Gradient-Informed Neural Network Statistical Robustness Estimation

Karim TIT
Thales Land and Air Systems, La Ruche
Rennes, France

Teddy Furon
Univ. Rennes, Inria, CNRS, IRISA
Rennes, France

Mathias Rousset
Univ. Rennes, Inria, CNRS, IRMAR
Rennes, France

Abstract

Deep neural networks are robust against random corruptions of the inputs to some extent. This global sense of safety is not sufficient in critical applications where probabilities of failure must be assessed with accuracy. Some previous works applied known statistical methods from the field of rare event analysis to classification. Yet, they use classifiers as black-box models without taking into account gradient information, readily available for deep learning models via auto-differentiation. We propose a new and highly efficient estimator of probabilities of failure dedicated to neural networks as it leverages the fast computation of gradients of the model through back-propagation.

1 INTRODUCTION

In many machine learning applications, test data are captured by a sensor, then quantized or compressed, and finally transmitted to the model. In each of these steps, uncertainties may corrupt the pieces of data. For instance, in autonomous driving, capturing the scene at night implies increasing the ISO parameter of the camera whence a photo-site noise increase (Foi et al., 2008). Statistical (a.k.a. Corruption) robustness is defined as the ability to perform correctly on test data corrupted by a given random noise distribution (Hendrycks and Dietterich, 2019).

Statistical robustness is a related but different concept from adversarial robustness, which is the ability of a model to perform correctly over maliciously perturbed data (Goodfellow et al., 2015). Gilmer et al. (2019) study in detail the relationship between these concepts. Moreover, Franceschi et al. (2018) provide theoretical and empirical evidence that, for input data of dimension d, there is a ratio of order $\frac{1}{\sqrt{d}}$ between the typical power of an adversarial signal and that of a random perturbation to trigger a misclassification in a neural network. Neural network classifiers are thus robust against random noise to a certain extent. Yet, safety requirements in critical applications, e.g. in autonomous driving (Koopman and Wagner, 2017), ask for very weak probabilities of failure under a typical noise power (Arief et al., 2021).

Webb et al. (2019) quantify the local statistical robustness of a model by its probability of failure. The major difficulty of this quantitative approach is the estimation of weak probabilities with accuracy and low complexity. The above-mentioned work resorts to a Sequential Monte Carlo technique (SMC, (Del Moral et al., 2006; Beskos et al., 2016)), dedicated to rare event analysis: the Adaptive Multi-Level Splitting (AMLS) procedure (Guyader et al., 2011). Querying the model is assumed to be time-consuming and the number of calls gauges the complexity of the estimator. Although their work proposes experiments on neural networks, it is not specific to deep learning as they use the model as a black box function. Our paper proposes an alternative gradient-informed SMC algorithm.

The proposed algorithm trades off the universality of the aforementioned work against a lower complexity, by leveraging a key feature of neural networks: the computation of the gradient of the model (here w.r.t. its input) is easily implemented thanks to auto-differentiation and cheaply run thanks to back-propagation.

2 RELATED WORKS

Statistical robustness can be considered either locally around an input point or globally for a given distribution on test data. Wang et al. (2021) measure statistical robustness globally via the probability that corrupted test data are misclassified with a crude Monte Carlo. This works because the global probability of failure is large as it is dominated by some non robust inputs. On the other hand, a local probability of failure can be extremely low and out of reach for a crude Monte Carlo simulation. This calls for specific rare event estimators (Webb et al., 2019).

Local robustness estimation is related to adversarial examples since we seek misclassified inputs locally around an

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206.

input z_0. Black-box attacks make no assumption about the model whereas white-box attacks use the gradient of the network function as we do. Attacks come without guarantee of finding an existing adversarial example. Moreover, when the attack finds an adversarial example, it is not clear whether this input is unique, isolated, or an element of a big adversarial set. The approach does not take into account the statistical distribution of the noise and thus can not compute the probability of failure (Baluta et al., 2021).

Formal verification methods answer to the lack of guarantee. They aim to prove or disprove the existence of an adversarial example in a given support set. Yet, achieving soundness (no false negative) and completeness (no false positive) is NP-hard w.r.t. the number of neurons Katz et al. (2017). Many works trade the completeness requirement off against speed and tractability, *e.g.* Weng et al. (2018). These methods do not consider any statistical model, which makes comparison difficult if not impossible.

Our work considers variants of SMC methods which have also been used in Bayesian inference. For instance, Ishikawa and Goda (2021) construct low-variance unbiased estimators of the model evidence thanks to the AMLS algorithm. However, the weights of the neural network are the random variables in this Bayesian inference work, whereas robustness estimation considers a statistical model of corruptions on the inputs.

The goal of statistical robustness evaluation is to detect whether the local probability of failure is higher than a given critical level $p_c < 1$. Webb et al. (2019) frame statistical robustness evaluation as an estimation problem, whereas Baluta et al. (2021) and Tit et al. (2021) frame it as a hypothesis test. In this paper we follow the estimation approach of Webb et al. (2019). This literature considers different tools. Baluta et al. (2021) use sequential testing with crude Monte Carlo, whereas Webb et al. (2019) and Tit et al. (2021) use variants of the AMLS algorithm. Both of the latter papers mention in their conclusions that the use of the gradient information is worth investigating as it could provide more accurate and faster robustness estimation.

To the best of our knowledge, this paper introduces the first gradient-informed neural network statistical robustness estimation algorithm. After introducing necessary background in section 3, we show how to include gradient information within a Sequential Monte Carlo in sections 4 and 5. Section 6 provides empirical evidence of a higher efficiency than the previous black box methods.

3 BACKGROUND

3.1 Local Robustness Assessment

Let function $f_\theta : \mathbb{R}^d \to \mathbb{R}^C$ be a classifier that maps elements in $\mathbb{R}^d$ into logits of C classes and z_0 be a clean input element. The label predicted by the model for this argument is denoted $\hat{y}(z_0) \triangleq \arg\max_{j\in 1:C} f_\theta(z_0)_j$. Local robustness gauges the correct classification of neighboring elements. Consider z an input in the neighbourhood of z_0 and the function:

$$r(z) \triangleq \max_{j\neq\hat{y}(z_0)} f_\theta(z)_j - f_\theta(z)_{\hat{y}(z_0)}. \tag{1}$$

If $r(z) \geq 0$, a misclassification occurs since $\hat{y}(z) \neq \hat{y}(z_0)$, otherwise $r(z)$ reveals to which extent the neighbouring element z is close to trigger a classification error.

Given an additional input distribution ρ_0, statistical robustness is then defined by:

$$\mathcal{R}[\theta, \rho_0] \triangleq \rho_0(r(Z) \geq 0) = \int_{\mathbb{R}^d} \mathbb{1}(r(z) \geq 0)\rho_0(dz) \tag{2}$$

The difficulty is then the estimation of this integral, in particular when the event $\{r(Z) \geq 0\}$ is rare under the distribution ρ_0. In the case of deep neural networks, f_θ is a complex non-convex function which makes the estimation problem harder.

For local statistical robustness, distribution ρ_0 is typically centered on the clean input z_0 s.t. $Z = z_0 + E$ where E models the random corruption errors. A common practice in Statistical Reliability Engineering resorts to a latent space where a random vector $X \sim \pi_0$ is first sampled and then mapped to $E = \zeta(X)$. This mapping is such that the push-forward distribution $\zeta\#\pi_0$ is identical to the distribution ρ_0 modelling the corruption errors. Examples of common transformations are given in (Tit et al., 2021) when π_0 is the multivariate normal law.

This paper follows the same path. We define the score function $s(X) \triangleq r(z_0 + \zeta(X))$ with $X \sim \pi_0$ and $\pi_0 = \mathcal{N}(\mathbf{0}, \mathbf{I})$. In the end, we express the probability of failure as

$$\mathcal{R}[\theta, \rho_0] = \mathbb{P}_{X\sim\pi_0}[s(X) \geq 0]. \tag{3}$$

3.2 Hamiltonian Monte Carlo

The Hamiltonian Monte Carlo (HMC) is a sampling method (Neal, 2011). Given a differentiable potential $U : \mathbb{R}^d \mapsto \mathbb{R}$, it generates samples asymptotically distributed according to $\pi_U \propto e^{-U}$. Define (up to a constant) the joint probability measure μ: $\mu(q, p) \propto e^{-H(q,p)}dqdp$, with $H(q, p) \triangleq U(q) + \|p\|^2/2$. Hamiltonian Monte Carlo method is based on the symplectic and time reversible discretization of the related Hamiltonian dynamics, also known as the Verlet scheme:

$$\begin{cases} P_{i+1/2} = P_i - \frac{1}{2}\nabla U(Q_i)\Delta t, \\ Q_{i+1} = Q_i + P_{i+1/2}\Delta t, \\ P_{i+1} = P_{i+1/2} - \frac{1}{2}\nabla U(Q_{i+1})\Delta t. \end{cases} \tag{4}$$

After L iterations of the Verlet scheme, the deterministic map $(Q_0, P_0) \mapsto (Q_L, -P_L)$ is reversible in time (an involution), and it conserves the flat phase-space measure $dqdp$. One can then apply the standard Metropolis accept-reject rule, which consists in accepting the proposal $(Q_L, -P_L)$ with probability:

$$\min(1, e^{-H((Q_L,-P_L))+H(Q_0,P_0)}). \tag{5}$$

The result is denoted $(\tilde{Q}_L; \tilde{P}_L) \in \{(Q_0, P_0), (Q_L, -P_L)\}$. The obtained Markovian kernel has stationary joint distribution μ, whose first marginal is the target π_U. When $L = 1$ the HMC kernel becomes a Metropolis-adjusted discretization of a diffusion with a drift $-\nabla U$, called the Metropolis Adjusted Langevin Algorithm (MALA, see Roberts and Tweedie (1996)).

4 SKETCH OF THE ALGORITHM

First, we define a one parameter family of smooth densities, which enables sampling with the gradient-informed HMC kernel defined above. Second, we recast the rare event problem as the normalization with respect to π_0 of the final density of this family. Finally, we construct our algorithm as a SMC approach (Beskos et al., 2016) integrating the gradient-informed HMC kernel.

4.1 A Family of Distributions

Let us consider the potential function:

$$x \in \mathbb{R}^d \mapsto V(x) \triangleq |-s(x)|_+ \geq 0, \tag{6}$$

where $|a|_+ = a$ if $a \geq 0$ and 0 otherwise, and function s defined in Sect. 3.1. Here the set $\{x : V(x) = 0\}$ is the set of latent vectors which give birth to violations of the local robustness property since the corresponding inputs are classified with a different label than the original z_0. Points with a low potential value are 'close' to being misclassified.

Given in addition a tempering parameter $\beta \geq 0$ (acting as an inverse temperature) and the reference measure π_0, we construct the following family of probabilities

$$\pi_\beta(dx) \triangleq \frac{e^{-\beta V(x)}}{Z_\beta}\pi_0(dx), \tag{7}$$

$$Z_\beta = \int e^{-\beta V(x)}\pi_0(dx) = \mathbb{E}_{\pi_0}\left[e^{-\beta V(X)}\right], \tag{8}$$

where $Z_\beta < +\infty$ is the normalizing constant assumed to be finite.

This family of distributions ranges from the reference measure π_0 for $\beta = 0$ to the same distribution but conditioned on the event of failure when $\beta \to +\infty$. In other words, $\pi_{+\infty}(dx) = \mathbb{1}(s(x) \geq 0)\pi_0(dx)/Z_{+\infty}$.

We denote the expectation of a test function t over the distribution π_β by $\mathbb{E}_{\pi_\beta}[t(X)]$. An interesting property used to construct our estimator in section 4.3 is that for any couple $(\alpha, \beta) \in \mathbb{R}^2_+$, the ratio Z_β/Z_α can be expressed as an expectation over the measure π_α:

$$Z_\beta = Z_\alpha \mathbb{E}_{\pi_\alpha}\left[e^{-(\beta-\alpha)V(X)}\right]. \tag{9}$$

Therefore, if one can sample from measure π_α, then one can estimate the ratio Z_β/Z_α by an empirical average.

4.2 Connection to the Rare Event

Normalizing constants are in general analytically intractable and hard to integrate numerically. In the family of distributions we consider (7), we simply have $Z_0 = 1$, while for $\beta = +\infty$ the normalizing constant is indeed the probability we want to estimate:

$$Z_{+\infty} = \mathbb{E}_{\pi_0}[\mathbb{1}(s(X) \geq 0)] = \mathcal{R}[\theta, \rho_0].$$

4.3 Key Ideas Supporting the Algorithms

Our algorithm is built upon the following key ideas.

The first step is to split the estimation problem into several easier intricate ones like in multilevel splitting algorithms (Kahn and Harris, 1951). This is driven by a series of increasing parameters $0 = \beta_0 \leq \beta_1 \leq \cdots \leq \beta_{K-1} \leq \beta_K = +\infty$. Indeed, using equation (9) recursively, we have

$$\begin{aligned} Z_{\beta_K} &= \mathbb{E}_{\pi_{\beta_{K-1}}}\left[e^{-(\beta_K-\beta_{K-1})V(X)}\right] \times Z_{\beta_{K-1}} \\ &= \prod_{k=0}^{K-1} \mathbb{E}_{\pi_{\beta_k}}\left[e^{-(\beta_{k+1}-\beta_k)V(X)}\right]. \end{aligned} \tag{10}$$

The main idea is then to estimate each expectation term of the product by an empirical average over a sample of N particles $\{X_k^{(n)}\}_{n=1}^N$ with empirical distribution $\pi^N_{\beta_k} \triangleq \frac{1}{N}\sum_{n=1}^N \delta_{X_k^{(n)}}$, and then to use a Sequential Monte Carlo (SMC) strategy (Del Moral et al., 2006) to sample $\pi^N_{\beta_{k+1}}$ from $\pi^N_{\beta_k}$. In the present context, we use the following simple iteration of selection and mutation steps:

Selection For each particle $X_k^{(n)}$ in the sample defining $\pi^N_{\beta_k}$, kill it with probability $e^{-(\beta_{k+1}-\beta_k)[V(X_k^{(n)})-\min_m V(X_k^{(m)})]}$. Let K_k be the number of killed particles. Draw K_k new particles uniformly among the $N - K_k$ survivors to keep the sample size equal to N.

Mutation Independently mutate particles using a Markov kernel $X_{k+1}^{(n)} = \texttt{Ker}(X_k^{(n)}, \beta_{k+1})$ which leaves invariant the distribution $\pi_{\beta_{k+1}}$ (see Sect. 5.1).

The following proposition establishes the soundness of this simulation method.

Proposition 1. *The following estimator of $\mathcal{R}[\theta, \rho_0] = Z_{+\infty}$ is unbiased and consistent:*

$$\widehat{Z}_{+\infty} = \prod_{k=0}^{K-1} \mathbb{E}_{\pi_{\beta_k}^N}\left[e^{-(\beta_{k+1}-\beta_k)V(X)}\right] = \prod_{k=0}^{K-1} \frac{\widehat{Z}_{k+1}}{\widehat{Z}_k}.$$

See App. B.1 for the proof.

5 PROPOSED ALGORITHM

This section details the algorithm whose pseudo-code is described in Alg. 1.

5.1 Sampling Kernel

Given a particle $X_k^{(n)}$ issued from the selection step, the mutation step in our main method uses the Hamiltonian Monte Carlo method of Sect. 3.2. We set $U = \beta_k V - \log \pi_0$ ($= -\log \pi_{\beta_k}$ up to an additive constant), $Q_0 = X_k^{(n)}$, and we sample P_0 independently according to a standard Gaussian law. We apply L steps of the Verlet scheme (4) and accept the outcome with probability given in (5). The result of this Metropolis step denoted $\tilde{Q}_L$ in Sect. 3.2 defines the sampling kernel:

$$\texttt{Ker}_{\Delta t, L}(X_k^{(n)}, \beta_k) \triangleq \tilde{Q}_L.$$

This kernel is invariant for the distribution π_{β_k} which is already approximated by the sample $(X_k^{(n)})_n$ after the selection step. We can adaptively tune the number of times the kernel is applied by applying the statistical stopping condition, based on auto-correlations, proposed by Buchholz et al. (2021). This corresponds to the `AutoCorrCond` function used at line 20 in the algorithm 1. In the experiments section 6 we chose T_{max} such that the condition is never reached, i.e. $T = T_{max}$. The auto-correlations condition is used to make sure that the proposed sample has low correlation to the initial value $X_k^{(n)}$. Similarly the time step Δt and the number L of Verlet iterations of the scheme (4) are tuned for each particle using the algorithm 5 of Buchholz et al. (2021), based on previous work from Fearnhead and Taylor (2010) in the context of Bayesian Analysis. This is a genetic algorithm adapting at each iteration of the kernel the value of Δt and L parameters using the ESJD (expectd square jump distance) performance metric (Buchholz et al., 2021). We also use in the experiments section the MALA (Metropolis-Adjusted Langevin Algorithm) method, in which case L is fixed to 1 and only Δt is adapted, using the same genetic algorithm. The `BatchKer` notation in line 18 outlines the batch processing of the particles through the Verlet scheme and the Metropolis steps.

5.2 Adaptive Tempering Parameters

The proposed algorithm also tunes the choice of the cooling schedule $(\beta_k)_k$. A too rapidly increasing schedule increases the variance of the terms in product (1). A too slow schedule increases the number of steps to reach the rare event, whence a bigger complexity. This trade-off has been studied in Beskos et al. (2016) for instance, where it is recommended to maintain the Efficient Sample Size constant across the steps. It amounts to set the next tempering parameter such that

$$\texttt{ESS}(\beta_{k+1}) = \alpha N \tag{11}$$

for a user-chosen $\alpha \in (1/N, 1)$ where

$$\texttt{ESS}(\beta) \triangleq \frac{\left(\sum_{n=1}^N e^{-(\beta-\beta_k)V(X_k^{(n)})}\right)^2}{\left(\sum_{n=1}^N e^{-2(\beta-\beta_k)V(X_k^{(n)})}\right)}. \tag{12}$$

Note that `ESS` is a continuous and decreasing function. The value of $\beta_{k+1} > \beta_k$ is numerically found using the bisection method. This process is called `AdaptESS` in lines 2 and 23. In practice the choice of parameter α in eq. 11 is crucial to obtain an efficient schedule. In absence of a theoretically founded rule for selecting this parameter, we propose to fine-tune it for each Neural Network architecture.

6 EXPERIMENTS

This section compares the efficiency of four Sequential Monte Carlo methods:

H-SMC This is our gradient-informed algorithm, *i.e.* the SMC Alg. 1 with the HMC kernel at line 18. The main parameters are T and α, introduced in Sect. 5.

MALA-SMC A variant of our method, where the HMC kernel at line 18 is constrained with $L = 1$ (instead of being adaptively tuned, see Sect. 5.1).

RW-SMC A black-box version of our method, the HMC at line 18 being replaced by a Metropolis-Hastings kernel (Hastings, 1970) with Gaussian proposals Q: $Q(X, s) \triangleq \frac{X+sG}{\sqrt{1+s^2}}$ with $G \sim \mathcal{N}(\mathbf{0}; \mathbf{I})$. The parameter s is called the kernel strength and is tuned adaptively at each iteration. RW stands for "Random Walk".

MLS-SMC This is the Adaptive Multi-Level Splitting SMC used in Webb et al. (2019). It works on a different family of (non-smooth) distributions $\pi_\tau(dx) \triangleq \mathbb{1}(s(x) > \tau)\pi_0(dx)$ and uses a Metropolis-Hastings kernel with uniform noise proposals. The main parameters are T and the survival rate of particles denoted ρ in Webb et al. (2019). Throughout the experiments we use directly their implementation and take $\rho = 0.1$ following the authors choice in their paper.

Algorithm 1 SMC Estimator of the probability of failure

Require: Number of particles N, Generator of i.i.d. reference measure samples $\texttt{Gen}$, Kernel $\texttt{Ker}$, Potential function V, Maximum number of iterations of the kernel T, Maximum number of iterations n_{max}, Minimum relative effective sample size α, Minimum rare event rate r.

Ensure: P_{est}

$X_0 \sim \texttt{Gen}(N)$ ▷ X is a $[N \times d]$ array
$g \leftarrow 1, k \leftarrow 1, \beta_1 \leftarrow \texttt{AdaptESS}(V(X_0), 0, \alpha)$
▷ see Sect. 5.2
while $\frac{1}{N}\sum_{n=1}^{N} \mathbb{1}(V(X_{k-1}^{(n)}) = 0) < r \;\;\&\;\; k \leq n_{max}$ **do**
  $G \leftarrow \exp(-(\beta_k - \beta_{k-1})V(X_{k-1}))$
  $g \leftarrow g \times \frac{1}{N}\sum_{n=1}^{N} G^{(n)}$
  $X_k \leftarrow X_{k-1}$ ▷ Initially copying previous particles
  $U \sim \mathcal{U}([0,1])^{\otimes N}$
  $\mathcal{I}_{killed} \leftarrow \{i : U^{(i)} > (G^{(i)} / \max_{1 \leq n \leq N} G^{(n)})\}$
  ▷ Selection phase
  for i in $\mathcal{I}_{killed}$ **do**
    $X_k^{(i)} \sim \mathcal{U}(\{X_{k-1}^{(j)}\}_{j \notin \mathcal{I}_{killed}})$
  ▷ Resampling particles uniformly amongst survivors
  end for
  $\texttt{StopFlag} \leftarrow \texttt{False}, t \leftarrow 1, Hist_1 \leftarrow \{X_k\}$
  while $t \leq T \;\;\&\;\; \texttt{StopFlag} = \texttt{False}$ **do**
    $X_k \leftarrow \texttt{BatchKer}((X_k^{(n)})_{1 \leq n \leq N}, \beta_k)$
    ▷ Mutations, see Sect. 5.1
    $Hist_{t+1} \leftarrow Hist_t \times \{X_k\}$
    $\texttt{StopFlag} \leftarrow \texttt{AutoCorrCond}(Hist_{t+1})$
    $t \leftarrow t + 1$
  end while
  $\beta_{k+1} \leftarrow \texttt{AdaptESS}(V(X_k), \beta_k, \alpha)$ ▷ see Sect. 5.2
  $k \leftarrow k + 1$
end while
$P_{est} \leftarrow g \times \frac{1}{N}\sum_{n=1}^{N} \mathbb{1}(V(X_{k-1}^{(n)}) = 0)$

Note that the HMC and MALA kernels only work with smooth densities, as those defined in Sect. 4.1. Thus it could not be applied directly to those used in MLS-SMC.
The comparison of H-SMC and RW-SMC allows to single out the efficiency gain of adding the gradient information in the kernel. Comparing H-SMC and MALA-SMC (which are both gradient-based methods) we can evaluate the relative performance of Hamiltonian Monte Carlo vs. Langevin Monte Carlo kernels for robustness estimation.

All variants of our method are fairly adaptive but still require the user to choose parameters α, T and N. In practice we noticed a good trade-off between speed and accuracy for α in the range $[0.8, 0.95]$. Throughout the experience we let N and T vary and compare algorithms based on their accuracy and variance for a given computational cost. All experiments run on a cluster using various GPUs, hence we report the number of calls to the model and its gradient rather than compute times. We count each call to the gradient as two calls to the model function to reflect the cost of the back-propagation. Our code will be available on GitHub.

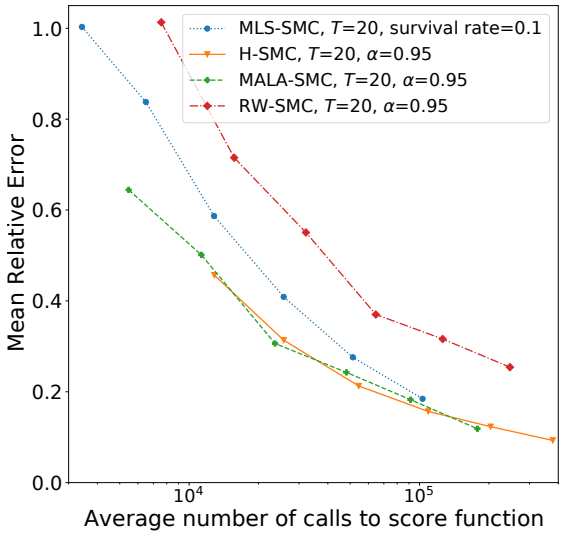


Figure 1: Mean Relative Error *vs.* the average number of calls, for target probability $p_\tau = 10^{-6}$, and number of particles $N \in \{32, 64, 128, 256, 512, 1024\}$. Gradient-informed methods (H/MALA-SMC) outperform black box methods (MLS/RW-SMC) on a linear toy model.

The tendency to underestimate rare event probabilities in Monte Carlo estimation is a well-known phenomenon and shall be interpreted (when assessing the quality of algorithms) as a manifestation of variance due to heavy tail towards larger values.

6.1 Toy Model

We begin our analysis by a sanity check of the algorithms on a simple problem where we know the true probability of failure. We consider a linear model $s(x) = u^\top x - \tau$, where u is a fixed unit norm vector in $\mathbb{R}^d$ and τ a bias such that the true probability is $p_\tau = \Phi(-\tau)$.

We evaluate the quality of the estimation by the Mean Relative Error ($\texttt{MRE}$). For an estimator run n_{run} times, this metric is given by $\texttt{MRE} = n_{run}^{-1} \sum_{i=1}^{n_{run}} |\hat{p}_c^{(i)}/p_\tau - 1|$. In this section, we used $n_{run} = 200$ for all the experiments. All the estimators are run with several numbers of particles. A bigger number provides a better estimation quality while demanding a larger number of calls.

Figure 1 and 2 compares the efficiency of the four algorithms by plotting the Mean Relative Error as a function of the average number of calls. Gradient-informed methods outperform black box methods, especially for lower number of calls to the score function. However for the relatively easier instance of rare event simulation ($p_\tau = 10^{-6}$ on Fig. 1), the difference in performance is smaller and vanishes for a higher number of calls. On the opposite, for the same

problem with a higher threshold ($p_\tau = 10^{-12}$ on Fig. 2), gradient-informed methods clearly outperform black box methods even at higher average number of calls. In particular the RW-SMC algorithm displays bad performance on this harder estimation problem, which shows that integrating gradient information in the proposal kernel can highly increase the efficiency of a SMC method. Finally, on these two problems we do not notice a real performance gap between the H-SMC and the MALA-SMC algorithms. Simply, for a given number of particles N and kernel iteration T the MALA-SMC makes fewer calls to the gradients, since it only makes one call to the gradient function per proposal.

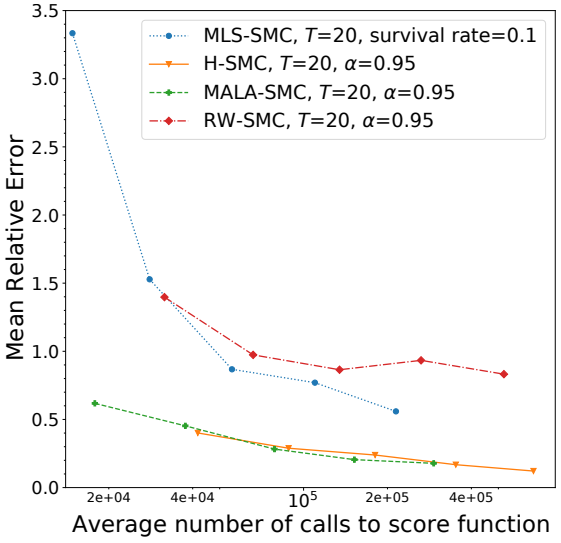


Figure 2: Mean Relative Error *vs.* the average number of calls, for target probability $p_\tau = 10^{-12}$, and number of particles $N \in \{64, 128, 256, 512, 1024\}$. Gradient-informed methods (H/MALA-SMC) outperform black box methods (MLS/RW-SMC) on a linear toy model.

6.2 MNIST

We next compare the aforementioned methods on a robustness estimation problem for a multilayer perceptron trained on the MNIST dataset (LeCun et al., 1990). Its architecture is given in Appendix A. The input dimension is $d = 784$. We consider a uniform perturbation $E \sim \mathcal{U}([-\varepsilon, \varepsilon]^d)$ added to a correctly classified image from the test set. To map the Gaussian distribution, presupposed by our methods, to this uniform distribution, we use the transform $\zeta_\varepsilon : x \mapsto \varepsilon(2\Phi(x) - 1)$, see Sect. 3.1. Note that the MLS-SMC does not use such a transform and works directly with the uniform distribution.

In absence of a ground truth we obtain a reference probability by running an expensive simulation with the H-SMC algorithm taking $N = 4096, T = 200$ and 200 repetitions. We plot it on each figure (as a blue dotted line) for reference only and it is not used to compute any errors. However it can

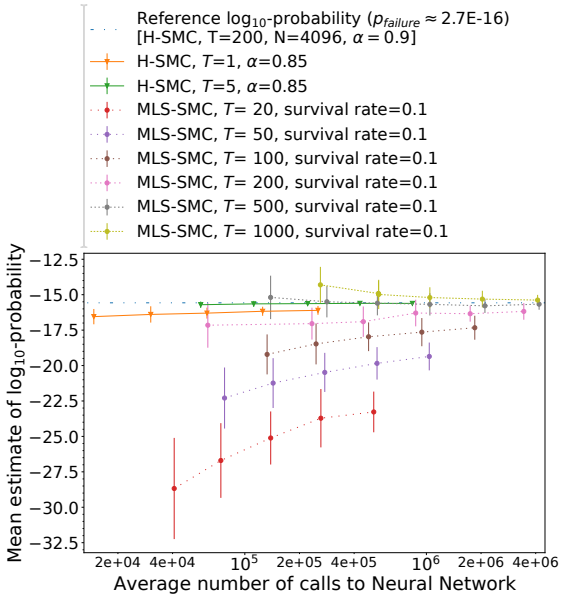


Figure 3: Mean estimate of $\log_{10}(p)$ *vs* Average number of calls, comparing MLS-SMC and H-SMC on MNIST data, for uniform random noise on hyper-rectangle of radius $\varepsilon = 0.15$. Error bars show empirical standard deviations over 100 runs.

be seen that all methods converge to this value, for T large enough, as the number of particles N increases. Figures 3, 4, 5 and 6 show the convergence of estimators for different values of T. Those were obtained with 100 runs and taking $N \in \{32, 64, 128, 256, 512, 1024\}$ (except for $T \geq 200$ in Fig. 3 where N only takes value in $\{8, 16\}$). Figures 3 and 4 correspond to the same value of epsilon ($\varepsilon = 0.15$), whereas Fig. 5 and 6 use a higher value ($\varepsilon = 0.18$).

Looking at figures 3 and 5, one sees that the failure probabilities estimated by MLS-SMC are very low for lower values of T. They increase steadily with T and come closer to the reference but with a higher average numbers of calls. On figure 3, MLS-SMC at last converges to the same mean log probability for $T \geq 500$ whereas the H-SMC only needs $T = 10$ and offers a much smaller standard deviation. This confirms the idea that the slow mixing of the Metropolis-Hastings kernel used by Webb et al. (2019) makes it difficult to estimate low probabilities efficiently. On the contrary, our gradient-informed algorithms (H-SMC and MALA-SMC) seem to benefit from a better mixing kernel.

This is further highlighted on figures 4 and 6, where we compare the convergence of the RW-SMC and H/MALA-SMC. These algorithms are in fact identical except for the transition kernels proposals, which are gradient-informed only for H/MALA-SMC. Figure 4 shows that the RW-SMC estimates are low compared to the reference value for low values of T. Figure 6 corresponds to an easier estimation problem where this negative bias is less evident but gradient-informed again display noticeably lower variances for a

given number of calls to the neural network.

Figures 4 and 6 also allow to compare the convergence of the MALA-SMC and H-SMC algorithms. Figure 4 shows that both algorithms underestimate the probability of failure for $T = 1$, with MALA-SMC having a lower bias. However both algorithms mean estimates are close to the reference probability for $T = 5$, and MALA-SMC is still slightly more efficient than H-SMC. This may seem somewhat surpising as MALA-SMC is only a constrained version of H-SMC. This might mean that our adaptive tuning of the parameter L, see section 5.2, is far from optimal.

6.3 ImageNet

We conclude our empirical analysis with experiments on two convolutional neural networks (CNN) architectures trained on ImageNet (Deng et al., 2009), namely MobileNetV2 (Sandler et al., 2018) and ResNet18 (He et al., 2015). The former architecture contains 53 layers and around 3.4 million parameters, while the latter contains 18 layers and around 11 million parameters. Though rather light-weight models, they still pose an important challenge for probability of failure estimation. Note that no experiment was done over ImageNet in Webb et al. (2019). We consider an additive uniform noise $E \sim \mathcal{U}([-\varepsilon, \varepsilon]^d)$ on two images from the ImageNet validation dataset correctly classified repspectively by the pre-trained MobileNetV2 model and the pre-trained ResNet18 model.

Figures 7 and 8 show the convergence of estimators for

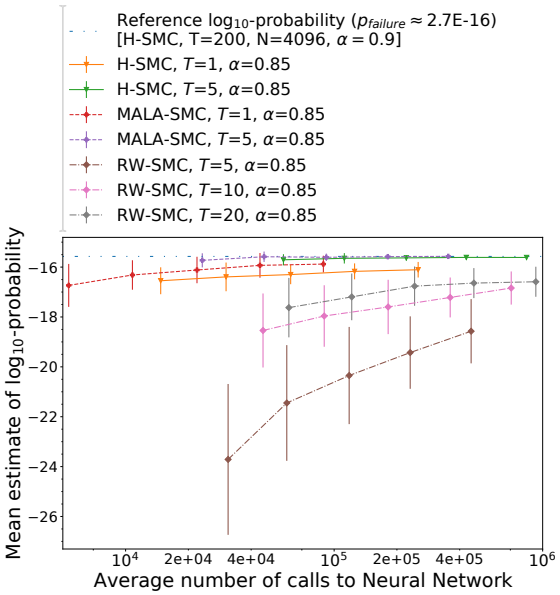


Figure 4: Mean estimate of $\log_{10}(p)$ *vs* Average number of calls, comparing RW-SMC, MALA-SMC and H-SMC on MNIST data, for uniform random noise on hyper-rectangle of radius $\varepsilon = 0.15$. Error bars show empirical standard deviations over 100 runs.

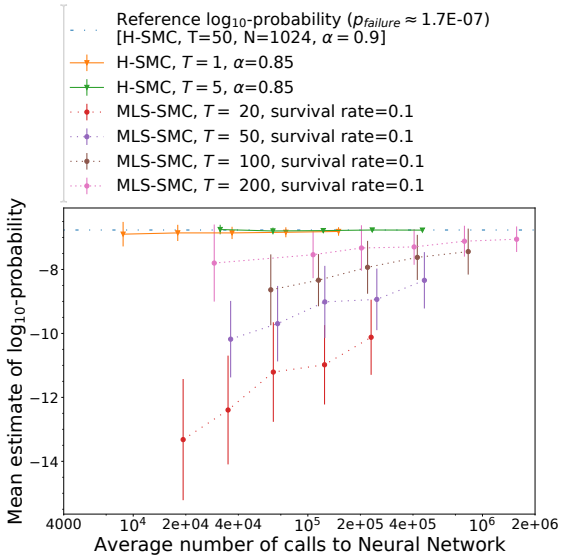


Figure 5: Mean estimate of $\log_{10}(p)$ *vs* Average number of calls, comparing MLS-SMC and H-SMC on MNIST data, for uniform random noise on hyper-rectangle of radius $\varepsilon = 0.18$. Error bars show empirical standard deviations over 100 runs.

different values of T. Those were obtained over 50 runs and taking $N \in \{64, 128, 256, 512\}$. In figure 7 a value for MLS-SMC is missing from the graph for $T = 20, N = 64$ (red-dotted line): this configuration gave a mean estimation of $\log_{10}(p)$ equal to $-\infty$ (a zero probability). For the ResNet18 network the MLS-SMC gave a null estimation of the failure probability, even for large values T and N, thus it is not represented on figure 8. As in MNIST experiments, the MLS-SMC tends to underestimate the probability of failure, especially for low values of T. Similarly, we see on figure 7 that the RW-SMC clearly underestimates the probability of failure. On figure 8, the negative bias of the RW-SMC is less striking, however it displays higher variance than the H/MALA-SMC. This highlight a lower efficiency of kernels based on Random Walk proposals (RW-SMC and MLS-SMC) in comparison to the gradient-informed kernels used in the H-SMC and MALA-SMC algorithms.

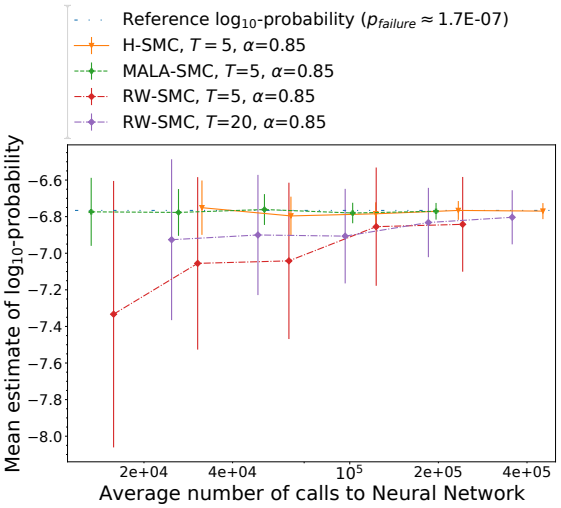


Figure 6: Mean estimate of $\log_{10}(p)$ *vs* Average number of calls, comparing RW-SMC, MALA-SMC and H-SMC on MNIST data, for uniform random noise on hyper-rectangle of radius $\varepsilon = 0.18$. Error bars show empirical standard deviations over 100 runs.

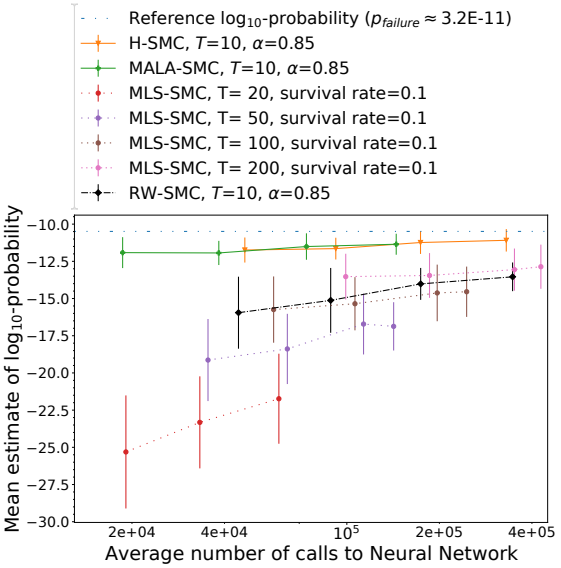


Figure 7: Mean estimate of $\log_{10}(p)$ vs Average number of calls to MobileNetV2, with uniform noise over a hyper-rectangle of radius $\varepsilon = 0.13$, on ImageNet data. Error bars represent empirical standard deviation over 50 runs.

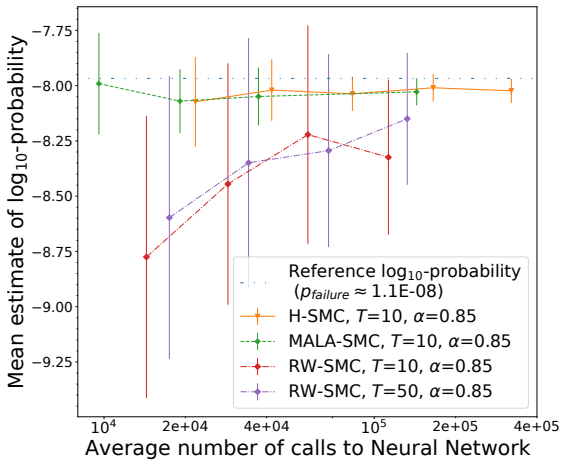


Figure 8: Mean estimate of $\log_{10}(p)$ vs Average number of calls to ResNet18, with uniform noise over a hyper-rectangle of radius $\varepsilon = 0.01$, on ImageNet data. Error bars represent empirical standard deviation over 50 runs.

7 LIMITATIONS

Our gradient-informed method can be applied in principle to any machine learning model that is (almost everywhere) differentiable w.r.t. its input. However, unlike black-box methods, it cannot be applied to other machine learning models such as random forests and k-nearest neighbors algorithms. An other issue is that, as for the method from Webb et al. (2019), we are currently limited to asymptotic guarantees, see Prop. 1. Deriving non-asymptotic statistical guarantees, as provided in Tit et al. (2021), is left for future work. Yet another point of contention is the use of Effective Sample Size metric for inverse temperature scheduling as presented in 5.2. In deed, as pointed recently by Elvira et al. (2022), this metric is based on an approximation relying itself on multiple assumptions which hard to verify. In practice we found that it gives good empirical results, however finding a better theoretically-founded scheduling method is an interesting direction for future research.

8 SOCIETAL IMPACT

With the development of DNN-based cyber-physical systems, a rigorous quantitative assessment of their reliability becomes of utmost importance. This paper has proposed and evaluated examples of state-of-the-art Monte Carlo methods to perform this task, especially to quickly detect unsatisfactory classifiers whose probabilities of failure are too large.

9 CONCLUSION

This paper shows how to estimate robustness to random noise in neural network classifiers more efficiently using a gradient-informed sampling technique. Our technique plugs MALA and Hamiltonian Monte Carlo methods within Sequential Monte Carlo to this rare event probability estimation problem. The performance of these gradient-informed methods is first checked on a linear toy model. We then compare these methods to a state-of-the-art black-box method for robustness estimation, and a black-box variant of our algorithm, on deep neural network models trained with MNIST and ImageNet data. We observe a faster mixing of our algorithm compared to black box methods, which in turn leads to more efficient estimation highlighted by lower variances for a given number of calls to the Neural Network functions. Future works include the investigation of novel non-asymptotic guarantees on error rates, and the extension of our methods to statistical hypothesis testing.

Acknowledgements

We thank French ANR and AID agencies for funding Chaire SAIDA ANR-20-CHIA-0011-01. KT gratefully acknowledges PhD funding via a grant from the AID (Agence pour l'Innovation de Défense).

References

Mansur Arief, Zhiyuan Huang, Guru Koushik Senthil Kumar, Yuanlu Bai, Shengyi He, Wenhao Ding, Henry Lam, and Ding Zhao. Deep probabilistic accelerated evaluation: A robust certifiable rare-event simulation methodology for black-box safety-critical systems. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 595–603. PMLR, 13–15 Apr 2021. URL https://proceedings.mlr.press/v130/arief21a.html.

Teodora Baluta, Zheng Leong Chua, Kuldeep S. Meel, and Prateek Saxena. Scalable quantitative verification for deep neural networks. In *Proc. of Int. Conf. on Software Engineering*, 2021.

Alexandros Beskos, Ajay Jasra, Nikolas Kantas, and Alexandre Thiery. On the convergence of adaptive sequential monte carlo methods. *The Annals of Applied Probability*, 26(2):1111–1146, 2016.

Alexander Buchholz, Nicolas Chopin, and Pierre E Jacob. Adaptive tuning of hamiltonian monte carlo within sequential monte carlo. *Bayesian Analysis*, 16(3):745–771, 2021.

P. Del Moral. *Feynman-Kac Formulae, Genealogical and Interacting Particle Systems with Applications*. Probability and its Applications. Springer, 2004.

P. Del Moral, A. Doucet, and A. Jasra. Sequential Monte-Carlo samplers. *J. Roy. Stat. Soc. B*, 68(3):411–436, 2006.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Víctor Elvira, Luca Martino, and Christian P. Robert. Rethinking the Effective Sample Size. *International Statistical Review*, 90(3):525–550, December 2022. doi: 10.1111/insr.12500. URL https://ideas.repec.org/a/bla/istatr/v90y2022i3p525-550.html.

Paul Fearnhead and Benjamin Taylor. An adaptive sequential monte carlo sampler. *Bayesian Analysis*, 8, 05 2010. doi: 10.1214/13-BA814.

Alessandro Foi, Mejdi Trimeche, Vladimir Katkovnik, and Karen Egiazarian. Practical poissonian-gaussian noise modeling and fitting for single-image raw-data. *IEEE Transactions on Image Processing*, 17(10):1737–1754, 2008. doi: 10.1109/TIP.2008.2001399.

Jean-Yves Franceschi, Alhussein Fawzi, and Omar Fawzi. Robustness of classifiers to uniform ℓ_p and gaussian noise, 2018.

Justin Gilmer, Nicolas Ford, Nicholas Carlini, and Ekin Cubuk. Adversarial examples are a natural consequence of test error in noise. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2280–2289. PMLR, 09–15 Jun 2019. URL http://proceedings.mlr.press/v97/gilmer19a.html.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. URL http://arxiv.org/abs/1412.6572.

Arnaud Guyader, Nicolas Hengartner, and E. Matzner-Løber. Simulation and estimation of extreme quantiles and extreme probabilities. *Applied Mathematics & Optimization*, 64:171–196, 10 2011. doi: 10.1007/s00245-011-9135-z.

W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57 (1):97–109, 1970. ISSN 00063444. URL http://www.jstor.org/stable/2334940.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HJz6tiCqYm.

Kei Ishikawa and Takashi Goda. Efficient debiased evidence estimation by multilevel monte carlo sampling. In *Uncertainty in Artificial Intelligence*, pages 34–43. PMLR, 2021.

Herman Kahn and Theodore E Harris. Estimation of particle transmission by random sampling. *National Bureau of Standards applied mathematics series*, 12:27–30, 1951.

Guy Katz, Clark 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 Kunčak, editors, *Computer Aided Verification*, pages 97–117, Cham, 2017. Springer International Publishing. ISBN 978-3-319-63387-9. URL https://arxiv.org/abs/1312.6199.

Philip Koopman and Michael Wagner. Autonomous vehicle safety: An interdisciplinary challenge. *IEEE Intelligent Transportation Systems Magazine*, 9(1):90–96, 2017. doi: 10.1109/MITS.2016.2583491.

Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, R. Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1990. URL https://proceedings.neurips.cc/paper/1989/file/53c3bce66e43be4f209556518c2fcb54-Paper.pdf.

Radford M. Neal. *Handbook of Markov Chain Monte Carlo*, chapter MCMC using Hamiltonian dynamics. Chapman and Hall/CRC, may 2011. doi: 10.1201/b10905. URL https://doi.org/10.1201/b10905.

Gareth O Roberts and Richard L Tweedie. Exponential convergence of langevin distributions and their discrete approximations. *Bernoulli*, pages 341–363, 1996.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. doi: 10.1109/CVPR.2018.00474.

Karim Tit, Teddy Furon, and Mathias Rousset. Efficient Statistical Assessment of Neural Network Corruption Robustness. In *NeurIPS 2021 - 35th Conference on Neural Information Processing Systems*, volume 34 of *Advances in Neural Information Processing Systems proceedings*, Sydney (virtual), Australia, December 2021. URL https://hal.archives-ouvertes.fr/hal-03407011.

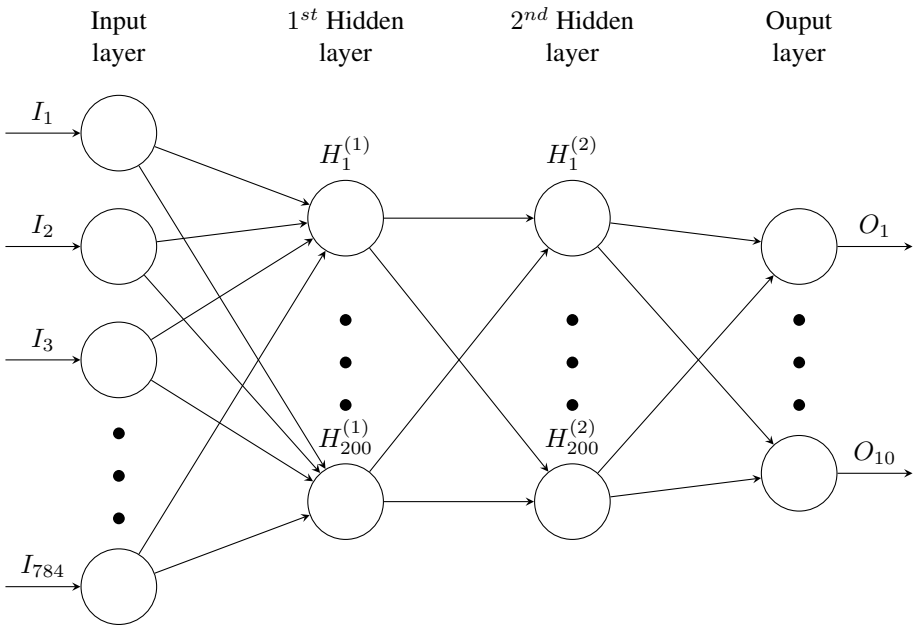
Benjie Wang, Stefan Webb, and Tom Rainforth. Statistically robust neural network classification. In *Uncertainty in Artificial Intelligence*, pages 1735–1745. PMLR, 2021.

Stefan Webb, Tom Rainforth, Yee Whye Teh, and M Pawan Kumar. A statistical approach to assessing neural network robustness. In *International Conference on Learning Representations*, 2019.

Lily Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane Boning, and Inderjit Dhillon. Towards fast computation of certified robustness for ReLU networks. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5276–5285. PMLR, 10–15 Jul 2018. URL http://proceedings.mlr.press/v80/weng18a.html.

A NEURAL ARCHITECTURE USED FOR MNIST EXPERIMENTS

The architecture used is a fully connected neural network with 2 hidden layers, each containing 200 units. It is illustrated below. Each hidden unit outputs a linear combination of it's inputs, composed with the ReLU activation, defined by: $ReLU(x) = x\mathbb{1}(x \geq 0)$.



B PROOFS

Proposition 1 is classical in the field of Sequential Monte Carlo. A complete mathematical treatment of such algorithms can be found in Del Moral (2004), in which not only unbiasedness and consistency is proved, but a also a central limit theorem, some concentration estimates, and other properties for $N \to +\infty$.

B.1 Proof of Proposition 1

First, we show that the selection step is indeed sufficient to obtain an unbiased estimation. Let $\pi^N_{\beta_k} = \frac{1}{N}\sum_{n=1}^N \delta_{X_k^{(n)}}$ denote the empirical distribution of particles at iteration k, and denote by $B_{k+1}^{(n)}$ the number of offsprings of the particle $X_k^{(n)}$ after the selection step. By construction of the selection step, in which surviving particles are duplicated uniformly, one has

$$\begin{aligned}\mathbb{E}\left[B_{k+1}^{(n)} \mid (X_k^{(1)}, \dots, X_k^{(N)})\right] &= \mathbb{P}\left[(n) \text{ is killed}\right] \times \tilde{C}_k^{-1}, \\ &= e^{-(\beta_{k+1}-\beta_k)V(X_k^{(n)})} C_k^{-1}\end{aligned}$$

where C_k and $\tilde{C}_k$ are constants independent of n, and the expectation is taken conditional on the particles $(X_k^{(1)}, \dots, X_k^{(N)})$. The latter constants can be simplified by remarking that since the number of particles is kept equal to N, then $\sum_n B_{k+1}^{(n)} = N$, which implies

$$C_k = \frac{1}{N}\sum_{n=1}^N e^{-(\beta_{k+1}-\beta_k)V(X_k^{(n)})} = \frac{\widehat{Z}_{k+1}}{\widehat{Z}_k}.$$

As a consequence, since just after the selection step one has

$$\frac{1}{N}\sum_{n=1}^N \delta_{X_{k+1}^{(n)}} = \frac{1}{N}\sum_{n=1}^N B_{k+1}^{(n)} \delta_{X_k^{(n)}},$$

the following key unbiasedness (consistency) is obtained – the expectation being again conditional on the particles $(X_k^{(1)}, \ldots, X_k^{(N)})$ at iteration k: For any test function t,

$$\widehat{Z}_{k+1}\mathbb{E}\left[\mathbb{E}_{\pi^N_{\beta_{k+1}}}\left[t(X)\right] \mid (X_k^{(1)}, \ldots, X_k^{(N)})\right] = \widehat{Z}_k \mathbb{E}_{\pi^N_{\beta_k}}\left[e^{-(\beta_{k+1}-\beta_k)V(X)}t(X)\right]. \tag{13}$$

Second, by induction on k with initialization for $k = 0$ given by $Z_0\mathbb{E}[\mathbb{E}_{\pi_0^N}\left[t(X)\right]] = \mathbb{E}_{\pi_0}(t(X))$, we obtain that (13) is unbiased in the sense that its full expectation is indeed given by $Z_{k+1}E_{\pi_{\beta_{k+1}}}\left(t(X)\right)$.

Third, the mutation step (which is crucial to enforce independence between particles) does not modify the above argument by induction on unbiasedness. Indeed, at each iteration k, the kernel is applied to each particle before the selection step of iteration $k+1$, and it is chosen chosen to leave invariant the distribution π_{β_k} (see Sect. 5.1). As a consequence, the full expectation of the right hand side of (13) – which consists of mutated particles – is unchanged.

In the end the quantity $\widehat{Z}_k$ is indeed an unbiased estimator of Z_k for each k. Induction on k can also be applied to the weak law of large number to obtain consistency at each iteration. This concludes the proof of Proposition 1.