Hongyang Zhang, Ruslan Salakhutdinov, and Kamalika Chaudhuri. A closer look at accuracy vs. robustness. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[10] Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Timothy A. Mann, and Pushmeet Kohli. A dual approach to scalable verification of deep networks. In Amir Globerson and Ricardo Silva, editors, *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, pages 550–559. AUAI Press, 2018.

[11] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[12] Ruitong Huang, Bing Xu, Dale Schuurmans, and Csaba Szepesvári. Learning with a strong adversary. *CoRR*, abs/1511.03034, 2015.

[13] Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya V. Nori, and Antonio Criminisi. Measuring neural net robustness with constraints. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2613–2621, 2016.

[14] Linyi Li, Tao Xie, and Bo Li. Sok: Certified robustness for deep neural networks. In *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*, pages 1289–1310. IEEE, 2023.

[15] Uri Shaham, Yutaro Yamada, and Sahand Negahban. Understanding adversarial training: Increasing local stability of neural nets through robust optimization. *CoRR*, abs/1511.05432, 2015.

[16] Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In Rupak Majumdar and Viktor Kuncak, editors, *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I*, volume 10426 of *Lecture Notes in Computer Science*, pages 97–117. Springer, 2017.

[17] Li Deng. The MNIST database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Process. Mag.*, 29(6):141–142, 2012.

[18] Guy Katz, Derek A. Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljic, David L. Dill, Mykel J. Kochenderfer, and Clark W. Barrett. The marabou framework for verification and analysis of deep neural networks. In Isil Dillig and Serdar Tasiran, editors, *Computer Aided Verification - 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part I*, volume 11561 of *Lecture Notes in Computer Science*, pages 443–452. Springer, 2019.

[19] Mark Niklas Müller, Christopher Brix, Stanley Bak, Changliu Liu, and Taylor T. Johnson. The third international verification of neural networks competition (VNN-COMP 2022): Summary and results. *CoRR*, abs/2212.10376, 2022.

[20] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin T. Vechev. AI2: safety and

robustness certification of neural networks with abstract interpretation. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 3–18. IEEE Computer Society, 2018.

[21] Nicolas Papernot, Patrick D. McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, pages 582–597. IEEE Computer Society, 2016.

[22] Nicholas Carlini, Guy Katz, Clark W. Barrett, and David L. Dill. Ground-truth adversarial examples. *CoRR*, abs/1709.10207, 2017.

[23] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 39–57. IEEE Computer Society, 2017.

[24] William Q. Meeker, Gerald J. Hahn, and Luis A. Escobar. *Distribution-Free Statistical Intervals*, chapter 5, pages 73–98. John Wiley & Sons, Ltd, 2017. ISBN 9781118594841. doi: https://doi.org/10.1002/9781118594841.ch5.

[25] Christian P. Robert and George Casella. Monte carlo integration. In *Monte Carlo Statistical Methods*, pages 79–122. Springer New York, 2004.

[26] Christian P. Robert and George Casella. Markov chains. In *Monte Carlo Statistical Methods*, pages 205–265. Springer New York, 2004.

Chapter 6

Appendix

Monte Carlo integration

We adapt the notations used by Robert and Casella [25]. Monte Carlo integration is used to numerically evaluate integrals that are difficult to solve analytically. Consider the following integral:

$$\mathcal{J} = \int_{\mathcal{X}} f(x)p(x)dx \tag{6.1}$$

With p being some probability density over x on the domain $\mathcal{X}$ and f being some integrable function over x. This type of integral is commonly found during statistical inference. An example relevant to this thesis would be the marginal distribution, given by:

$$p(\epsilon) = \int p(\epsilon \mid \theta)p(\theta)d\theta \tag{6.2}$$

Where the unknown density over ϵ is marginalized over the parameters θ for which the density and conditional density are known. We can numerically approximate the integral from equation 6.1 by sampling from the distribution $p(x)$ and then calculating the following:

$$\bar{f}_m = \frac{1}{m}\sum_{i=1}^{m} f(x_i) \approx \mathcal{J} \tag{6.3}$$

Where $x_1, \ldots, x_m$ are the resulting samples.

Calculating distribution $\mathbb{P}(\epsilon \mid D)$

We use Monte Carlo integration to calculate the integral in equation 3.4. We sample $\theta_1, \ldots, \theta_m$ from the prior $\mathbb{P}(\theta)$ and put it into equation 6.3 to get:

$$\mathbb{P}(\epsilon \mid D) \approx \frac{K}{m} \sum_{i=1}^{m} \mathbb{P}(D \mid \theta_i) \tag{6.4}$$

As the dimension of the parameter space low in this work a simple Monte Carlo integration works well. However, if one is working with a more complex surrogate model the dimension of the parameter space might be too large to efficiently sample with the prior. In that case, using Markov Chains [26] might be preferred, as they will guide the samples to areas under the integral that have the most effect on the outcome.

Since the integral in equation 3.4 is over parameters θ such that $\int_0^a S_\theta(\epsilon)d\epsilon = \sigma$, we have to take samples of θ in a special way. S_θ is log-normal, it depends on two parameters μ and s as follows:

$$S_\theta(x) = S_{\mu,s}(x) = \frac{1}{xs\sqrt{2\pi}} \exp\left(-\frac{(\ln x - \mu)^2}{2s^2}\right) \tag{6.5}$$

We have that $\int_0^\epsilon S_\theta(x)dx = \sigma$, thus we lose one degree of freedom. We choose to sample over the median $M := \exp(\mu)$ as it simplifies calculations.

Bosman et al. [8] have only ever found critical epsilon values lower or equal to 0.4 in their experiments. We take that into account by not sampling any M over 0.4, as this gives us distributions that always place low probabilities on critical epsilon values much larger than 0.4. By definition we know that $\int_0^M S_\theta(x)dx = 0.5$, since $\sigma < 0.5$ we know that $\epsilon < M$. We thus sample M uniformly over the range $(\epsilon, 0.4]$.

Once the median is fixed we get $\mu = \ln(M)$ and s by rearranging the log-normal cumulative density function, which is given by:

$$F_{\mu,s}(x) = \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{\ln x - \mu}{s\sqrt{2}}\right)\right] \tag{6.6}$$

Using $\int_0^\epsilon S_\theta(x)dx = \sigma$ and rearranging then gives us:

$$s = \frac{\ln \epsilon - \mu}{\sqrt{2}\operatorname{erf}^{-1}(2\sigma - 1)} \tag{6.7}$$

We can then use Monte Carlo integration as given by equation 6.4 to approximate $\mathbb{P}(\epsilon \mid D)$.

To get the probability of a bin with edges ϵ_1 and ϵ_2 we take 10 samples of ϵ uniformly over the range $(\epsilon_1, \epsilon_2]$. The bins we use get quite small, thus only taking 10 samples of ϵ works well. For each value of ϵ sampled, we then sample 1000 medians M uniformly over the range $(\epsilon, 0.4]$. From this we can calculate the parameters μ and s of the log-normal distribution as described above, resulting in $m = 10000$ log-normal parameters $\theta_1, \ldots, \theta_m$. We can then use Monte Carlo integration to get the probability over the bin. Using equation 3.6 we now get:

$$\mathbb{P}(\epsilon_1 < \epsilon \leq \epsilon_2 \mid D) \approx \frac{K}{m} \sum_{i=1}^{m} \mathbb{P}(D \mid \theta_i) = \frac{K}{m} \sum_{i=1}^{m} \prod_{j=1}^{n} \int_{\epsilon_j^L}^{\epsilon_i^U} S_{\theta_i}(x) dx \tag{6.8}$$

We can calculate $\int_{\epsilon^L}^{\epsilon^U} S_\theta(x) dx$ using the cumulative density function given by equation 6.6. We can ignore K and set it to 1. This is because algorithm 1 calculates $K\mathbb{P}(\epsilon \mid D)$ across the entire nonzero area. The sum of the results for each bin is thus equal to $\frac{1}{K}$ from which we can retrieve K.