IMA Journal of Numerical Analysis (2024) **00**, 1–28
https://doi.org/DOI HERE

Deterministic and probabilistic backward error analysis of neural networks in floating-point arithmetic

THÉO BEUZEVILLE*
Toulouse INP, IRIT, 2 Rue Charles Camichel, F-31071, Toulouse, France

ALFREDO BUTTARI
CNRS, IRIT, 2 Rue Charles Camichel, F-31071, Toulouse, France

SERGE GRATTON
ENSEEIHT, IRIT, 2 Rue Charles Camichel, F-31071, Toulouse, France

AND

THEO MARY
Sorbonne Université, CNRS, LIP6, 4 Place Jussieu, F-75005, Paris, France
*Corresponding author: theo.beuzeville@toulouse-inp.fr

[Received on Date Month Year; revised on Date Month Year; accepted on Date Month Year]

The use of artificial neural networks is now becoming widespread across a wide variety of tasks. In this context of very rapid development, issues related to the storage and computational performance of these models emerge, since networks are sometimes very deep and comprise up to billions of parameters. For all these reasons, the use of reduced precision is increasingly being considered although, until now, its accuracy and robustness had been approached mostly from a practical standpoint or verified by software. The aim of this work is to provide formal tools to better understand, explain, and predict the accuracy and stability of neural networks when using floating-point arithmetic. To this end, we first extend to neural networks some well-known concepts from numerical linear algebra, such as condition number and backward error. We then apply a rounding error analysis based on existing tools in numerical linear algebra to obtain both forward and backward error bounds. This includes both deterministic worst-case bounds as well as probabilistic bounds that are sharper on average. These bounds both ensure the proper functioning of neural networks once trained, and provide recommendations on architectures and training methods to enhance the robustness of neural networks.

Keywords: floating-point arithmetic; error analysis; artificial neural networks; rounding errors; backward error; probabilistic error analysis.

1. Introduction

In the context of Artificial Intelligence (AI), where algorithms process vast amounts of data and execute complex computations in safety-critical tasks such as medicine with the detection of cancer [11] and self-driving cars [46], maintaining numerical accuracy is crucial also because it is often a prerequisite to ensure other properties such as explainability. Machine learning methods and their software implementations, even the most advanced ones, are however subject to rounding errors resulting from the use of finite precision arithmetic in calculations. The success of deep learning mostly stems from the ability of modern computing platforms to handle and optimize an increasingly large number of parameters in neural networks but also from the availability of larger and larger training datasets. This leads to higher energy and computational costs for the training and use of deep learning models which makes their use difficult when computing resources are limited such as in embedded devices. Therefore,

in modern machine learning, low-precision arithmetics are becoming increasingly attractive due to their higher speed and their lower memory and energy consumption [4, 23, 47]. Proposed solutions demonstrate significant enhancements in power consumption, processing speed, and memory usage by recommending the replacement of widely used 32-bit floating-point arithmetic by lower precision arithmetics.

Rounding errors can have a considerable impact on the robustness of AI methods and tools, where robustness is defined as the ability to maintain correct behaviour in the presence of disturbances. To be able to deploy neural networks in critical systems, we have to ensure that these errors do not modify their functioning, their operation and properties. Hence, the ability to measure the accuracy and stability of these systems, to analyse the errors due to rounding in the computations, and therefore to estimate which computations are more sensitive to changes of arithmetic is essential [10, 25]. The work on neural networks error analysis available in the literature is mainly based on experimental approaches [29, 35, 38] and very few studies address a more theoretical framework [36]. Software such as CADNA [18, 30] and FLUCTUAT [21] can be used to assess, with some precision, how reliable the result of an algorithm is. The extension of these types of verifier for artificial neural networks is still in progress [41] and raises many questions in terms of computation time as they are based on computationally expensive methods such as SMT (Satisfiability Modulo Theories) solving [32] or mixed integer linear programming [44], but also because of their own rounding errors [31].

The objective of this work is therefore to produce theoretical and experimental results enabling to understand the impact of rounding errors in neural networks architectures when using reduced precision floating-point arithmetic. To do so, we extend some error analysis concepts, which are commonly used in numerical linear algebra, to artificial neural networks.

In numerical analysis, *backward error* is a particularly well-established tool [25, 45], as it enables one to know if an inexact solution to a problem is in fact the exact solution to a nearby problem with slightly perturbed input data. Then depending on prior knowledge on the problem, such as uncertainty on the input data, one can say that said problem is backward stable if the backward error is close to these uncertainties which essentially means that the algorithm has computed a solution which is as good as it can be. Until now, the concept of backward error for artificial neural networks has only been partially explored, from the adversarial perspective [5, 7].

As a first contribution, this work develops generic definitions of the backward error and condition number for deep neural networks, as well as generic formulas for computing them numerically. This theoretical framework provides us with tools to quantify and better explain how generic perturbations affect artificial neural networks.

We then provide a deterministic and probabilistic rounding error analysis of artificial neural networks, which consists in obtaining bounds on backward and forward errors. These bounds provide a better understanding of how sensitive neural networks are to changes in arithmetic, depending on the choice of architecture, training, and scaling.

This work is organized as follows. Section 2 provides a background on rounding error analysis. In section 3 we first establish formulas and ways to compute the backward error of artificial neural networks; existing work focuses on numerical linear algebra, therefore our goal is to extend it by integrating activation functions, which induce nonlinearities. We also show how to compute the condition number, which, in turn, helps in establishing bounds on the forward error once we have determined bounds on the backward error. In section 4 we focus on producing a deterministic rounding error analysis of neural networks, therefore finding bounds on the backward and forward error. We then show how it can be extended to probabilistic bounds that are sharper in section 5. We then validate

these bounds experimentally in section 6 on both random and trained neural networks. Finally, the work concludes with a summary of the main findings and some perspectives of future works in section 7.

2. Background on rounding error analysis

This work will focus on the effects of the use of floating-point arithmetic on multilayered artificial neural networks. In this section, we recall key notions and results that will serve as the basis for our approach to quantify and predict the effects of rounding errors on artificial neural networks. We will use a componentwise analysis as this metric enables to take into account the structure of matrices, such as sparsity or scaling. Unlike normwise metrics, when using componentwise metrics, each element of a perturbation on the data is measured relatively to a given tolerance, which can for example be its absolute value. Therefore, divisions and inequalities between vectors in this work are meant componentwise.

2.1. *Backward error*

Let $\widehat{y}$ be a computed result that is an approximation of $y = f(x)$, with $f : \mathbb{R}^n \to \mathbb{R}^m$. When the forward error, defined for $y \neq 0$,

$$\varepsilon_{\text{fwd}}(\widehat{y}) = \max_i \frac{|\widehat{y} - y|_i}{|y|_i} \tag{2.1}$$

is large, we cannot distinguish if either the mathematical problem is sensitive to perturbations or the algorithm used to solve the problem behaves badly when perturbations exist on data or computations. The backward error is a quantity that makes it possible to discriminate between these two cases. Formally we can define the relative componentwise backward error as

$$\varepsilon_{\text{bwd}}(\widehat{y}) = \min\{\varepsilon : \widehat{y} = f(x + \Delta x),\ |\Delta x| \leq \varepsilon |x|\}. \tag{2.2}$$

It is then said that if there is an uncertainty in the data or computations (physical measurements, approximations, rounding errors...), it is sufficient that the backward error is of the same order as this uncertainty for the computed solution $\widehat{y}$ to be as good as one could expect.

2.2. *Condition number*

Forward error and backward error are linked by the condition number of the problem, which measures how sensitive the solution to a problem is to perturbations in the data. The use of condition number in numerical analysis is therefore particularly prominent [3, 14, 15, 24, 42] to understand and deploy strategies to diminish the impact of perturbations on a system. The condition number of the problem f at x is defined by Rice [39] and Lyubich [34], we have the relative componentwise condition number in

$$\kappa_f(x) = \lim_{\varepsilon \to 0} \sup_{\max_i |\Delta x_i| \leq \varepsilon |x_i|} \left(\max_i \frac{|f_i(x + \Delta x) - f_i(x)|}{|f_i(x)|} \Big/ \max_i \frac{|\Delta x_i|}{|x_i|} \right). \tag{2.3}$$

The link between forward, backward error and condition number is then given at first order by:

$$\text{forward error} \leq \text{condition number} \times \text{backward error}. \tag{2.4}$$

This relation is considered as a rule of thumb. When it holds for a set of problems, it is then used to predict the forward error on problems of the same class. In this work we will focus on providing a backward error analysis of artificial neural networks, deriving bounds on the backward error, which, in turn, lead to bounds on the forward error using inequality (2.4).

2.3. *Model of arithmetic*

In the context of floating-point arithmetic, since a finite number of bits is used to represent numbers, rounding errors occur during computations. Denoting $\mathrm{fl}(a \text{ op } b)$ the result of a floating-point elementary operation, we will use the following standard model for floating-point arithmetic:

Model 1 (Standard model for floating-point computations)

$$\mathrm{fl}(a \text{ op } b) = (a \text{ op } b)(1+\delta),\ |\delta| \leq u,\ \text{op} \in \{+,\ -,\ \times,\ /,\ \sqrt{}\},$$

where u is the *machine epsilon*, which is an upper bound on the relative approximation error due to rounding.

2.4. *Classical backward error analysis*

Model 1 bounds the error introduced by *one* floating-point operation. Traditional error analysis in numerical linear algebra typically involves sequences of several operations, which will therefore bring up products of the form

$$\prod_{k=1}^{n}(1+\delta_k).$$

These products are then simplified using the following lemma, which corresponds to Lemma 3.1 from Higham [25].

Lemma 1 (Deterministic error bound) *If $|\delta_k| \leq u$ and $\rho_k = \pm 1$ for $k = 1, \ldots, n$, and $nu < 1$, then*

$$\prod_{k=1}^{n}(1+\delta_k)^{\rho_k} = 1+\theta_n,\ |\theta_n| \leq \gamma_n.$$

The constant γ_n is defined as

$$\gamma_n = \frac{nu}{1-nu}, \tag{2.5}$$

with $nu < 1$, n being the number of elementary operations considered, and u the machine epsilon [25].

2.5. *Probabilistic backward error analysis*

For large problems or computations in reduced precision, bounds obtained using classical rounding error analysis, which are worst-case bounds, may be less useful because too pessimistic. Indeed, these traditional bounds involve the number n of elementary operations performed. Recent approaches [12, 26, 27] enable to relax this constant by replacing n by $\sqrt{n}$, based on probabilistic assumptions about the rounding errors. We will later derive bounds for artificial neural networks, which are based on the results from Connolly et al. [12]. Their analysis is based on some probabilistic tools defined below. In the following, $\mathbb{E}(X)$ denotes the expectation of a random variable X and $\mathbb{E}(X \mid Y)$ denotes the conditional expectation of X given Y.

Definition 1 (Martingale) *A sequence of random variables $E_0, \ldots, E_n$ is said to be a martingale if it satisfies for all k*

$$\mathbb{E}(|E_k|) < +\infty,$$
$$\mathbb{E}(E_k \mid E_0, \ldots, E_{k-1}) = E_{k-1}.$$

Lemma 2 (Azuma–Hoeffding inequality) *If $E_0,\dots,E_n$ is a martingale that satisfies $|E_k - E_{k-1}| \le c_k$ for $k = 1,\dots,n$, then for all $\lambda > 0$,*

$$\Pr\left(|E_n - E_0| \ge \lambda \left(\sum_{k=1}^{n} c_k^2\right)^{\frac{1}{2}}\right) \le 2\exp\left(\frac{-\lambda^2}{2}\right).$$

Connolly et al. [12] use the following model of rounding errors.

Model 2 (Probabilistic model of rounding errors) *Let $\delta_1,\dots,\delta_n$ be random variables of mean zero with $|\delta_k| \le u$ such that $\mathbb{E}(\delta_{k+1} \mid \delta_1,\dots,\delta_k) = \mathbb{E}(\delta_{k+1})$ for any $k = 1,\dots,n-1$.*

Define

$$\widetilde{\gamma}_n(\lambda) = \exp\left(\frac{\lambda\sqrt{n}u + nu^2}{1-u}\right) - 1 = \lambda\sqrt{n}u + O(u^2). \tag{2.6}$$

The next result of Connolly et al. [12] proves that the deterministic constant γ_n can be replaced, with high probability, by this relaxed constant $\widetilde{\gamma}_n(\lambda)$ provided that the probabilistic Model 2 holds.

Theorem 1 (Probabilistic error bound) *Let $\delta_1,\dots,\delta_n$ satisfy Model 2, then for $\rho_i = \pm 1$, $i = 1,\dots,n$ and any constant $\lambda > 0$,*

$$\prod_{i=1}^{n}(1+\delta_i)^{\rho_i} = 1 + \theta_n,\ |\theta_n| \le \widetilde{\gamma}_n(\lambda)$$

holds with probability at least

$$P(\lambda) = 1 - 2\exp\left(\frac{-\lambda^2}{2}\right).$$

Note that another important result of Connolly et al. [12] is that Model 2 holds when stochastic rounding [13] is used.

3. Backward error and condition number for artificial neural networks

First introduced by Beuzeville et al. [7], then built upon by Beerens and Higham [5] and further investigated in Savostianova et al. [40], the concepts of backward error and condition number of neural networks are the subject of a rising interest to better understand and explain the sensitivity of neural networks to perturbations. The backward error represents the solution to a minimization problem, which often means that closed formulas are not readily derived. On the other hand, the condition number can be used to bound the forward error and will thus be of special interest in the context of rounding error analysis. The goal of this section is therefore to explain how to establish explicit expressions of the backward error as well as the condition number for artificial neural networks.

3.1. *Neural network expression*

Consider a feed-forward network of depth $p \in \mathbb{N}$ layers, each layer with its associated weight matrix $A_i \in \mathbb{R}^{n_i \times n_{i-1}}$ and activation function ϕ_i applied entrywise that we will suppose differentiable. In the machine learning context, activation functions are typically chosen so that they are differentiable, since their gradient is necessary for the training phase. Several methods already exist for dealing with functions

that are not differentiable at certain points, since gradients are needed during the training phase. The most widely known example is ReLU, in which case one typically sets the value of the derivative to zero at zero [6].

Note that a bias can easily be integrated as part of the matrix–vector product and thus, without loss of generality, we will assume that one layer of the neural network corresponds to a matrix–vector product and drop the bias for the sake of readability. For a given input $x \in \mathbb{R}^{n_0}$ we then have the following expression for the output of this neural network:

$$y = m(A,x) = \phi_p(A_p\phi_{p-1}(A_{p-1}\dots A_2\phi_1(A_1x)\dots)). \tag{3.1}$$

Our work focuses on producing a general theoretical framework to evaluate a neural network's sensibility to rounding errors. In order to get generic formulas, we will consider perturbations on the parameters $(A_i)_{i=1,\dots,p}$ and input x. As shown in section 2 rounding errors are typically proportional to the machine epsilon, we will thus use a first order approximation of the model with respect to the perturbed parameters.

Consider a given neural network model, assuming perturbations on the model's parameters and on its input we have

$$\widehat{y} = \phi_p((A_p+\Delta A_p)\phi_{p-1}((A_{p-1}+\Delta A_{p-1})\dots\phi_1((A_1+\Delta A_1)(x+\Delta x))\dots)).$$

A first order approximation leads to the following equality:

$$\begin{aligned}\widehat{y}-y = {}& \phi_p'(A_py_{p-1})\Delta A_py_{p-1}+\dots\\ &+\phi_p'(A_py_{p-1})A_p\phi_{p-1}'(A_{p-1}y_{p-2})\dots A_{i+1}\phi_i'(A_iy_{i-1})\Delta A_iy_{i-1}\\ &+\dots+\phi_p'(A_py_{p-1})A_p\phi_{p-1}'(A_{p-1}y_{p-2})\dots A_2\phi_1'(A_1x)\Delta A_1x\\ &+\phi_p'(A_py_{p-1})A_p\phi_{p-1}'(A_{p-1}y_{p-2})\dots A_2\phi_1'(A_1x)A_1\Delta x.\end{aligned} \tag{3.2}$$

For the sake of readability let us define, for $i = 1,\dots,p$, the Jacobian of our neural network model m computed with respect to the parameters A_i

$$J_m^i(A,x) = \phi_p'(A_py_{p-1})A_p\phi_{p-1}'(A_{p-1}y_{p-2})\dots A_{i+1}\phi_i'(A_iy_{i-1})$$

and the Jacobian of the model with respect to the input

$$J_m^0(A,x) = \phi_p'(A_py_{p-1})A_p\phi_{p-1}'(A_{p-1}y_{p-2})\dots A_2\phi_1'(A_1x)A_1.$$

Equation (3.2) can thus be rewritten

$$\widehat{y}-y = \sum_{i=1}^{p} J_m^i(A,x)\Delta A_iy_{i-1} + J_m^0(A,x)\Delta x. \tag{3.3}$$

This expression can then be rearranged, using the Kronecker product denoted by $\bigotimes$, to have the form of a linear system. Indeed, since for any given matrices A and X and any given vector b we have

$AXb = (b^T \bigotimes A)\overrightarrow{X}$, where $\overrightarrow{X}$ is the vectorization operator applied on X, we can then say that

$$\widehat{y} = y + \sum_{i=1}^{p} (y_{i-1}^T \bigotimes J_m^i(A,x))\overrightarrow{\Delta A}_i + J_m^0(A,x)\Delta x. \tag{3.4}$$

Let us then define the vector $\overrightarrow{\Delta A}$ as the concatenation of all the vectorized perturbations

$$\overrightarrow{\Delta A} = \begin{bmatrix} \Delta x \\ \overrightarrow{\Delta A}_1 \\ \vdots \\ \overrightarrow{\Delta A}_p \end{bmatrix} \tag{3.5}$$

and the Jacobian matrix of our model with respect to the input and parameters

$$\mathscr{J}_m(A,x) = \left[J_m^0(A,x), \quad y_0^T \bigotimes J_m^1(A,x), \quad \ldots, \quad y_{p-1}^T \bigotimes J_m^p(A,x) \right]. \tag{3.6}$$

We then can then rewrite equation (3.4) as the following linear system

$$\widehat{y} = y + \mathscr{J}_m(A,x)\overrightarrow{\Delta A}. \tag{3.7}$$

3.2. *Backward error expression*

We can define, using the above first order approximation, the componentwise relative backward error as:

$$\varepsilon_{\text{bwd}} = \min\{\varepsilon \geq 0 : \widehat{y} = y + \mathscr{J}_m(A,x)\overrightarrow{\Delta A},\ |\overrightarrow{\Delta A}| \leq \varepsilon |\overrightarrow{A}|\}, \tag{3.8}$$

where $\mathscr{J}_m(A,x) \in \mathbb{R}^{M \times N}$. Since we defined the sizes of the weight matrices and of the output as in equation (3.1), we have $M = n_p$, $N = \sum_{i=1}^{p} n_i \times n_{i-1} + n_0$ and therefore the system $\widehat{y} - y = \mathscr{J}_m(A,x)\overrightarrow{\Delta A}$ is underdetermined. We hence have no closed formula to compute the componentwise backward error for a general neural network. In that case, finding the backward error defined as in equation (3.8) is equivalent to solving the following optimization problem

$$\begin{aligned} &\underset{\overrightarrow{\Delta A}}{\text{minimize}} \quad \|\overrightarrow{\Delta A}\|_\infty \\ &\text{subject to} \quad \widehat{y} - y = \mathscr{J}_m(A,x)\overrightarrow{\Delta A}. \end{aligned} \tag{3.9}$$

Several methods [1, 9, 17, 43] focus on solving problems that are similar in form to the problem (3.9).

3.3. *Condition number expression*

Consider a given neural network model, assuming perturbations on the model's parameters and on its input, we can define the vector $\overrightarrow{\Delta A}$ as in equation (3.5) and the Jacobian matrix $\mathscr{J}_m(A,x)$ as in equation (3.6). This leads to the first order expression of equation (3.7). We can then define the relative

componentwise condition number in the same way as Gohberg and Koltracht [20], which gives the following expression

$$\kappa_m(A,x) = \|\mathrm{diag}(m(A,x))^{-1}\mathscr{J}_m(A,x)\mathrm{diag}(\overrightarrow{A})\|_\infty.$$

Note that, depending on which variables are considered to be perturbed, the expression of the Jacobian matrix and the vector containing the perturbed variables varies. Therefore, in order to have the following relation between forward error, backward error; and condition number,

$$\varepsilon_{\mathrm{fwd}} \leq \kappa_m(A,x)\varepsilon_{\mathrm{bwd}},$$

one must ensure that quantities are defined with the same appropriate metrics and perturbations.

4. Deterministic rounding error analysis for artificial neural networks

This section presents a rounding error analysis for artificial neural networks. The goal of such an analysis is to provide bounds on the backward error in order to explain and quantify how the use of a given arithmetic precision impacts the accuracy and stability of an algorithm. Using the backward error over the forward error provides several advantages. Indeed, by focusing on backward error, the analysis can offer insights into the root causes of instabilities in the neural network's predictions and behaviours. Moreover, once bounds on the backward error are found, bounds on the forward error can be directly derived using the condition number of the problem.

The primary contribution of this section will be to explain in section 4.1 how to integrate the nonlinear activation functions into the deterministic analysis of rounding errors. This requires understanding how rounding errors produced by the computation of the function can be interpreted as errors on the input of the function. Subsequently, we will integrate this analysis to obtain bounds for the computations of a single layer neural network in section 4.2 and this will finally lead us to generalize these bounds to the case of deeper neural networks in section 4.3.

4.1. *Activation function*

Here and for the remainder of this document we will assume that for all activation functions, similarly to Model 1, the following model stands:

Model 3 (Floating-point arithmetic model for activation functions)

$$\mathrm{fl}(f(x)) = f(x)(1+\delta_f),\ |\delta_f| \leq \ell u.$$

Knowing that for each activation function, the constant ℓ has to be evaluated for each framework. As an example, for NVIDIA GPUs (Graphics Processing Units) the constant for activation functions such as tanh or ReLU can be found on the Appendix E of the documentation [37].

To integrate activation functions into the backward error analysis, we need to know how the rounding error obtained by applying the function, according to Model 3, can be interpreted back as a perturbation on the data.

Let us define $f : \mathbb{R}^n \to \mathbb{R}^n$. Assuming that f is differentiable at the point x and that $f(x) \neq 0$, at first order, we have

$$f(x+\delta) = f(x) + \delta f'(x) = f(x)(1+\delta\frac{f'(x)}{f(x)}).$$

Let us define

$$K_f(x) = \frac{xf'(x)}{f(x)}$$

and note $\kappa_f(x) = |K_f(x)|$, which represents the componentwise condition number of f at x, then

$$f(x+\delta) = f(x)(1+\frac{\delta}{x}K_f(x)). \tag{4.1}$$

Here we want to use equation (4.1) to understand the impact of rounding errors, in terms of perturbations on the input data x, when Model 3 stands. Let f be a given activation function, differentiable at x, that satisfies this model. The computed solution $\widehat{y}$ then satisfies

$$\widehat{y} = f(x)(1+\delta_f) = f(x)(1+\frac{\delta_f}{x}K_f(x)\frac{x}{K_f(x)}).$$

Replacing δ in equation (4.1) by

$$\Delta x = \delta_f \frac{x}{K_f(x)},$$

we have

$$\widehat{y} = f(x)(1+\delta_f) = f(x+\delta_f\frac{x}{K_f(x)}) = f(x+\Delta x),$$

with

$$|\Delta x| \leq \frac{\ell u}{\kappa_f(x)}|x|.$$

This result shows that the relative perturbation on the input that is needed to get the computed output $\widehat{y}$ is bounded by $\ell u/\kappa_f(x)$. Therefore, the perturbations are small when $\kappa_f(x) \geq \ell u$. This states that the bigger the condition number of f is, the smaller the perturbation needed on the input will be to attain a same output. This is consistent with the fact that functions with small condition number are less sensitive to perturbations.

4.2. *Entire layer*

We now seek to combine bounds usually obtained for the matrix–vector product to the results of section 4.1 to obtain rounding error bounds for a complete layer of artificial neural network.

Consider an entire layer composed of a matrix–vector product followed by an activation function, let $y = \phi(Ax)$ with x a given input vector. Each output component $\widehat{y}_i$ then satisfies:

$$\begin{aligned}\widehat{y}_i &= \phi\left(\sum_{k=1}^{n}(a_{ik}x_k)(1+\varepsilon_k)\prod_{j=\max(k,2)}^{n}(1+\delta_j)\right)(1+\delta_\phi)\\ &= \phi\left(\sum_{k=1}^{n}(a_{ik}x_k)(1+\psi_k)\right)(1+\delta_\phi)\end{aligned}$$

where

$$\psi_k = (1+\varepsilon_k)\prod_{j=\max(k,2)}^{n}(1+\delta_j)$$

accounts for the n rounding errors in the matrix–vector product, and δ_ϕ accounts for those in the activation function. Note that ψ_k and δ_ϕ also depend on i but are not indexed to simplify the notation.

From section 4.1 we know how to take into account rounding errors introduced by the activation function in terms of perturbations on the input data, which yields

$$\widehat{y}_i = \phi\left(\sum_{k=1}^{n}(a_{ik}x_k)(1+\psi_k)\left(1+\frac{\delta_\phi}{K_\phi(a_i^T x)}\right)\right) \tag{4.2}$$

$$= \phi\left(\sum_{k=1}^{n}(a_{ik}x_k)(1+\Phi_k)\right) \tag{4.3}$$

where

$$\Phi_k = \psi_k + \frac{\delta_\phi}{K_\phi(a_i^T x)} + \frac{\psi_k\delta_\phi}{K_\phi(a_i^T x)}.$$

Our goal is then to bound the absolute value of Φ_k. From Lemma 1 we have $|\psi_k| \le \gamma_n$, and from Model 3 we have $|\delta_\phi| \le \ell u$. Therefore we obtain

$$|\Phi_k| = \left|\psi_k + \frac{\delta_\phi}{K_\phi(a_i^T x)} + \frac{\psi_k\delta_\phi}{K_\phi(a_i^T x)}\right| \le \frac{nu}{1-nu} + \frac{\ell u}{\kappa_\phi(a_i^T x)} + \frac{\ell nu^2}{\kappa_\phi(a_i^T x)(1-nu)}.$$

Gathering all terms under the same denominator leads to

$$|\Phi_k| \le \frac{\kappa_\phi(a_i^T x)nu + \ell u(1-nu) + \ell nu^2}{\kappa_\phi(a_i^T x)(1-nu)},$$

which means that

$$|\Phi_k| \le \frac{(n+\frac{\ell}{\kappa_\phi(a_i^T x)})u}{1-nu},$$

which can finally be further weakened to

$$|\Phi_k| \le \frac{(n+\frac{\ell}{\kappa_\phi(a_i^T x)})u}{1-(n+\frac{\ell}{\kappa_\phi(a_i^T x)})u} = \gamma_{n+\ell/\kappa_\phi(a_i^T x)}.$$

Combining equation (4.3) for the m rows of A yields

$$\widehat{y} = \phi((A+\Delta A)x),\ |\Delta A| \leq \gamma_{n+\ell/\kappa_\phi(Ax)}|A|. \tag{4.4}$$

Note that the inequality is to be taken componentwise, where $\kappa_\phi(Ax)$ is a vector whose i-th component is equal to $\kappa_\phi(a_i^T x)$. This result shows that each relative perturbation on the input is bounded by $\gamma_{n+\ell/\kappa_\phi(Ax)}$. Hence, the perturbations are small when

$$\left(n + \frac{\ell}{\kappa_\phi(Ax)}\right) u \ll 1.$$

This can be interpreted as a combination of the known results on the matrix–vector product and the obtained result of section 4.1. The bounds reflect the fact that n basic operations are done by the matrix–vector product and then the ℓ errors due to the activation are amplified or reduced depending on its condition number.

4.3. *Multi-layer neural network*

We have obtained in the previous section bounds on the backward error for a single layer of artificial neural network; we now consider the case of a general neural network of p layers.

In order to better explain our approach, let us first develop how rounding error propagates on a couple of layers. Let $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{p\times m}$ and $x \in \mathbb{R}^n$. Assuming that we compute $y = \phi_2(B\phi_1(Ax))$ by first performing $z = \phi_1(Ax)$ and then $y = \phi_2(Bz)$, we thus have two layer applications in a row. Applying equation (4.4) on the first layer yields

$$\widehat{z} = \phi_1((A+\Delta A)x),\ |\Delta A| \leq \gamma_{n+\ell_{\phi_1}/\kappa_{\phi_1}(Ax)}|A|.$$

and applying it to the second layer yields

$$\widehat{y} = \phi_2((B+\Delta B)\widehat{z}),\ |\Delta B| \leq \gamma_{m+\ell_{\phi_2}/\kappa_{\phi_2}(B\widehat{z})}|B|.$$

Hence, overall,

$$\widehat{y} = \phi_2((B+\Delta B)\phi_1((A+\Delta A)x)),$$

with

$$|\Delta A| \leq \gamma_{n+\ell_{\phi_1}/\kappa_{\phi_1}(Ax)}|A|, \qquad |\Delta B| \leq \gamma_{m+\ell_{\phi_2}/\kappa_{\phi_2}(B\widehat{z})}|B|.$$

This result reveals a pattern for bounds on the backward error when chaining multiple layers. Indeed, for a given layer, the bound combines, first the aggregation of errors due to the matrix–vector product, in the form of the number of columns of the weight matrix n_{i-1}, and then the impact of the activation function. Through induction, we can extend the previous result to a network with multiple layers, which leads to the following key result.

Theorem 2 (Deterministic error bound for artificial neural networks) *Consider a neural network composed of p layers whose output can be expressed as follows* $y = \phi_p(A_p\phi_{p-1}(A_{p-1}\ldots A_2\phi_1(A_1x)\ldots))$,

with $A_i \in \mathbb{R}^{n_i \times n_{i-1}}$, for $i = 1, \dots, p$ and $x \in \mathbb{R}^{n_0}$. If y is evaluated in floating-point arithmetic satisfying Model 1 and Model 3, the computed result $\widehat{y}$ satisfies

$$\widehat{y} = \phi_p((A_p + \Delta A_p)\phi_{p-1}((A_{p-1} + \Delta A_{p-1}) \dots (A_2 + \Delta A_2)\phi_1((A_1 + \Delta A_1)x) \dots)),$$

with

$$|\Delta A_i| \leq \gamma_{n_{i-1}+\ell_{\phi_i}/\kappa_{\phi_i}(A_i \widehat{y}_{i-1})} |A_i|, \ i = 1, \dots, p.$$

This bound is obtained by using both the standard model of arithmetic, given by Model 1, for the matrix–vector computations, and the Model 3 for the computation of the activation function. The bound quantifies the impact of activation functions in the context of computations performed by neural networks in finite precision. This impact appears in the form of the constant ℓ_{ϕ_i}, which, for each activation function ϕ_i, corresponds to the rounding errors introduced by the application of the function, and this constant is then amplified or not depending on the condition number of the activation.

5. Probabilistic rounding error analysis for artificial neural networks

The goal of this section is to show how the bounds obtained on artificial neural networks in section 4.3 can be adapted to the probabilistic case. We will therefore focus on integrating activation functions in the probabilistic setting. However, since activation functions are not standard numerical linear algebra operations, as shown in section 4.2, we will demonstrate how this changes the approach and results we had in Theorem 2. Two different ways of integrating activation functions in the probabilistic error analysis can be distinguished. In section 5.1 we first make probabilistic assumptions only on the errors due to linear algebra operations, such as matrix–vector product, and use only deterministic assumptions for the errors due to the computation of the activation function. In section 5.2 we extend our probabilistic model to also cover the activation functions. Finally in section 5.3 we summarize the different bounds that we have obtained and discuss their significance.

As shown in section 4.2, the result of a neural network layer followed by an activation function results in a product of rounding error terms of the following form, seen in equation (4.2):

$$\prod_{i=1}^{n}(1 + \delta_i)\left(1 + \frac{\delta_{n+1}}{K_\phi(\widehat{c})}\right), \tag{5.1}$$

where $\widehat{c}$ is the computed output of a matrix–vector product and $\delta_1, \dots, \delta_n$ its associated rounding errors.

Since under Model 2, the rounding errors $\delta_1, \dots, \delta_n$ are random variables, then $\widehat{c}$, which is a function of these rounding errors, is also a random variable. Then the value of $\widehat{c}$ and thus of $\kappa_\phi(\widehat{c})$ will fluctuate with respect to the values of the rounding errors. In the following, we assume that there exists a $\zeta > 0$ such that $\kappa_\phi(\widehat{c}) \geq \zeta$, that is, that despite the randomness $\kappa_\phi(\widehat{c})$ never becomes too close to zero. For the sake of readability, let us note for the remainder of this section $K_\phi = K_\phi(\widehat{c})$ and $\kappa_\phi = |K_\phi(\widehat{c})| = \kappa_\phi(\widehat{c})$.

Theorem 1 can be used to directly replace the deterministic γ_n with its probabilistic equivalent $\widetilde{\gamma}_n(\lambda)$ for products of $(1 + \delta_i)$ terms. The goal of this section is therefore extend this result to the particular case of a product of rounding errors of the form of equation (5.1).

5.1. *Deterministic activation function's rounding error*

We first combine the probabilistic bound on the matrix–vector product error with the deterministic bound on the activation error. This leads to the following mixed error bound for equation (5.1).

Lemma 3 *Let $\delta_1,\dots,\delta_n$ be random variables of mean zero with $|\delta_k| \le u$ for all k such that $\mathbb{E}(\delta_{k+1} \mid \delta_1,\dots,\delta_k) = \mathbb{E}(\delta_{k+1}) = 0$ for $k = 1,\dots,n-1$. Let $|\delta_{n+1}| \le \ell u$. Then for any constant $\lambda > 0$,*

$$\prod_{i=1}^{n}(1+\delta_i)\left(1+\frac{\delta_{n+1}}{K_\phi(x)}\right) = 1+\Psi_n,$$

$$|\Psi_n| \le \widetilde{\gamma}_n(\lambda) + \frac{\ell}{\kappa_\phi(x)}u(1+\widetilde{\gamma}_n(\lambda))$$

holds with probability at least $1-2\exp\left(-\lambda^2/2\right)$.

Proof Denoting $1+\theta_n = \prod_{i=1}^{n}(1+\delta_i)$ we have

$$\Psi_n = \theta_n + \frac{\delta_{n+1}}{K_\phi(x)} + \theta_n\frac{\delta_{n+1}}{K_\phi(x)}$$

and so

$$|\Psi_n| \le |\theta_n| + \left|\frac{\delta_{n+1}}{K_\phi(x)}\right| + \left|\theta_n\frac{\delta_{n+1}}{K_\phi(x)}\right|.$$

Using Theorem 1,

$$|\Psi_n| \le \widetilde{\gamma}_n(\lambda) + \frac{\ell u}{\kappa_\phi(x)}(1+\widetilde{\gamma}_n(\lambda)),$$

holds with probability at least $1-2\exp\left(-\lambda^2/2\right)$. □

Using Lemma 3 on the expression in equation (4.2), we obtain the following mixed error bound for a general artificial neural network.

Theorem 3 (Mixed error bound for artificial neural networks) *Consider a general neural network composed of p layers whose output can be expressed as follows $y = \phi_p(A_p\phi_{p-1}(A_{p-1}\dots A_2\phi_1(A_1x)\dots))$, with $A_i \in \mathbb{R}^{n_i\times n_{i-1}}$, for $i = 1,\dots,p$ and $x \in \mathbb{R}^{n_0}$. If the computation of y in floating-point arithmetic generates rounding errors in the matrix–vector product that satisfy Model 2 and rounding errors in the activation functions that satisfy Model 3, then the computed result $\widehat{y}$ satisfies*

$$\widehat{y} = \phi_p((A_p+\Delta A_p)\phi_{p-1}((A_{p-1}+\Delta A_{p-1})\dots(A_2+\Delta A_2)\phi_1((A_1+\Delta A_1)x)\dots))$$

with

$$|\Delta A_i| \le \left(\widetilde{\gamma}_{n_{i-1}}(\lambda) + \frac{\ell_{\phi_i}u}{\kappa_{\phi_i}(A_i\widehat{y}_{i-1})}(1+\widetilde{\gamma}_{n_{i-1}}(\lambda))\right)|A_i|,\ i = 1,\dots,p,$$

with probability at least

$$Q(\lambda, \sum_{i=1}^{p} n_i n_{i-1}),$$

where $Q(\lambda,n) = 1-n(1-P(\lambda))$.

Proof The proof to obtain the bounds is almost identical to the work of section 4.3, replacing Lemma 1 by Lemma 3. For a given layer i the bound holds with probability at least $Q(\lambda, n_i n_{i-1})$, by the same logic as in [12]. Therefore, the bound fails to hold for a given i with probability at most $1-Q(\lambda, n_i n_{i-1})$, hence it fails to hold for at least one layer i with probability at most $\sum_{i=1}^{p}(1-Q(\lambda, n_i n_{i-1})))$. This means that the bound holds for any layer with probability at least

$$1-\left(\sum_{i=1}^{p} n_i n_{i-1}(1-P(\lambda))\right) = 1-(1-P(\lambda))\sum_{i=1}^{p} n_i n_{i-1} = Q(\lambda, \sum_{i=1}^{p} n_i n_{i-1}).$$

□

This result, which combines a probabilistic approach for errors arising from the matrix–vector product and a deterministic approach for the accumulation of these errors with those of the activation function, logically yields a bound reflecting the approach. Indeed, we obtain a bound that adds to the usual bound on the matrix–vector product, $\widetilde{\gamma}_n(\lambda)$, a second term which reflects the addition of the errors coming from the activation function, ℓu, which are amplified or not depending on the value of the condition number.

Compared to the deterministic approach of Theorem 2, this approach only adds the assumptions of the standard probabilistic Model 2 for the computations of the matrix–vector product (as mentioned in section 2.5, these assumptions are notably satisfied if stochastic rounding is used in the matrix–vector product [12]).

5.2. *Probabilistic activation function's rounding error*

We now seek to further refine the previous analysis by integrating the activation function into our probabilistic model. To do so, we need to make assumptions about the ℓ rounding errors in the term δ_{n+1}. In general, the intermediate ℓ errors cannot be assumed to be mean independent between each other without any specific information on how the activation function is computed. Therefore, we will only assume that the global rounding error δ_{n+1}, which encompasses all ℓ intermediate errors, is of mean zero and is mean independent of the errors $\delta_1, \dots, \delta_n$ incurred in the matrix–vector product. These additional probabilistic assumptions on the error δ_{n+1} lead to the following model, which extends Model 2.

Model 4 *Let $\delta_1, \dots, \delta_{n+1}$ be random variables of mean zero with $|\delta_k| \leq u$ for all $k = 1, \dots, n$ and $|\delta_{n+1}| \leq \ell u$, such that $\mathbb{E}(\delta_{k+1} \mid \delta_1, \dots, \delta_k) = \mathbb{E}(\delta_{k+1}) = 0$ for $k = 1, \dots, n$.*

The goal will now be to adapt the proof of [12, Theorem 4.6] (which corresponds to Theorem 1 in this paper) to this new model.

Lemma 4 *Let $\delta_1, \dots, \delta_{n+1}$ be random variables that satisfy Model 4. Let K_ϕ be a function of $\delta_1, \dots, \delta_n$ and assume that there exists $\zeta > 0$ such that $\kappa_\phi \geq \zeta$. Let $E_k = \sum_{i=1}^{k} \delta_i$ for $k = 1, \dots, n$, $E_0 = 0$ and*

$$E_{n+1} = \sum_{i=1}^{n} \delta_i + \frac{\delta_{n+1}}{K_\phi}.$$

Then for any constant $\lambda > 0$*,*

$$|E_{n+1}| \leq \lambda \sqrt{n + \frac{\ell^2}{\zeta^2} u}$$

holds with probability at least $1 - 2\exp\left(-\lambda^2/2\right)$.

Proof Since $|\delta_k| \leq u$ for $k = 1, \ldots, n$, we have $|E_k| \leq ku$, and since $|\delta_{n+1}| \leq \ell u$ and $\zeta \leq |K_\phi| = \kappa_\phi$, we have

$$|E_{n+1}| \leq (n + \frac{\ell}{\zeta})u.$$

Hence $\mathbb{E}(|E_k|) < +\infty$ for all $k = 1, \ldots, n+1$. Moreover, for $k = 1, \ldots, n-1$,

$$\mathbb{E}(E_{k+1} \mid E_1, \ldots, E_k) = E_k + \mathbb{E}(\delta_{k+1} \mid \delta_1, \ldots, \delta_k) = E_k.$$

We also have

$$\mathbb{E}(E_{n+1} \mid E_1, \ldots, E_n) = E_n + \mathbb{E}(\frac{\delta_{n+1}}{K_\phi} \mid \delta_1, \ldots, \delta_n).$$

Since K_ϕ depends only on $\delta_1, \ldots, \delta_n$, K_ϕ is then fixed when $\delta_1, \ldots, \delta_n$ are fixed, therefore

$$\mathbb{E}(\frac{\delta_{n+1}}{K_\phi} \mid \delta_1, \ldots, \delta_n) = \frac{\mathbb{E}(\delta_{n+1} \mid \delta_1, \ldots, \delta_n)}{K_\phi} = 0,$$

which implies that $\mathbb{E}(E_{n+1} \mid E_1, \ldots, E_n) = E_n$ and hence $E_0, \ldots, E_{n+1}$ is a martingale. Moreover, $|E_{k+1} - E_k| \leq u$ for $k = 1, \ldots, n-1$ and

$$|E_{n+1} - E_n| \leq \frac{\ell u}{\zeta}.$$

By the Azuma–Hoeffding inequality, given by Lemma 2, we therefore have for any $\lambda > 0$,

$$\Pr\left(|E_{n+1} - E_0| \geq \lambda \left(nu^2 + \frac{\ell^2 u^2}{\zeta^2}\right)^{\frac{1}{2}}\right) \leq 2\exp\left(\frac{-\lambda^2}{2}\right)$$

which means that

$$\Pr\left(|E_{n+1}| \geq \lambda \sqrt{n + \frac{\ell^2}{\zeta^2} u}\right) \leq 2\exp\left(\frac{-\lambda^2}{2}\right)$$

and concludes the proof. □

We are now ready to derive a fully probabilistic error bound for equation (5.1).

Lemma 5 *Let $\delta_1, \dots, \delta_{n+1}$ be random variables that satisfy Model 4. Let K_ϕ be a function of $\delta_1, \dots, \delta_n$ and assume that there exists $\zeta > 0$ such that $\kappa_\phi \geq \zeta$. Then for any constant $\lambda > 0$,*

$$\prod_{i=1}^{n}(1+\delta_i)(1+\frac{\delta_{n+1}}{K_\phi}) = 1+\theta_n, \quad |\theta_n| \leq \widetilde{\gamma}_n^{\text{act}}(\lambda)$$

holds with probability at least $1-2\exp\left(-\lambda^2/2\right)$, where

$$\widetilde{\gamma}_n^{\text{act}}(\lambda) = \exp\left(\lambda\sqrt{n+\frac{\ell^2}{\zeta^2}}u + \frac{nu^2}{1-u} + \frac{\left(\frac{\ell u}{\zeta}\right)^2}{1-\frac{\ell u}{\zeta}}\right) - 1 \tag{5.2}$$

$$= \lambda\sqrt{n+\frac{\ell^2}{\zeta^2}}u + O(u^2)$$

extends the definition of $\widetilde{\gamma}_n(\lambda)$ to include the activation error.

Proof Let $E_k = \sum_{i=1}^{k} \delta_i$ for $k = 1, \dots, n$, $E_0 = 0$ and

$$E_{n+1} = \sum_{i=1}^{k} \delta_i + \frac{\delta_{n+1}}{K_\phi}.$$

From Lemma 4 we know that

$$|E_{n+1}| \leq \lambda\sqrt{n+\frac{\ell^2}{\zeta^2}}u,$$

holds with probability at least $1-2\exp\left(-\lambda^2/2\right)$.

We now will use the bound we found for E_{n+1} to bound the product of rounding error terms of equation (5.1). By taking the logarithm of this product we have

$$\log\left(\prod_{i=1}^{n}(1+\delta_i)\left(1+\frac{\delta_{n+1}}{K_\phi}\right)\right) = \sum_{i=1}^{n}\log(1+\delta_i) + \log\left(1+\frac{\delta_{n+1}}{K_\phi}\right).$$

Using the Taylor expansion of $\log(1+\delta_i)$, since $|\delta_i| \leq u < 1$, we have

$$-\delta_i - \sum_{k=2}^{+\infty}\frac{\delta_i^k}{k} \leq \log(1+\delta_i) \leq \delta_i + \sum_{k=2}^{+\infty}\frac{\delta_i^k}{k}.$$

These bounds can be further weakened to

$$-\delta_i - \sum_{k=2}^{+\infty}\delta_i^k \leq \log(1+\delta_i) \leq \delta_i + \sum_{k=2}^{+\infty}\delta_i^k$$

and then, taking the closed form of these geometric series, we have

$$-\delta_i - \frac{|\delta_i|^2}{1-|\delta_i|} \leq \log(1+\delta_i) \leq \delta_i + \frac{|\delta_i|^2}{1-|\delta_i|}, \tag{5.3}$$

which then implies, since $|\delta_i| \leq u$, that

$$-\delta_i - \frac{u^2}{1-u} \leq \log(1+\delta_i) \leq \delta_i + \frac{u^2}{1-u},$$

which, by adding the inequalities for $i = 1, \ldots, n$, then means

$$-E_n - \frac{nu^2}{1-u} \leq \sum_{i=1}^{n} \log(1+\delta_i) \leq E_n + \frac{nu^2}{1-u}. \tag{5.4}$$

At this point equation (5.4) enables us to obtain bounds on the error terms coming from the n first rounding errors, we now need to incorporate the error term that comes from the activation function. Assuming that $\ell u/\zeta < 1$, we have

$$\log(1+\frac{\delta_{n+1}}{K_\phi}) = \sum_{k=1}^{+\infty} (-1)^{k+1} \frac{\delta_{n+1}^k}{kK_\phi^k}$$

and hence, as in equation (5.3), we get

$$-\frac{\delta_{n+1}}{K_\phi} - \frac{|\frac{\delta_{n+1}}{K_\phi}|^2}{1-|\frac{\delta_{n+1}}{K_\phi}|} \leq \log(1+\frac{\delta_{n+1}}{K_\phi}) \leq \frac{\delta_{n+1}}{K_\phi} + \frac{|\frac{\delta_{n+1}}{K_\phi}|^2}{1-|\frac{\delta_{n+1}}{K_\phi}|}$$

which implies

$$-\frac{\delta_{n+1}}{K_\phi} - \frac{\left(\frac{\ell u}{\zeta}\right)^2}{1-\frac{\ell u}{\zeta}} \leq \log(1+\frac{\delta_{n+1}}{K_\phi}) \leq \frac{\delta_{n+1}}{K_\phi} + \frac{\left(\frac{\ell u}{\zeta}\right)^2}{1-\frac{\ell u}{\zeta}}. \tag{5.5}$$

Adding inequalities from equation (5.4) and equation (5.5) we get

$$-E_{n+1} - \frac{nu^2}{1-u} - \frac{\left(\frac{\ell u}{\zeta}\right)^2}{1-\frac{\ell u}{\zeta}} \leq \log(\prod_{i=1}^{n}(1+\delta_i)(1+\frac{\delta_{n+1}}{K_\phi})) \leq E_{n+1} + \frac{nu^2}{1-u} + \frac{\left(\frac{\ell u}{\zeta}\right)^2}{1-\frac{\ell u}{\zeta}}.$$

Then using the bound we computed for E_{n+1} from Lemma 4 we can weaken the inequality to obtain

$$-\lambda\sqrt{n+\frac{\ell^2}{\zeta^2}}u - \frac{nu^2}{1-u} - \frac{\left(\frac{\ell u}{\zeta}\right)^2}{1-\frac{\ell u}{\zeta}} \leq \log(\prod_{i=1}^{n}(1+\delta_i)(1+\frac{\delta_{n+1}}{K_\phi})) \leq \lambda\sqrt{n+\frac{\ell^2}{\zeta^2}}u + \frac{nu^2}{1-u} + \frac{\left(\frac{\ell u}{\zeta}\right)^2}{1-\frac{\ell u}{\zeta}}$$

which holds with probability at least $1-2\exp\left(-\lambda^2/2\right)$. By exponentiating this inequality we finally obtain

$$\prod_{i=1}^{n}(1+\delta_i)(1+\frac{\delta_{n+1}}{K_\phi}) = 1+\theta_n,\ |\theta_n| \leq \widetilde{\gamma}_n^{\text{act}}(\lambda).$$

□

By applying Lemma 5 to the expression in equation (4.2), we finally obtain our last result, a fully probabilistic error bound for a general artificial neural network.

Theorem 4 (Probabilistic error bound for artificial neural networks) *Consider a general neural network composed of p layers whose output can be expressed as follows*

$$y = \phi_p(A_p\phi_{p-1}(A_{p-1}\dots A_2\phi_1(A_1x)\dots)),$$

with $A_i \in \mathbb{R}^{n_i\times n_{i-1}}$, *for* $i = 1,\dots,p$ *and* $x \in \mathbb{R}^{n_0}$. *If the computation of* y *in floating-point arithmetic generates rounding errors that satisfy Model 4, then the computed result* $\widehat{y}$ *satisfies*

$$\widehat{y} = \phi_p((A_p+\Delta A_p)\phi_{p-1}((A_{p-1}+\Delta A_{p-1})\dots(A_2+\Delta A_2)\phi_1((A_1+\Delta A_1)x)\dots))$$

with

$$|\Delta A_i| \le \widetilde{\gamma}_{n_{i-1}+\ell^2_{\phi_i}/\kappa_{\phi_i}(A_i\widehat{y}_{i-1})^2}(\lambda)|A_i|,\ i = 1,\dots,p,$$

with probability at least

$$Q(\lambda, \sum_{i=1}^{p} n_i n_{i-1}).$$

5.3. *Summary and discussion*

TABLE 1 *Summary of the dominant term in each of the bounds.*

	Theorem 2	Theorem 3	Theorem 4
Assumptions	Models 1 and 3	Models 2 and 3	Model 4
Bound	$\left(n+\frac{\ell_\phi}{\kappa_\phi(Ax)}\right)u$	$\left(\lambda\sqrt{n}+\frac{\ell_\phi}{\kappa_\phi(Ax)}\right)u$	$\lambda\sqrt{n+\frac{\ell_\phi^2}{\kappa_\phi(Ax)^2}}u$

We provide in Table 1 a brief summary of the dominant term in the bounds that we obtained in Theorems 2, 3, and 4. The probabilistic approach improves the deterministic bounds by reducing the impact of n consecutive rounding errors from nu to $\sqrt{n}u$. This is the observed difference between the bounds of Theorem 2 and of Theorem 3. The latter uses a probabilistic model only for the matrix–vector product (Model 2), which is notably satisfied when stochastic rounding is used in the matrix–vector product [12].

The bound of Theorem 3 can be even further refined to that of Theorem 4 by also modelling the ℓ_ϕ rounding errors introduced by the activation function as random variables. Model 4 does not assume the individual ℓ_ϕ errors to be mean independent but does require them to be globally of mean zero and to be mean independent of the errors in the matrix–vector product. Whether these assumptions are realistic will depend on the specific implementation of the activation function. Therefore, in situations where Model 4 does hold, we can obtain the refined bound of Theorem 4, and otherwise we should revert to the mixed bound of Theorem 3, which only requires Model 2 and is thus guaranteed to hold with stochastic rounding.

Moreover, all our bounds also assume that the condition number of the activation is strictly positive at the computation point, since otherwise the term $1/\kappa_\phi(Ax)$ makes the bounds meaningless. This assumption should not be a problem, since if the conditioning is zero, then the function is constant, in which case there are no rounding errors. These bounds however show that the activation function's condition number is a quantity that can dictate whether the backward error is large or not. Indeed, the ℓ_{ϕ_i} rounding errors can be amplified or not by the condition number of ϕ_i.

We can interpret this in terms of adversarial attacks on the parameters. Since it has been shown, in [7], that there is a direct link between backward error and adversarial attacks on a neural network's parameters, we can expect these two quantities to behave similarly when the condition number varies. A small condition number is typically linked with more robust neural networks. This means that when the condition number increases we expect adversarial attacks with small norm to become easier to find, as demonstrated by Beerens and Higham [5]. The bounds in Theorem 4 show that for a fixed $\widehat{y}$ and a given layer, if the condition number of the activation function increases then the backward error will decrease and therefore the perturbations needed on the layer's parameters will have smaller norm. This is consistent with the fact that in this case, it will be easier to find adversarial attacks with smaller norm.

In terms of rounding error coming from the use of reduced arithmetic precision, our bounds suggest that one should use higher precision for the activation functions. Indeed, it seems that the error terms coming from the activation function can be arbitrarily large depending on the condition number. For example, given a layer with tanh activation, if the result of the matrix–vector falls within the threshold region of the hyperbolic tangent function, the condition number tends to zero, and using low precision on the activation function would result in quickly pushing all outputs to one or minus one. In this context, it would be appropriate to use higher precision to avoid this phenomenon. This finding seems to align with results of Hubara et al. [28] which show that we expect neural networks with low precision to behave better when adding more precision to its activation functions than to its parameters.

6. Numerical experiments

In this section, we will seek to validate the bounds obtained in sections 4 and 5 and summarized in Table 1. To do so, we will use the formulas obtained in section 3 to compute the backward error and the condition number for different neural networks. Once these quantities are obtained, we can compare the computed value of the backward error with its different theoretical bounds based on deterministic and/or probabilistic approaches. Additionally, we will compare the value of the forward error with its corresponding bound, which is obtained by multiplying the condition number by the bound on the backward error.

These experiments are carried out with Python 3.8. Computations are performed in single precision while "exact" quantities are computed in double precision. Experiments are conducted $N_{\text{test}} = 10$ times in order to compute the average, maximum, backward and forward errors to compare them with their associated deterministic and probabilistic bounds, using the condition number $\kappa_m(A,x)$. The probabilistic bounds are computed using $\lambda = 1$, as in the experiments of Higham and Mary [26]. For the tanh activation function, a value of $\ell = 2$ was found and confirmed experimentally in our framework. Note that $\kappa_\phi(Ax)$ is a vector since it is a componentwise condition number, we therefore choose to take its smallest component in order to get the worst-case componentwise bound. The matrix–vector product computations have been implemented in C using loops so that the code corresponds to the floating-point Model 1 and to our analysis. If the matrix–vector product is implemented differently, using blocking [8] for example, we typically expect our bounds to be more pessimistic.

We showed that in order to compute the backward error of deep neural networks, an optimization problem needs to be solved; this will be done using the CVXPY library [2, 16].

Three different bounds have been obtained for the backward error: a deterministic bound in Theorem 2, a mixed bound in Theorem 3, and a probabilistic bound in Theorem 4 as shown in Table 1. The aim of the initial experiments is to compare these bounds to better understand the relevance of probabilistic approaches.

6.1. *Backward and forward errors and their bounds on random neural networks*

We will first perform some experiments on untrained neural networks randomly initialized with different distributions, which allows us to easily vary some parameters, such as the size of the layers. We will compare two different random distributions. The first is a Gaussian $\mathcal{N}(0, \frac{1}{\sqrt{n}})$, where n is the number of neurons of a given layer. This choice comes from both the observation that trained layers' weight values typically converge to this type of distribution and the Xavier's initialization from Glorot and Bengio [19] which is the most widely used type of parameters' initialization before training the neural networks. This type of distribution considers the number of parameters of each layer to determine the scale of the random initialization. This allows the activation functions and gradients to work effectively during both the forward phase and the backpropagation used during training. The other distribution that we will test is a uniform distribution $\mathcal{U}(0, \frac{1}{\sqrt{n}})$. This distribution is not centred in zero and hence allows for observing different behaviours.

Figures 1 and 2 present the case of a one-layer neural network with tanh activation and varying size n. The figures show the evolution of the backward and forward errors and their corresponding theoretical bounds as a function of the number of neurons n. The network parameters and entries and randomly drawn from the $\mathcal{N}(0, \frac{1}{\sqrt{n}})$ distribution in Figure 1 and from the $\mathcal{U}(0, \frac{1}{\sqrt{n}})$ distribution in Figure 2.

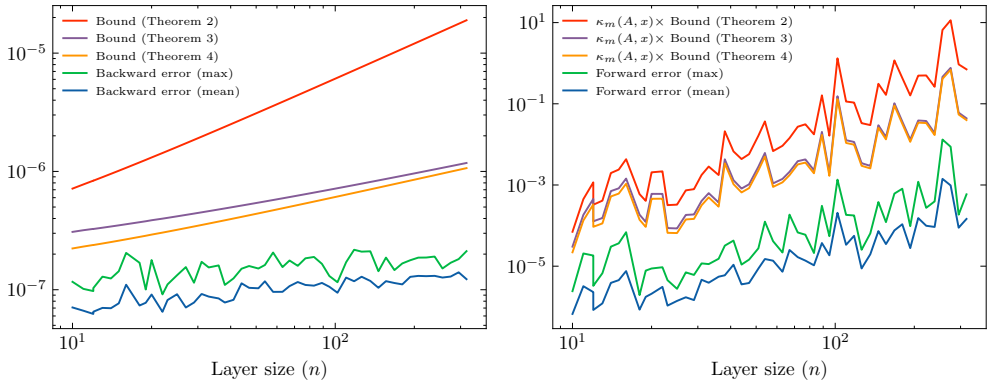

FIG. 1. Backward and forward errors and their bounds for a one-layer neural network of size n with random parameters and entries taken from the $\mathcal{N}(0, \frac{1}{\sqrt{n}})$ distribution.

Figures 1 and 2 clearly illustrate the differences between the various bounds obtained in this section. The gain between the deterministic approach and the two bounds using probabilistic assumptions is significant and is mostly due to the assumptions made about the matrix–vector product operations. The full probabilistic approach allows for having slightly sharper bounds, especially when n decreases since,

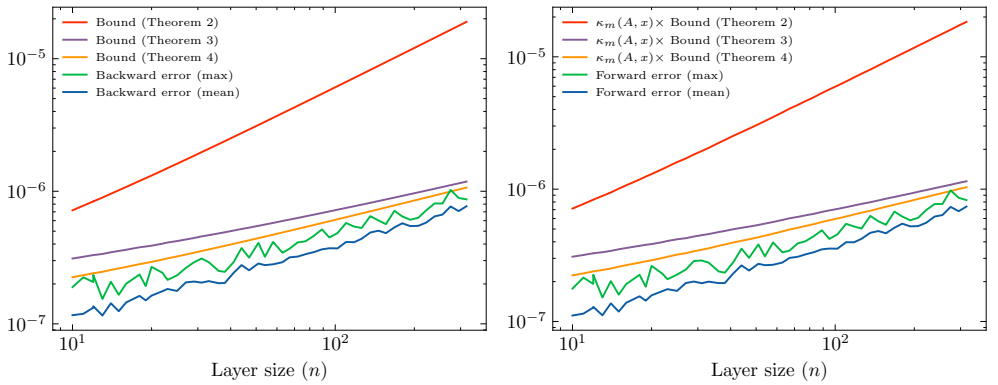

FIG. 2. Backward and forward errors and their bounds for a one-layer neural network of size n with random parameters and entries taken from the $\mathscr{U}(0, \frac{1}{\sqrt{n}})$ distribution.

in that case, errors introduced by the matrix–vector product get smaller while errors introduced by the activation function do not depend on n and therefore remain the same.

We observe that the probabilistic bound captures almost exactly the behaviour of the error for input data and parameters taken from a uniform positive distribution (in Figure 2) while being more pessimistic for the Gaussian distribution (in Figure 1). This can be explained by the analysis of Higham and Mary [27], who showed that when floating-point computations are performed on random data centred on zero, then the probabilistic backward error bound of order $\sqrt{n}u$ can be further refined to a constant of order u independent of n. In the absence of any assumptions regarding the distributions of the parameters and entries, the probabilistic bounds cannot be further enhanced.

The analysis of Higham and Mary [27] also explains the observed increase of the forward error when random values are taken from a mean zero distribution. In this case, the condition number grows at least proportionally to $\sqrt{n}$, but can be arbitrarily larger when the output of the matrix–vector product falls close to zero. The relative forward error may consequently be much larger in case of mean zero distributions.

Next, Figures 3 and 4 show the evolution of the errors and their bounds as the number of layers increases in a fully connected neural network with tanh activation function.

In Figure 3, which considers a normal distribution of data, we observe a slight decrease in the backward error while the condition number and forward error increase as the number of layers grows. The diminishing backward error is not captured by the bounds, which might come from the fact that we consider the maximum bound over all the layers, which may decrease with more layers if the perturbations are more equitably spread across layers.

In Figure 4, which considers a uniform positive distribution of data, after a few layers the accumulation of positive values will result in an output of the tanh activation function close to one. In this case, the condition number decreases towards zero, which leads to a smaller forward error and a correspondingly larger backward error. Indeed, when the input values are large enough, it requires higher perturbations to get an output smaller than one. We chose a $\mathscr{U}(0, \frac{1}{n^\alpha})$ distribution, with $\alpha = 0.6$, to better highlight the effect of a decrease in condition number. For smaller values of α this decrease would be faster and lead to a steeper increase in backward error and since the model utilized to derive

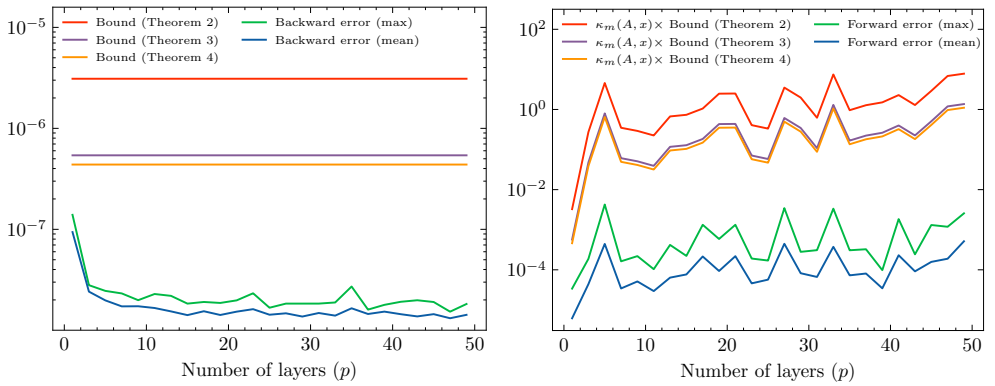

FIG. 3. Backward and forward errors and their bounds for neural networks of increasing number of layers, each layer is of size 50 with tanh as activation function, with random parameters and entries taken from the $\mathcal{N}(0, \frac{1}{\sqrt{n}})$ distribution.

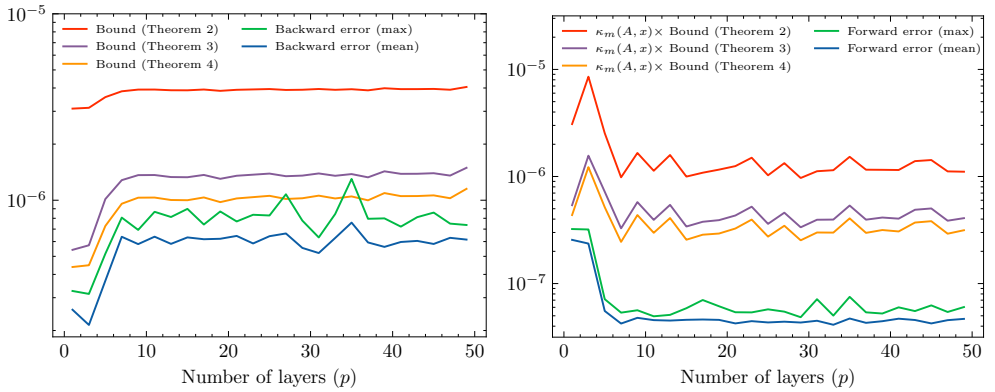

FIG. 4. Backward and forward errors and their bounds for neural networks of increasing number of layers, each layer is of size 50 with tanh as activation function, with random parameters and entries taken from the $\mathscr{U}(0, \frac{1}{n^{\alpha}})$ distribution.

bounds on the backward error depends on the condition number not approaching zero, the bounds' accuracy should diminish.

6.2. *Backward and forward errors and their bounds on trained neural networks*

To better assess the robustness of the obtained bounds, we will apply our results to trained networks. Our experiments consist in training a network and, for each training step, evaluating the backward error, forward error, conditioning, and associated bounds on $N_{\text{test}} = 10$ images from the testing dataset. Errors are evaluated on the output vector before the classification decision, which usually consists in associating the input to the class k when the output's k-th component contains the maximum of its components.

The trained neural networks architectures are provided in Table 2. They are initialized with Xavier's initialization and trained on FashionMNIST [48], a dataset which allows for a more challenging classification task than MNIST while maintaining the same input sizes. Networks are trained using

a cross entropy loss and Adam optimizer with a default learning rate of 10^{-3} and batch size 128. Both networks attain approximately 90% accuracy on the testing dataset.

TABLE 2 *Neural networks architectures details.*

Fully connected model		Convolutional model	
Layer	Shape	Layer	Shape
linear1-tanh	784×500	conv1-ReLU	$1 \times 6 \times 25$
linear2-tanh	500×500	max-pooling	2×2 (stride 2)
linear3-tanh	500×500	conv2-ReLU	$6 \times 12 \times 25$
linear4-ReLU	500×10	max-pooling	2×2 (stride 2)
		linear3-tanh	192×500
		linear4-tanh	500×120
		linear5-tanh	120×60
		linear6-ReLU	60×10

Figure 5 presents the backward and forward errors and their respective bounds for the fully connected network given on the left side of Table 2.

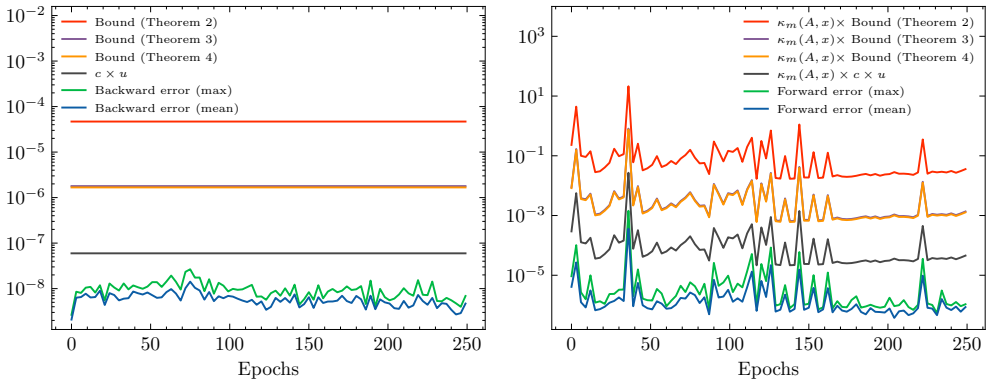

FIG. 5. Backward and forward errors and their bounds during the training of a small connected network on FashionMNIST.

Bounds on the backward error provide information that is in good agreement with the errors' behaviour. However, they seem to be rather pessimistic. Since these neural networks are trained using Glorot and Bengio [19] initialization, and that their parameters typically converge to a zero-mean Gaussian distribution, we are in the case where the sharper bounds of Higham and Mary [27] are applicable. In Figure 5 we assess whether these sharper bounds are indeed valid by integrating a bound that is proportional to cu, where c is independent of n. In order to estimate the constant c, we use Higham and Mary [27, Theorem 3.3]. Since in our case parameters typically follow a normal distribution, let say $\mathcal{N}(0,\sigma)$, then with very high probability we know that these parameters are bounded by 2σ. Moreover, the absolute value of these parameters then follow the folded normal distribution of mean $\sqrt{2/\pi}\sigma$ [33]. This implies that c will typically be of the same order as $2\sigma/\sqrt{2/\pi}\sigma = \sqrt{2\pi}$. The resulting bounds are much sharper with respect to both the computed forward and backward error.

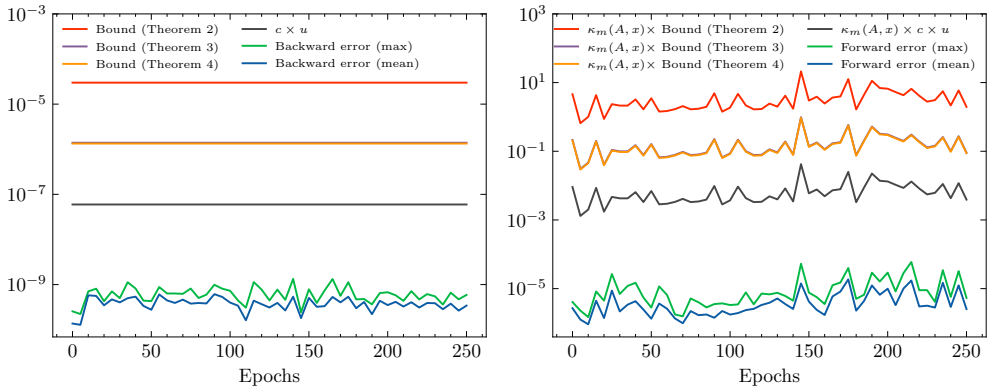

FIG. 6. Backward and forward errors and their bounds during the training of a convolutional network on FashionMNIST.

Figure 6 provides similar experiments for the convolutional neural network given on the right side of Table 2. These results show that the convolutional layer has an approximately ten times lower backward error, compared to the fully connected network of Figure 5, while maintaining the same order of magnitude for the forward error. This means that the condition number of this neural network is approximately ten times larger than for the fully connected one, suggesting that convolutional layers typically lead to networks that are more sensitive to perturbations on their input and/or parameters.

Note that the backward error bound involves the number n of columns of a given layer's weight matrix in the case of a fully connected network. In case of a convolutional layer the weight matrix is sparse and at most k coefficients are non-zero in each row, k being the kernel size. Therefore, for convolutional layers, the number n is replaced by $k = 5$ in our case. This means that we expect convolutional layers to have much smaller backward errors than fully connected ones. However, since the neural network used in Figure 6 also comprises fully connected layers, the bounds do not benefit from these observations.

7. Conclusion

The goal of this work was to provide formal tools to better understand, explain, and predict the accuracy and stability of neural networks when using floating-point arithmetic. We have achieved this goal in three steps. First, we have identified key quantities, the backward error and condition number of a neural network, and established formulas to compute these quantities in section 3. Then, in sections 4 and 5, we carried out rounding error analyses to derive several theoretical bounds on the backward error, based on different deterministic or probabilistic models for rounding errors. Finally, in section 6, we confirmed experimentally that these bounds are both valid and sharp, thanks to the probabilistic approach.

The derived bounds on the backward error provide insight on the identification of layers or building blocks that have the most impact on a neural network stability and sensitiveness to small perturbations such as rounding errors. This allows for a better understanding and control on the design or choice of neural network architectures and training setup. Indeed, our results suggest that the use of zero-mean and rightly scaled initialization, such as proposed by Glorot and Bengio [19], in addition to maintaining the magnitudes of the activation functions from one layer to another during the propagation phase, can

also lead to significant rounding errors reduction. Moreover, our results also suggest that while fully connected layers are fairly resilient to changes in precision, activation functions should be given more priority in terms of precision, particularly when applied to values that induce a condition number close to zero.

Concerning the computation of backward errors for neural networks, this work started from existing approaches in numerical linear algebra and thus employed a formulation of layers as matrix–vector products. This establishes a foundation upon which to work, both from the perspective of the computation of the error itself and from the rounding error analysis. Indeed, most existing layers base their computations upon such formulations; in the other cases, the formulas and bounds obtained will need to be adapted, building upon existing work.

The cost of using verification tools such as CADNA or FLUCTUAT, to ensure correct behaviour of algorithms when using finite precision, can be prohibitively large. Recent research of Graillat et al. [22], presenting the PROMISE algorithm, aims at using CADNA to produce mixed-precision quantization of neural networks that ensure a given accuracy. By providing tools that can identify the most sensitive blocks of a given network, the rounding error analysis performed in this work could help in guiding such software so that they only target robust layers and therefore gain significant computation time.

Acknowledgments

This work was supported by the NumPEx Exa-MA (ANR-22-EXNU-0002), MixHPC (ANR-23-CE46-0005-01), and InterFLOP (ANR-20-CE46-0009) projects of the French National Research Agency (ANR), the Artificial and Natural Intelligence Toulouse Institute (ANITI) of Université de Toulouse (France) and the Eviden company.

References

1. Nabih N. Abdelmalek. Minimum l_∞ solution of underdetermined systems of linear equations. *Journal of Approximation Theory*, 20(1):57–69, 1977.
2. Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen P. Boyd. A rewriting system for convex optimization problems. *CoRR*, abs/1709.04494, 2017. URL http://arxiv.org/abs/1709.04494.
3. Mario Arioli, James Weldon Demmel, and Iain S. Duff. Solving sparse linear systems with sparse backward error. *SIAM Journal on Matrix Analysis and Applications*, 10(2):165–190, 1989.
4. Ron Banner, Itay Hubara, Elad Hoffer, and Daniel Soudry. Scalable methods for 8-bit training of neural networks. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 5151–5159, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/e82c4b19b8151ddc25d4d93baf7b908f-Abstract.html.
5. Lucas Beerens and Desmond J. Higham. Adversarial ink: Componentwise backward error attacks on deep learning. *CoRR*, abs/2306.02918, 2023. doi: 10.48550/ARXIV.2306.02918. URL https://doi.org/10.48550/arXiv.2306.02918.
6. David Bertoin, Jérôme Bolte, Sébastien Gerchinovitz, and Edouard Pauwels. Numerical influence of relu'(0) on backpropagation. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 468–479, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/043ab21fc5a1607b381ac3896176dac6-Abstract.html.
7. Théo Beuzeville, Pierre Boudier, Alfredo Buttari, Serge Gratton, Théo Mary, and Stéphane Pralet. Adversarial attacks via backward error analysis. *hal-03296180*, December 2021. URL https://ut3-toulouseinp.hal.science/hal-03296180.

8. Pierre Blanchard, Nicholas J. Higham, and Théo Mary. A class of fast and accurate summation algorithms. *SIAM J. Sci. Comput.*, 42(3):A1541–A1557, 2020. doi: 10.1137/19M1257780. URL https://doi.org/10.1137/19M1257780.
9. James A. Cadzow. An efficient algorithmic procedure for obtaining a minimum l_∞-norm solution to a system of consistent linear equations. *SIAM Journal on Numerical Analysis*, 11(6):1151–1165, 1974.
10. Françoise Chaitin-Chatelin and Valérie Frayssé. *Lectures on finite precision computations*. Software, environments, tools. SIAM, 1996. ISBN 978-0-89871-358-9.
11. Dan C. Ciresan, Alessandro Giusti, Luca Maria Gambardella, and Jürgen Schmidhuber. Mitosis detection in breast cancer histology images with deep neural networks. In *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2013 - 16th International Conference, Nagoya, Japan, September 22-26, 2013, Proceedings, Part II*, volume 8150 of *Lecture Notes in Computer Science*, pages 411–418. Springer, 2013. doi: 10.1007/978-3-642-40763-5_51. URL https://doi.org/10.1007/978-3-642-40763-5_51.
12. Michael P. Connolly, Nicholas J. Higham, and Théo Mary. Stochastic rounding and its probabilistic backward error analysis. *SIAM J. Sci. Comput.*, 43(1):A566–A585, 2021. doi: 10.1137/20M1334796. URL https://doi.org/10.1137/20M1334796.
13. Matteo Croci, Massimiliano Fasi, Nicholas J. Higham, Theo Mary, and Mantas Mikaitis. Stochastic rounding: Implementation, error analysis and applications. *Royal Society Open Science*, 9(3):1–25, 2022. doi: 10.1098/rsos.211631.
14. James Weldon Demmel. On condition numbers and the distance to the nearest ill-posed problem. *Numerische Mathematik*, 51:251–289, 1987.
15. James Weldon Demmel and William Kahan. Accurate singular values of bidiagonal matrices. *SIAM J. Sci. Comput.*, 11(5):873–912, 1990. doi: 10.1137/0911052. URL https://doi.org/10.1137/0911052.
16. Steven Diamond and Stephen P. Boyd. CVXPY: A python-embedded modeling language for convex optimization. *J. Mach. Learn. Res.*, 17:83:1–83:5, 2016. URL http://jmlr.org/papers/v17/15-408.html.
17. Adam Christopher Earle. *Minimum l_∞ norm solutions to finite dimensional algebraic underdetermined linear systems*. PhD thesis, University of the Witwatersrand, 2015.
18. Pacôme Eberhart, Julien Brajard, Pierre Fortin, and Fabienne Jézéquel. High performance numerical validation using stochastic arithmetic. *Reliable Computing*, 21:35–52, 2015.
19. Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, volume 9 of *JMLR Proceedings*, pages 249–256. JMLR.org, 2010. URL http://proceedings.mlr.press/v9/glorot10a.html.
20. Israel Gohberg and Israel Koltracht. Mixed, componentwise, and structured condition numbers. *SIAM J. Matrix Anal. Appl.*, 14(3):688–704, 1993. doi: 10.1137/0614049. URL https://doi.org/10.1137/0614049.
21. Eric Goubault. Static analysis by abstract interpretation of numerical programs and systems, and FLUCTUAT. In *Static Analysis - 20th International Symposium, SAS 2013, Seattle, WA, USA, June 20-22, 2013. Proceedings*, volume 7935 of *Lecture Notes in Computer Science*, pages 1–3. Springer, 2013. doi: 10.1007/978-3-642-38856-9_1. URL https://doi.org/10.1007/978-3-642-38856-9_1.
22. Stef Graillat, Fabienne Jézéquel, Romain Picot, François Févotte, and Bruno Lathuilière. Auto-tuning for floating-point precision with discrete stochastic arithmetic. *J. Comput. Sci.*, 36, 2019. doi: 10.1016/J.JOCS.2019.07.004. URL https://doi.org/10.1016/j.jocs.2019.07.004.
23. Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1737–1746. JMLR.org, 2015. URL http://proceedings.mlr.press/v37/gupta15.html.
24. Nicholas J. Higham. A survey of condition number estimation for triangular matrices. *Siam Review*, 29(4):575–596, 1987.
25. Nicholas J. Higham. *Accuracy and stability of numerical algorithms*. SIAM, 1996. ISBN 978-0-89871-355-8.
26. Nicholas J. Higham and Théo Mary. A new approach to probabilistic rounding error analysis. *SIAM J. Sci. Comput.*, 41(5):A2815–A2835, 2019. doi: 10.1137/18M1226312. URL https://doi.org/10.1137/18M1226312.

27. Nicholas J. Higham and Théo Mary. Sharper probabilistic backward error analysis for basic linear algebra kernels with random data. *SIAM J. Sci. Comput.*, 42(5):A3427–A3446, 2020. doi: 10.1137/20M1314355. URL https://doi.org/10.1137/20M1314355.
28. Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *J. Mach. Learn. Res.*, 18: 187:1–187:30, 2017. URL http://jmlr.org/papers/v18/16-456.html.
29. Arnault Ioualalen and Matthieu Martel. Neural network precision tuning. In *Quantitative Evaluation of Systems, 16th International Conference, QEST 2019, Glasgow, UK, September 10-12, 2019, Proceedings*, volume 11785 of *Lecture Notes in Computer Science*, pages 129–143. Springer, 2019. doi: 10.1007/978-3-030-30281-8_8. URL https://doi.org/10.1007/978-3-030-30281-8_8.
30. Fabienne Jézéquel and Jean-Marie Chesneaux. Cadna: a library for estimating round-off error propagation. *Computer Physics Communications*, 178(12):933–955, 2008.
31. Kai Jia and Martin C. Rinard. Exploiting verified neural networks via floating point numerical error. In *Static Analysis - 28th International Symposium, SAS 2021, Chicago, IL, USA, October 17-19, 2021, Proceedings*, volume 12913 of *Lecture Notes in Computer Science*, pages 191–205. Springer, 2021. doi: 10.1007/978-3-030-88806-0_9. URL https://doi.org/10.1007/978-3-030-88806-0_9.
32. 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 *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. doi: 10.1007/978-3-319-63387-9_5. URL https://doi.org/10.1007/978-3-319-63387-9_5.
33. Fred C Leone, Lloyd S Nelson, and RB Nottingham. The folded normal distribution. *Technometrics*, 3(4): 543–550, 1961.
34. Yurii Il'ich Lyubich. Conditionality in general computational problems. In *Doklady Akademii Nauk*, volume 171:4, pages 791–793. Russian Academy of Sciences, 1966.
35. Ravi Mangal, Aditya V. Nori, and Alessandro Orso. Robustness of neural networks: a probabilistic and practical approach. In *Proceedings of the 41st International Conference on Software Engineering: New Ideas and Emerging Results, ICSE (NIER) 2019, Montreal, QC, Canada, May 29-31, 2019*, pages 93–96. IEEE / ACM, 2019. doi: 10.1109/ICSE-NIER.2019.00032. URL https://doi.org/10.1109/ICSE-NIER.2019.00032.
36. Matthew Mirman, Maximilian Baader, and Martin T. Vechev. The fundamental limits of interval arithmetic for neural networks. *CoRR*, abs/2112.05235, 2021. URL https://arxiv.org/abs/2112.05235.
37. NVIDIA. *CUDA C Programming Guide*, 2019. URL https://docs.nvidia.com/cuda/archive/10.1/pdf/CUDA_C_Programming_Guide.pdf.
38. Giacomo De Palma, Bobak Toussi Kiani, and Seth Lloyd. Adversarial robustness guarantees for random deep neural networks. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 2522–2534. PMLR, 2021. URL http://proceedings.mlr.press/v139/de-palma21a.html.
39. John R. Rice. A theory of condition. *SIAM Journal on Numerical Analysis*, 3(2):287–310, 1966.
40. Dayana Savostianova, Emanuele Zangrando, Gianluca Ceruti, and Francesco Tudisco. Robust low-rank training via approximate orthonormal constraints. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/d073692637b4fb8c4eb4b81f0fa2df7b-Abstract-Conference.html.
41. Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin T. Vechev. Fast and effective robustness certification. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 10825–10836, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/f2f446980d8e971ef3da97af089481c3-Abstract.html.

42. Robert D. Skeel. Scaling for numerical stability in gaussian elimination. *J. ACM*, 26(3):494–526, 1979. doi: 10.1145/322139.322148. URL https://doi.org/10.1145/322139.322148.
43. Wai Sun Tang, Jun Wang, and Yangsheng Xu. Infinity-norm torque minimization for redundant manipulators using a recurrent neural network. In *Proceedings of the 38th IEEE Conference on Decision and Control (Cat. No. 99CH36304)*, volume 3, pages 2168–2173. IEEE, 1999.
44. 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. URL https://openreview.net/forum?id=HyGIdiRqtm.
45. Lloyd N. Trefethen and David Bau. *Numerical linear algebra*. SIAM, 1997. ISBN 978-0-89871-361-9. doi: 10.1137/1.9780898719574. URL https://doi.org/10.1137/1.9780898719574.
46. Bichen Wu, Forrest N. Iandola, Peter H. Jin, and Kurt Keutzer. Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 446–454. IEEE Computer Society, 2017. doi: 10.1109/CVPRW.2017.60. URL https://doi.org/10.1109/CVPRW.2017.60.
47. Shuang Wu, Guoqi Li, Feng Chen, and Luping Shi. Training and inference with integers in deep neural networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=HJGXzmspb.
48. Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017. URL http://arxiv.org/abs/1708.07747.