# What happens if neurons crash?

El Mahdi El Mhamdi, Rachid Guerraoui, Sergei Volodin

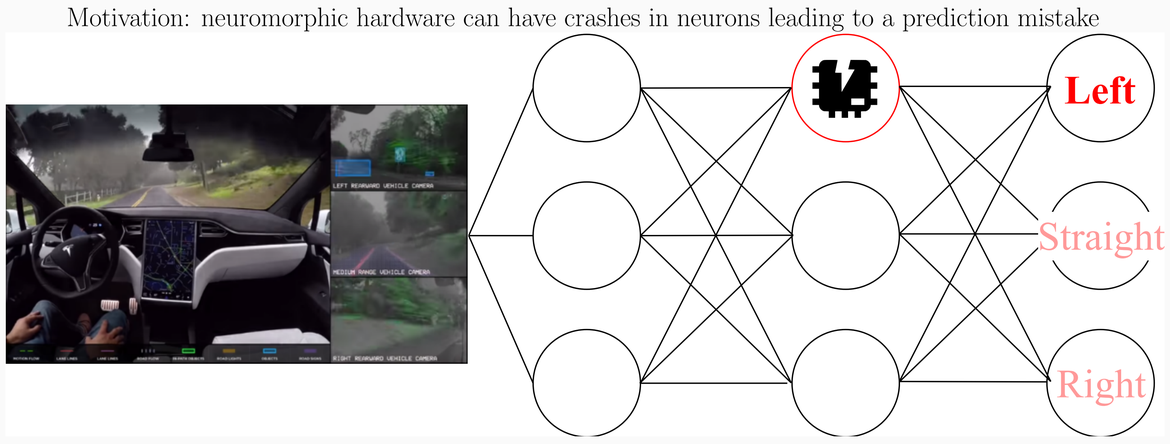
Swiss Federal Institute of Technology in Lausanne (EPFL)

## Abstract

Neuromorphic hardware is gaining momentum. In order to make neuromorphic hardware safe, it is necessary to understand fault tolerance of neural networks. In this project we first show that this problem is hard, and then introduce the necessary assumptions to calculate this quantity. We test the bounds in proof-of-concept experiments. Our contribution is an algorithm for certifying achieving fault tolerance. Compared to other approaches, we have formal guarantees of fault tolerance.

## Why fault tolerance?

Motivation: neuromorphic hardware can have crashes in neurons leading to a prediction mistake




## Introduction

Fault tolerance was a popular line of research before the last AI winter (1990s) because neural networks were expected to be implemented in neuromorphic hardware. However, researchers then were limited to shallow architectures and their results are not applicable to modern networks. Right now neuromorphic hardware is becoming popular again, which means that fault tolerance is an essential desirable property of neural networks. In this work we study error propagation in neural networks and connect fault tolerance to other properties of a neural network such as robustness to adversarial examples and generalization properties.

## Our model

**Notations.** $(x, y) = \sum_{i=1}^{n} x_i y_i$ is the scalar product. Matrix $p$-norm is defined as an operator norm for $p = (0, +\infty]$: $\|A\|_p = \sup_{x \neq 0} \|Ax\|_p / \|x\|_p$. We call a vector $0 \neq x \in \mathbb{R}^n$ $q$-balanced if $\min |x_i| \geq q \max |x_i|$. We define KL-divergence between numbers $a, b \in (0, 1)$ as $d_{KL}(a, b) = a \log \frac{a}{b} + (1 - a) \log \frac{1-a}{1-b}$. By $\Theta_{\pm}(1)$ we denote any function taking values in $[-1, 1]$.

**Definition 1.** *(Neural network) A neural network with $L$ layers is a function $y_L \colon \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ defined by a tuple $(L, W, B, \varphi)$ with weights matrices $W = (W_1, ..., W_L)$ of size $W_l \colon n_l \times n_{l-1}$ (or their distributions), biases $B = (b_1, ..., b_L)$ of size $b_l \in \mathbb{R}^{n_l}$ (or their distributions) by the expression $y_l = \varphi(z_l)$, $z_l = W_l y_{l-1} + b_l$, $l \in \overline{1, L-1}$, $z_0 = y_0 = x$ and $z_L = y_L = W_L y_{L-1} + b_L$. We additionally require $\varphi$ to be 1-Lipschitz $|\varphi(x) - \varphi(y)| \leqslant |x - y|$[1]. Note that the last layer is linear. We assume that the network was trained using data $x, y \sim X \times Y$ using Empirical Risk Minimization for some loss function $l(\cdot, \cdot)$: $\mathcal{L}(W, B) = \frac{1}{K} \sum_{k=1}^{K} l(y_L(x_k), y_k) \to \min_{W,B}$. Loss layer for input $x$ and label $y^*$ is defined as $y_{L+1}(x) = l(y_L(x), y^*)$*

**Definition 2.** *(Weight failure) Network $(L, W, B, \varphi)$ with weight failures with distribution $D|x, W$ is the network $(L, W + U, B, \varphi)$ for $U \sim D|x, W$. We denote a (random) output of this network as $y^{W+U}(x) = \hat{y}_L(x)$ with activations $\hat{y}_l$ and pre-activations $\hat{z}_l$.*

**Definition 3.** *(Bernoulli neuron failures) Bernoulli neuron crash distribution is the distribution with fully independent $\xi_i^l \sim \mathrm{Be}(p_l)$, $U_l^{ij} = -\xi_i^l \cdot W_l^{ij}$. For each possible crashing neuron $i$ at layer $l$ we define $U_l^i = \sum_j |U_l^{ij}|$ and $W_l^i = \sum_j |W_l^{ij}|$*

**Definition 4.** *(Output error for a weight distribution) The error in case of weight failure with distribution $D|x, W$ is $\Delta_l(x) = y_l^{W+U}(x) - y_l^W(x)$ for layers $l \in \overline{1, L+1}$*

**Definition 5.** *($(\varepsilon, \delta)$-fault tolerance) A network $(L, W, B, \varphi)$ is said to be $(\varepsilon, \delta)$-fault tolerant over an input distribution $(x, y^*) \sim X$ and a crash distribution $U \sim D$ if $\mathbb{P}_{(x,y^*) \sim X \times Y, U \sim D|x}\{\Delta_{L+1}(x) \geq \varepsilon\} \leq \delta$. In case if a network $(L, W, B, \varphi)$ is $(\varepsilon, \delta)$-fault tolerant, we write $(W, B) \in \mathrm{FT}(L, \varphi, p, \varepsilon, \delta)$.*

## Hardness of the fault tolerance in the general case

We show first that the task of computing the fault tolerance is NP-hard in general. In addition, there are "pathological" cases for which even a small weight perturbation could lead to a large change in the output. We overcome these issues by introducing additional assumptions.

**Proposition 1** (NP-hardness)**.** *The task of evaluating $\mathbb{E}\Delta^k$ for any $k = 1, 2, ...$ with constant additive or multiplicative error for a neural network with $\varphi \in C^\infty$, Bernoulli neuron crashes and a constant number of layers is NP-hard.*

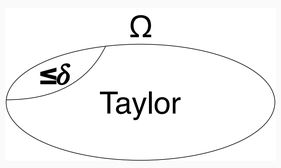
The reason our next considerations work is because we do not provide a constant-factor approximation, but one dependent on the network. In other words, for weights $W$ we give two numbers $\overline{\Delta}$ and $\underline{\Delta}$ dependent on $W$ such that $\underline{\Delta}(W) \leq \mathbb{E}\Delta \leq \overline{\Delta}(W)$.

**Proposition 2** (Pessimistic spectral bound (4))**.** $\|y_L(x_2) - y_L(x_1)\|_2 \leqslant \|x_2 - x_1\| \cdot \prod_{l=1}^{L} \|W_l\|_2$

This proposition shows the worst case which is sometimes achieved (4).

## Sufficient conditions for fault tolerance

We overcome issues stated in the previous section by using additional assumptions. We will first bound the probability $\delta$ that too many neurons crash. Next, for the probable case, we will use a Taylor expansion.




### Perturbation is rarely big

**Assumption 1.** *The probability of failure $p = \max\{p_l | l \in [L]\}$ is small, $p \sim 10^{-4} - 10^{-3}$*

This assumption is based on the properties of neuromorphic hardware (5).

**Assumption 2.** *The number of neurons at each layer $n_l$ is sufficiently big, $n \sim 10^2$*

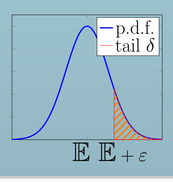
This is true for modern networks.

**Assumption 3.** *Vectors $W_l^i = \sum_j |W_l^{ij}|$ are $q$-balanced for $q > 0$ for each layer $l$.*

This is enforced by using a regularizer.

**Proposition 3.** *Under assumptions 1, 2 and 3, for $\alpha > p$, the norm of the weight perturbation $U_l^i$ at layer $l$ is probabilistically bounded as: $\delta_0 = \mathbb{P}\{\|U_l^i\|_1 \geq \alpha \|W_l^i\|\} \leq \exp\left(-n_l q d_{KL}(\alpha || p_l)\right)$*

This is a tail bound on the weight perturbation.




### Small perturbation allows for a Taylor expansion

**Assumption 4.** *We assume that the largest eigenvalue of the Hessian of $y_{L+1}$ is globally bounded as an absolute constant: $\|H\|_2 \leq H_{\max}$*

Assumption 4 is consistent with experimental studies (2) and can be connected to generalization properties via minima sharpness (3). A possible explanation would be that since NN fits some ground truth function $\overline{y}$, its derivatives could also fit that function's derivatives which are bounded.

**Assumption 5.** *We assume that with increasing $n_l$, sums of the input weights for a neuron $W_i^l$ stay the same, and derivatives $\frac{\partial y}{\partial \xi_i^l}$ decay as $1/n_l$*

For example, consider $y(x) = \frac{1}{n} \sum_{i=1}^{n} w_i^T x$, a network with weights decaying as $1/n$, for which this trivially holds when the size of $x$ or $n$ increases. Weights must decay if we assume that the network approximates the same function and the magnitude of values at each layers stays the same. This assumption is consistent with our experiments.

**Proposition 4.** *For crashes at layer $l$ and output of layer $L$ under assumptions 4 and 5 the mean and variance of the error can be approximated as*

$$\mathbb{E}\Delta_L = p_l \sum_{i=1}^{n_l} \frac{\partial y_L}{\partial \xi_i} + \Theta_{\pm}(1) H_{\max} C_1 \frac{p_l}{n_l}, \ \mathrm{Var}\Delta_L = p_l \sum_{i=1}^{n_l} \left(\frac{\partial y_L}{\partial \xi_i}\right)^2 + \Theta_{\pm}(1) H_{\max} C_2 \frac{p_l}{n_l}$$

**Proposition 5.** *A neural network under assumptions 1-5 is $(\varepsilon, \delta)$-fault tolerant for $t = \varepsilon - \mathbb{E}\Delta_L > 0$ with $\delta = \delta_0 + t^{-2}\mathrm{Var}\Delta_L$ for $\mathbb{E}\Delta$ and $\mathrm{Var}\Delta$ calculated by Proposition 4 and $\delta_0$ from Proposition 3.*

**Fault tolerance and architecture.** Under the assumptions 1-5, the variance of the error decays as

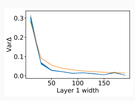
$$\mathrm{Var}\Delta \sim \sum \frac{C_l p_l}{n_l}$$

**Proposed regularizer** to satisfy Assumption 3 and increase Fault Tolerance by propositions 5, 4.

$$\hat{\mathcal{L}}(W) = \mathcal{L}(W) + \lambda \sum_{i=1}^{n} \left(\frac{\partial L}{\partial w_i}\right)^2 w_i^2 + \mu \left(\frac{W_{\max}^i}{W_{\min}^i}\right)^2 + \nu \|W\|_2^2 \tag{1}$$

## Experiments

First, we test Propositions 4 and 3 on small fully-connected networks $n \sim 50$ and $L \sim 5$ on the MNIST and Boston Housing datasets. We use larger networks (VGG-16, MobileNet) to test the Proposition 4 and the decay of the error experimentally as well. We confirm the predictions of our theory on the decay of the error. Then we test an algorithm based on Proposition 5 to guarantee fault tolerance.




Layer width and fault tolerance for Boston housing-trained NNs

## Conclusion

Crash fault tolerance is an overlooked concrete AI safety problem (1). We introduce a probabilistic framework to study fault tolerance. We first show that the problem is hard in general. Next we introduce the necessary assumptions to obtain results for a practical case of neuromorphic hardware. We design and test a novel algorithm for certifying fault tolerance. Compared to other approaches, our method can guarantee fault tolerance formally.

## Bibliography

[1] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[2] Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density. *arXiv preprint arXiv:1901.10159*, 2019.

[3] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[4] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. Exploring Generalization in Deep Learning. 2017.

[5] Catherine D Schuman, Thomas E Potok, Robert M Patton, J Douglas Birdwell, Mark E Dean, Garrett S Rose, and James S Plank. A survey of neuromorphic computing and neural networks in hardware. *arXiv preprint arXiv:1705.06963*, 2017.