Évaluation statistique efficace de la robustesse de classifieurs

Karim Tit, Teddy Furon, Mathias Rousset, Louis-Marie Traonouez

Évaluation statistique efficace de la robustesse de classifieurs ⋆

Karim TIT[1,2], Teddy Furon[1], Mathias Rousset[1], and Louis-Marie Traonouez[2]

[1] INRIA/IRISA, LinkMedia & SimSmart Teams, Rennes, France
{karim.tit,teddy.furon,mathias.rousset}@inria.fr
[2] Thales Land & Air Systems, La Ruche, Rennes, France
{karim.tit,louis-marie.traonouez}@thalesgroup.com

Abstract. Nous proposons de quantifier la robustesse d'un classifieur aux incertitudes d'entrée avec une simulation stochastique. L'évaluation de la robustesse est présentée comme un test d'hypothèse : le classifieur est considéré comme localement robuste si la probabilité de défaillance estimée est inférieure à un niveau critique. La procédure est basée sur une simulation d'Importance Splitting générant des échantillons d'événements rares. Nous dérivons des garanties théoriques non-asymptotiques par rapport à la taille de l'échantillon. Des expériences portant sur des classifieurs à grande échelle mettent en évidence l'efficacité de notre méthode.

Keywords: Apprentissage profond · Robustesse · Monte Carlo séquentiel

1 Introduction

Malgré des performances de pointe dans de nombreuses tâches de vision par ordinateur et de traitement automatique des langues, les réseaux neuronaux profonds (DNN) se sont révélés sensibles aux perturbations aléatoires et adverses [6,5].

Certification et évaluation de robustesse. La certification a posteriori vérifie le comportement correct d'un réseau entraîné $f : \mathbb{R}^n \to \mathbb{R}^m$. La propriété attendue est généralement définie localement : le réseau fonctionne correctement dans le voisinage $\mathcal{V}(\mathbf{x}_o) \subset \mathbb{R}^n$ d'une entrée particulière $\mathbf{x}_o \in \mathbb{R}^n$. En classification, la propriété prend le nom de *robustesse* et se lit comme suit : la sortie du réseau reste inchangée sur le voisinage $\mathcal{V}(\mathbf{x}_o)$. Cela certifie que le réseau est robuste aux incertitudes de support limité ou des perturbations adverses de distorsion contrainte. Le mécanisme de certification présente deux caractéristiques :

- Consistance : il ne certifie pas le réseau lorsque la propriété ne tient pas.
- Complétude : il certifie toujours le réseau lorsque la propriété est vérifiée.

Robustesse de corruption. La robustesse adversariale correspond à une analyse au pire cas, tandis que la *robustesse aux incertitudes* considère les perturbations aléatoires des entrées. L'ingrédient clé est l'introduction d'un

⋆ Thèse financée par l'Agence de l'Innovation de Défense et Thales

modèle statistique π_0 des incertitudes épistémiques survenant le long de la chaîne d'acquisition de l'entrée. Par exemple, [5] prend des distributions Gaussiennes ou uniformes sur la boule $\mathcal{B}_{p,\epsilon}(\mathbf{x}_o)$ de rayon ϵ en norme ℓ_p centrée sur $\mathbf{x}_o$. Ils obtiennent des limites précises pour les classificateurs linéaires qu'ils étendent aux classificateurs non linéaires tels que des réseaux neuronaux profonds en supposant leurs "frontières de décision localement approximativement plates". L'approche présentée ici ne nécessite pas une telle hypothèse sur les classificateurs DNN.

Cette tendance récente s'accompagne d'une évaluation *quantitative* déterminant dans quelle mesure une propriété donnée est ou n'est pas présente. Par exemple, [13] estime la probabilité p qu'une propriété soit violée sous un modèle statistique donné des entrées. Cette approche n'a besoin d'aucune hypothèse sur le réseau examiné car il est utilisé comme une boîte noire. Elle peut donc s'appliquer aux réseaux profonds. La principale difficulté réside dans l'efficacité, c'est-à-dire la puissance de calcul nécessaire pour estimer les probabilités faibles. Leur manque de solidité provient de l'incapacité à déterminer si la probabilité p de violation est exactement nulle ou trop faible pour être estimée.

La section 2 présente un bref aperçu des procédures de certification en soulignant les hypothèses faites sur le réseau et leurs limites.

Ce travail présente une procédure efficace et passant à l'échelle pour évaluer la robustesse à la corruption sous un large panel de modèles statistiques. Il fournit une complétude et des garanties théoriques sur le manque de consistance.

2 Etat de l'art en matière d'évaluation de robustesse

Évaluation par l'exemple. Les attaques adverses avec contrainte de distorsion, comme l'attaque de descente de gradient projeté (PGD)[3], recherchent des violations de propriétés, c'est-à-dire des exemples adverses, à l'intérieur de la boule $\mathcal{B}_{p,\epsilon}(\mathbf{x}_o)$. Elles tirent parti du calcul rapide du gradient de la fonction du réseau grâce à la rétro-propagation. Ils sont rapides, mais ni consistants ni complets. Le réseau n'est pas certifié si l'attaque réussit, mais un échec ne dit rien sur la propriété : les attaques sont des processus empiriques sans garantie.

Certification formelle. En utilisant un solveur SMT (*Satisfiability Modulo Theories*), ReLUplex [8] fournit une méthode de certification consistante et complète, conçue pour les réseaux neuronaux avec des fonctions d'activation ReLU. Cependant, le même article montre que le problème de la certification consistante et complète des réseaux neuronaux (même restreint aux activations ReLU) est NP-complet. Le passage à l'échelle des grands réseaux modernes semble difficile. De plus, bien que ces méthodes formelles soient complètes en théorie, dans la pratique, la procédure peut abandonner ou se terminer de manière indécise avec un 'timeout' si le solveur sous-jacent est trop lent.

Certification incomplète. Pour passer à l'échelle, certains proposent des méthodes de vérification solides mais incomplètes par conception, en recourant à des approximations convexes. Le papier [12] obtient une accélération significative par rapport à ReLUplex et à d'autres certificateurs complets. Il introduit un benchmark de vérification appelé ERAN (voir Sect. 4). [15] présente une autre

certification incomplète basée sur des bornes fonctionnelles linéaires inférieures et supérieures de perceptrons multicouches (MLP) avec activation ReLU. Elle est généralisée aux MLP avec une fonction d'activation quelconque dans [16] et aux réseaux de neurones à convolution (CNNs) dans [2,14].

Ces méthodes de certification reposent sur des bornes inférieures de la distance minimale des exemples adverses. Elles sont donc pessimistes dans le sens où elles peuvent rejeter de nombreuses propriétés valides car la borne inférieure n'est pas toujours assez fine. Afin d'être "plus complet", [11] unifie ces méthodes de relaxation dans un cadre général et résout exactement la relaxation convexe optimale (pour des problèmes spécifiques sur CIFAR10 et MNIST) avec des ressources de calcul importantes. Les auteurs ont noté q'une légère amélioration de la précision des bornes inférieures par rapport à l'état de l'art, ce qui suggère que cette approche a atteint sa limite.

Évaluation statistique. La robustesse à la corruption suppose un modèle statistique π_0 des entrées comme les distributions Gaussiennes ou uniformes sur la boule $\mathcal{B}_{p,\epsilon}(\mathbf{x}_o)$. Le papier [5] étudie la robustesse des réseaux de neurones linéaires et profonds. Ils obtiennent des limites précises pour les classifieurs linéaires qu'ils étendent aux classifieurs non linéaires avec des "frontières de décision localement approximativement plates". Le travail [13] définit la robustesse par la probabilité de défaillance suivante (meilleure d'autant que cette valeur est petite) :

$$p := \pi_0(\iota(\mathbf{X}|\mathbf{x}_o) = 1) = \int_{\mathbb{R}^n} \iota(\mathbf{x}|\mathbf{x}_o)\pi_0(d\mathbf{x}), \tag{1}$$

où $\iota(\cdot|\mathbf{x}_o)$ est la fonction indicatrice d'une défaillance. Cette évaluation statistique contraste fortement avec la littérature sur la robustesse adversariale qui adopte une analyse au pire cas. Un lien est établi lorsque π_0 est la distribution uniforme sur la boule $\mathcal{B}_{p,\epsilon}(\mathbf{x}_o)$: le volume de l'ensemble des exemples adverses est égal à $p \cdot \mathsf{vol}(\mathcal{B}_{p,\epsilon}(\mathbf{x}_o))$. Cette probabilité p est parfois appelée la "densité adverse" [1].

La principale difficulté réside dans l'estimation de cette intégrale, en particulier lorsque l'événement $\{\iota(\mathbf{X}|\mathbf{x}_o) = 1\}$ est rare sous la distribution π_0. Le papier [1] utilise une simulation de Monte Carlo peu efficace, [13] le fractionnement multi-niveaux (en anglais *multi-level splitting*) avec un mécanisme de rajeunissement basé sur l'algorithme de Metropolis-Hastings. Ces deux derniers travaux ne font aucune hypothèse sur le réseau car leurs procédures l'utilisent comme une boîte noire. Cela garantit le passage à l'échelle (dans le sens où elle s'applique aux réseaux profonds). L'efficacité du test statistique est mesurée par le temps d'exécution ou le nombre d'appels à la boîte noire.

L'évaluation quantitative revient à la certification en prenant une décision finale : le réseau est *réputé* correct si la probabilité de violation est inférieure à $p_c > 0$, une probabilité critique fixée par l'utilisateur.

3 L'évaluation de robustesse comme un test d'hypothèse

Notre approche utilise le test d'hypothèse statistique comme un ersatz de la certification. Comme dans [1], l'utilisateur fixe une faible probabilité critique p_c

et le test évalue si la probabilité de défaillance p est inférieure ou supérieure. Nous utilisons à ce moyen la simulation de la "dernière particule". Cette simulation dite de la "dernière particule" a été inventée par A. Guyader *et al.* [7]. C'est une variante efficace de l'échantillonnage multi-niveaux adaptatif employé par [13]. Nous montrons qu'avec une condition de terminaison bien choisie, cet algorithme est avantageux à la fois en termes d'efficacité et de garanties théoriques.

La section 3.1 présente l'algorithme "dernière particule" (*cf.* Alg. 1 en appendice), un classique dans le domaine de la simulation d'événements rares. La section 3.2 fait le lien entre la certification et les tests d'hypothèses statistiques dans le cadre de l'évaluation de la robustesse.

3.1 La simulation de la "dernière particule"

L'objectif de la simulation de la "dernière particule" est de générer efficacement des échantillons tirés aléatoirement selon une distribution de référence π_0 mais dans une région $\mathcal{R} := \{\mathbf{y} : h(\mathbf{y}) > 0\} \subset \mathbb{R}^n$, où $h : \mathbb{R}^n \to \mathbb{R}$, est la fonction dite *score*. Son efficacité est la capacité d'effectuer cette tâche en utilisant peu d'appels à la fonction de score, même lorsque la probabilité $\pi_0(\mathcal{R})$ est faible.

La simulation gère un ensemble de N particules (*i.e.* échantillons) qui sont i.i.d. par rapport à π_0. Le nom "dernière particule" vient du fait que la simulation 'élimine' l'échantillon dont le score est le plus bas. Son score donne la valeur du niveau intermédiaire L_k à l'itération k (Alg. 1, ligne 6). Ensuite, cette particule est rafraîchie par échantillonnage selon π_0 mais conditionnée à l'événement $\{h(\mathbf{X}) > L_k\}$. Cette procédure d'échantillonnage est effectuée par la procédure $\mathsf{Gen}(L_k, 1)$ (Alg. 1, ligne 11) détaillée dans l'Alg. 2. $\mathsf{Gen}(-\infty, N)$ signifie alors simplement échantillonner N vecteurs aléatoires selon π_0 (ligne 3).

3.2 Lien avec la certification

Le test évalue si la probabilité de défaillance p est inférieure ou supérieure à une valeur critique p_c. Cela se fait en constatant que la simulation a atteint un nombre maximum d'itérations m (détaillé ci-après). L'algorithme s'arrête lorsque

- le nombre k d'itérations atteint l'entier m. Alors le réseau est certifié car on pense que $p < p_c$.
- le seuil intermédiaire $L_k > 0$ pour une itération $k < m$. Cela signifie que la simulation a généré quelques échantillons provoquant une défaillance. Alors le réseau n'est pas certifié.

Une certification peut être erronée pour deux raisons:

- La probabilité de défaillance est trop faible telle que $0 < p < p_c$. Cet écueil est évité en prenant une probabilité critique plus faible, mais cela est plus coûteux en temps de simulation.
- La probabilité de défaillance $p > p_c$, et un tel cas aurait du conduire à $L_k > 0$ pour un certain $k < m$. Mais comme la simulation est aléatoire, il y a une probabilité α que cette erreur se réalise.

On peut montrer que la relation entre le nombre d'itérations maximum m et le cahier des charges donné par les paramètres (p_c, α) est tel que :

Le quantile associé à la probabilité α pour la distribution $\Gamma(m, N)$ égale à $-\log p_c$.

Ainsi, l'entier m est une fonction décroissante de α pour p_c fixée, et évolue en $O(\log 1/p_c)$ pour α donné. Autrement dit, passer de $p_c = 10^{-30}$ à 10^{-60} multiplie m par deux et ainsi le temps de simulation. On voit ainsi clairement l'avantage de "dernière particule" par rapport à une simulation Monte Carlo.

4 Investigation expérimentale

Pour évaluer notre approche nous étudions un dispositif breveté [4] de surveillance aérienne qui permet l'identification du type d'aéronef à partir de données cinématiques des trajectoires. Ce dispositif s'intègre à des systèmes de contrôle qui collectent les informations issues de capteurs afin de suivre les trajectoires (modules de *tracking*) et de classer le type des trajectoires.

Dans cette étude nous analysons un jeu de données de trajectoires issues de capteurs ADSB (*Automatic Dependent Surveillance Broadcast*). Ce système équipe une majorité du trafic aérien, dont la totalité des vols commerciaux. Contrairement aux radars, c'est un système passif qui se repose sur le transpondeur interne aux aéronefs qui émet en continue les informations (non chiffrées) sur la position et l'identification de l'aéronef. Ce système permet de constituer facilement une base de données 'labelisée' pour entraîner et évaluer le dispositif de classification. Par ailleurs, l'intégration de ces données ouvertes aux systèmes de contrôle militaires est de plus en plus étudiée, mais il est préalablement nécessaire d'en vérifier la cohérence. Ce dispositif de classification pourrait par exemple détecter un aéronef qui falsifierait son identification.

Le dispositif d'apprentissage automatique étudié améliore la capacité de classification du type d'aéronef des systèmes de contrôle en calculant des caractéristiques cinématiques de la trajectoire. Pour chaque point sur la trajectoire, on calcule 9 caractéristiques, dont des mesures de vitesse, d'accélération, de courbure et de torsion en 2 ou 3 dimensions. Deux approches sont alors utilisées pour effectuer une classification des trajectoires. La première consiste à calculer pour chaque trajectoire et pour chacune des caractéristiques des mesures statistiques sur les valeurs des séries temporelles (valeurs minimales, maximales, 4 premier moments, quantiles). La seconde analyse les trajectoires entières de longueur variable à l'aide de réseaux de neurones récurrents.

Le jeu de données utilisé dans ce papier contient 26609 trajectoires de longueurs variables (de 26 à 1610 points). Ces trajectoires sont réparties en 6 classes selon le type d'aéronef (avion ou hélicoptère), la taille, le nombre et le type de moteur. La classe majoritaire contient 21529 trajectoires, la classe minoritaire 169. Le jeu de données est partitionné en un jeu d'entraînement de 21287 trajectoires et un jeu de validation de 5322 trajectoires.

Les expériences illustrent l'algorithme de la dernière particule présenté en section 3 d'une part et le système de certification formelle ERAN avec la méthode

Table 1. Données ADSB statiques – Comparaison ERAN [DeepPoly], Last Particle $[N = 2, p_c = 10^{-10}, t = 40]$ et Monte Carlo simple $[N = 10^6, p_c = 10^{-10}]$. Moyennes sur 11 modèles de réseaux de neurones. Temps de vérification pour 100 trajectoires.

	ERAN		Last Particle		Monte Carlo simple	
ε	Certifié (%)	temps (sec. ± std)	Validé (%)	temps (sec. ± std)	Validé (%)	temps (sec. ± std)
0.0001	100	5.0 ± 5.0	100	5.26 ± 0.1	100	7.74 ± 0.13
0.0005	100	5.01 ± 5.07	100	5.26 ± 0.10	100	8.19 ± 0.3
0.001	99	5.03 ± 5.06	100	5.26 ± 0.08	100	8.14 ± 0.27
0.005	98	4.91 ± 5.1	99	5.28 ± 0.11	99.8	7.86 ± 0.17
0.01	95	4.97 ± 5.2	98	5.21 ± 0.10	99	8.0 ± 0.21
0.05	20	6.88 ± 7.8	61	4.82 ± 0.3	89	7.74 ± 0.28
0.1	0.05	6.95 ± 8.33	43	4.0 ± 0.5	64	8.12 ± 0.2

DeepPoly [12] d'autre part. Les expériences ont été réalisées avec une carte graphique NVIDIA V100 et un processeur Intel Xeon Processor E5-2698 v4.

4.1 Expériences sur données ADSB statiques

Dans ces expériences, on analyse le jeu de données au format tabulaire comportant 72 caractéristiques. On entraîne pour commencer 3 modèles de réseaux de neurones avec respectivement 1 couche dense de 100, 500 ou 1000 neurones, et 1 modèle avec 3 couches denses de 500, 100 et 50 neurones. On compare les taux de certification d'ERAN et les taux de validation de la méthode Last Particle et d'un algorithme Monte Carlo naïf (avec 10^6) dans le tableau 1. Sur ces modèles de petites tailles on voit que la méthode DeepPoly d'ERAN a des temps similaires à notre procédure. Cependant, la variance des temps en fonctions des modèles et des trajectoires est plus élevé pour ERAN. Par comparaison, il est possible de borner facilement à l'avance le nombre d'appels fait au classifieurs que l'algorithme Last Particle. Par ailleurs pour des valeurs élevées du paramètre ε on voit que le temps de vérification du système DeepPoly augmente alors que notre méthode a tendance à accélérer avec ε croissant. Enfin, notons que jusqu'à un certain niveau de distortion l'algorithme de la dernière particule et ERAN donne les mêmes résultats et que ceux-ci divergent seulement pour des valeurs élevés d'ε. Cette divergence s'explique d'ailleurs par la différence d'objectif: ERAN ne certifie que s'il n'existe aucune violation d'une sous-région donnée, tandis que notre méthode doit simplement vérifier que la probabilité d'échec dans cette même sous-région est assez faible.

Un des avantages de la méthode Last Particle proposée dans ce papier est qu'elle s'applique en boîte noire, indifféremment du type de modèle pour peu que l'on puisse définir une fonction score continue. Nous pouvons ainsi l'appliquer sur des modèles d'ensemble d'arbres de décision tels que des Random Forest ou du Gradient Boosting. On voit que le temps de vérification augmente globlement

Table 2. Données ADSB statiques – Analyse de modèles Random Forest (RF) et Gradient Boosting (GB) avec Last Particle [$N = 2, p_c = 10^{-10}$,$t = 40$]. Moyenne pour $\varepsilon \in \{0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1\}$.

	RF		GB	
Nb. estimateurs	Validé (%)	temps (sec. ± std)	Validé (%)	temps (sec. ± std)
100	87.6	7.38 ± 0.004	73.6	11.9 ± 0.08
500	89.6	15.43 ± 0.065	76.2	32.98 ± 0.35
1000	90.6	28.55 ± 0.10	78.5	25.42 ± 0.09

Table 3. Données ADSB statiques – Entraînement adverse et entraînement stochastique sur un réseau de neurone. Comparaison de ERAN [DeepPoly] et Last Particle [$N = 2, p_c = 10^{-10}$,$t = 40$]. Temps de vérification pour 100 trajectoires.

	ERAN	Last Particle	
Modèle	Certifié (%)	Validé (%)	faux positifs (%)
Sans adv. training	73.14	86.29	13.14
adv. training ($\varepsilon = 0.025$, norme: l_2)	73.14	87.0	13.85
adv training ($\varepsilon = 0.025$, norme: l_∞)	73.57	96.0	22.42
random training ($\varepsilon = 0.01$, norme: l_2)	72.71	91.42	18.71
random training ($\varepsilon = 0.01$, norme: l_∞)	72.71	91.14	18.42

avec le nombre d'estimateurs du modèle d'ensemble. Cette augmentation en temps est cependant moins que linéaire (e.g. pour les forêts aléatoires, avec 10 fois plus d'estimateurs, le temps de calcul est seulement quadruplé). Le tableau 2 présente les résultats de la méthode Last Particle sur 6 modèles d'ensemble de tailles croissantes.

Nous présentons une dernière expérience qui compare le taux de certification de modèles de réseaux de neurones entraînés à l'aide de techniques de 'robustification' telles que l'entraînement adverse [9] et l'entraînement adverse stochastique [10]. Le tableau 3 montre que la certification formelle est quasiment insensible à ces techniques, alors que notre procédure détecte une amélioration de la robustesse.

4.2 Expériences sur données ADSB dynamiques

Cette section présente les résultats d'expériences sur des modèles utilisant l'approche dynamique du dispositif de classification, qui consiste à analyser les séries temporelles des trajectoires. Les données utilisées sont donc des trajectoires de longueurs variables comportant à chaque instant 9 caractéristiques. On entraîne pour commencer un premier modèle comportant des couches de convolutions avec un total de 289030 neurones. Le tableau 4 présente les résultats de certification de ce modèle avec ERAN et Last Particle.

Table 4. Données ADSB dynamiques – Comparaison ERAN [DeepPoly] et Last Particle [$N = 2, p_c = 10^{-10}$,$t = 40$] pour un réseau de neurones convolutif profond. Temps de vérification pour 100 trajectoires.

	ERAN		Last Particle		
ε	Certifié (%)	temps (sec. ± std)	Validé (%)	temps (sec. ± std)	faux positifs (%)
0.01	100	1553.5 ± 2396	100	332 ± 22	0
0.05	100	10686 ± 10368	100	319 ± 26	0

Pour conclure, nous présentons dans le tableau 5 des résultats de certification de 3 modèles plus complexes (31k, 63k et 134k paramètres, resp.), utilisant notamment des neurones récurrents de type LSTM. Pour ces modèles nous n'avons pu appliquer que la méthode Last Particle proposée dans ce papier.

Table 5. Données ADSB dynamiques – Certification de réseaux de neurones récurrents avec Last Particle [$N = 2, p_c = 10^{-10}$,$t = 40$]. Temps de vérification pour 100 trajectoires.

	Modèle 1		Modèle 2		Modèle 3	
ε	Validé (%)	temps (sec.)	Validé (%)	temps (sec.)	Validé (%)	temps (sec.)
0.01	100	639.4	100	5883	98.0	1460
0.05	97	562	100	5352	92.0	1330
0.1	90	619	100	6372	85.0	1292
0.5	27	311	13	3069	14.0	712

5 Conclusion

L'article propose une simulation stochastique pour évaluer la robustesse de modèles. Il prend les points de vue du test d'hypothèse (faux positif/faux négatif) et de la certification (complétude/consistance). La procédure proposée est efficace, complète et s'accompagne de garanties théoriques. Elle est aussi générale, fonctionnant avec des classifieurs en 'boîte noire' qu'il s'agisse de réseaux de neurones ou des forêts aléatoires par exemple. La principale limitation est que la simulation de la dernière particule est séquentielle, ce qui n'est pas compatible avec le GPU. Cependant, notre implémentation permet de traiter plusieurs entrées en parallèle.

Nos futurs travaux concernent son accélération lorsque la procédure est appliquée à un réseau de neurones en particulier. En effet, la procédure utilise le réseau seulement en 'boîte noire' et n'exploite pas le gradient ∇f de la fonction réseau f pourtant facilement calculable grâce à la rétro-propagation.

References

1. Baluta, T., Chua, Z.L., Meel, K.S., Saxena, P.: Scalable quantitative verification for deep neural networks. In: Proc. of Int. Conf. on Software Engineering (2021)
2. Boopathy, A., Weng, T.W., Chen, P.Y., Liu, S., Daniel, L.: CNN-Cert: An efficient framework for certifying robustness of convolutional neural networks. In: AAAI (Jan 2019)
3. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on Security and Privacy (SP). pp. 39–57. IEEE Computer Society, Los Alamitos, CA, USA (may 2017). https://doi.org/10.1109/SP.2017.49, https://doi.ieeecomputersociety.org/10.1109/SP.2017.49
4. Chopin, P., Barbaresco, F., Jouaber, S.: Dispositif d'identification d'un type d'aéronef, procédé d'identification et programme d'ordinateur associés, Office Européen des Brevets, 20169176.3, 14 octobre 2020
5. Franceschi, J.Y., Fawzi, A., Fawzi, O.: Robustness of classifiers to uniform ℓ_p and gaussian noise. In: Storkey, A., Perez-Cruz, F. (eds.) Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 84, pp. 1280–1288. PMLR (09–11 Apr 2018), http://proceedings.mlr.press/v84/franceschi18a.html
6. Gilmer, J., Ford, N., Carlini, N., Cubuk, E.: Adversarial examples are a natural consequence of test error in noise. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 2280–2289. PMLR (09–15 Jun 2019), http://proceedings.mlr.press/v97/gilmer19a.html
7. Guyader, A., Hengartner, N., Matzner-Løber, E.: Simulation and estimation of extreme quantiles and extreme probabilities. Applied Mathematics & Optimization **64**, 171–196 (10 2011). https://doi.org/10.1007/s00245-011-9135-z
8. Katz, G., Barrett, C., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: An efficient SMT solver for verifying deep neural networks. In: Majumdar, R., Kunčak, V. (eds.) Computer Aided Verification. pp. 97–117. Springer International Publishing, Cham (2017), https://arxiv.org/abs/1312.6199
9. Kurakin, A., Goodfellow, I.J., Bengio, S.: Adversarial machine learning at scale. In: 5th International Conference on Learning Representations, ICLR 2017, Conference Track Proceedings. OpenReview.net (2017), https://openreview.net/forum?id=BJm4T4Kgx
10. Pinot, R., Meunier, L., Araujo, A., Kashima, H., Yger, F., Gouy-Pailler, C., Atif, J.: Theoretical evidence for adversarial robustness through randomization. In: Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada. pp. 11838–11848 (2019), https://proceedings.neurips.cc/paper/2019/hash/36ab62655fa81ce8735ce7cfdaf7c9e8-Abstract.html
11. Salman, H., Yang, G., Zhang, H., Hsieh, C.J., Zhang, P.: A convex relaxation barrier to tight robustness verification of neural networks. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alche Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 32. Curran Associates, Inc. (2019), https://proceedings.neurips.cc/paper/2019/file/246a3c5544feb054f3ea718f61adfa16-Paper.pdf
12. Singh, G., Gehr, T., Püschel, M., Vechev, M.: An abstract domain for certifying neural networks. Proc. ACM Program. Lang. **3**(POPL) (Jan 2019). https://doi.org/10.1145/3290354, https://doi.org/10.1145/3290354

13. Webb, S., Rainforth, T., Teh, Y.W., Kumar, M.P.: A statistical approach to assessing neural network robustness. In: International Conference on Learning Representations (2019)
14. Weng, L., Chen, P.Y., Nguyen, L., Squillante, M., Boopathy, A., Oseledets, I., Daniel, L.: PROVEN: Verifying robustness of neural networks with a probabilistic approach. In: Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 6727–6736. PMLR (09–15 Jun 2019), http://proceedings.mlr.press/v97/weng19a.html
15. Weng, L., Zhang, H., Chen, H., Song, Z., Hsieh, C.J., Daniel, L., Boning, D., Dhillon, I.: Towards fast computation of certified robustness for ReLU networks. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 5276–5285. PMLR (10–15 Jul 2018), http://proceedings.mlr.press/v80/weng18a.html
16. Zhang, H., Weng, T.W., Chen, P.Y., Hsieh, C.J., Daniel, L.: Efficient neural network robustness certification with general activation functions. In: Advances in Neural Information Processing Systems (NeurIPS) (dec 2018)

A Pseudo-codes des algorithmes utilisés

Dans l'algorithme 1 ci-dessous, $\mathsf{Comp\_m}(p_c, \alpha, N)$ est une approximation numérique de plus petit entier tel que $P_{X\sim\Gamma(m,N)}[X \leq -log(p_c)] = \alpha$.

Algorithm 1 Évaluation de robustesse avec l'algorithme de la dernière particule

Require: Nombre de particules N, niveau critique de probabilité p_c, niveau de confiance α
Ensure: Cert

Initialize: $p \leftarrow 1 - 1/N$, $k \leftarrow 1$, $\mathsf{Cert} \leftarrow False$, $\mathsf{Stop} \leftarrow False$
$m \leftarrow \mathsf{Comp\_m}(p_c, \alpha, N)$
$\{\mathbf{x}_i\}_{i=1}^N \leftarrow \mathsf{Gen}(-\infty, N)$
while $k \leq m$ & $\mathsf{Stop} = False$ **do**
    $i^\star \leftarrow \arg\min_{i\in 1:N} h(\mathbf{x}_i)$
    $L_k \leftarrow h(\mathbf{x}_{i^\star})$
    if $L_k > 0$ **then**
        $\mathsf{Stop} \leftarrow True$
        $P_{est} \leftarrow p^{k-1}$
    end if
    $\mathbf{x}_{i^\star} \leftarrow \mathsf{Gen}(L_k, 1)$
    $k \leftarrow k + 1$
end while
if $\mathsf{Stop} = False$ **then**
    $\mathsf{Cert} \leftarrow True$
    $P_{est} \leftarrow p_c$
end if
return Cert, P_{est}

Algorithm 2 Échantillonage conditionnelle d'une particule $\mathsf{Gen}(L, 1)$

Require: seuil limite L, ensemble fini $\mathcal{X}$ de particules dont le score est plus grand que L
Ensure: nouvelle particule $\mathbf{X}$

$\mathbf{X} \leftarrow \mathcal{U}(\mathcal{X})$ ▷ On tire uniformément une particule dans $\mathcal{X}$
for $k = 1 : t$ **do**
    $\mathbf{Z} \leftarrow K(\mathbf{X}, s)$ ▷ Transition π_0-réversible.
    if $h(\mathbf{Z}) > L$ **then** ▷ Rejet
        $\mathbf{X} \leftarrow \mathbf{Z}$
    end if
end for
return $\mathbf{X}$