Entrainement robuste et quantification dans Aidge

Augustin Lemesle [1], Julien Lehmann [2], Zakaria Chihani[1]

[1] CEA-List; prénom.nom@cea.fr
[2] OCamlPro; prénom.nom@ocamlpro.com

15 avril 2026

Résumé

Face à une demande croissante d'embarquer des composants IA dans des systèmes critiques, nous nous intéressons à la robustesse d'un modèle d'IA après sa quantification. Pour ce faire, nous entrainons et quantifions des modèles IA robustes et non-robustes avec la plateforme AIDGE et nous utilisons l'outil PyRAT pour vérifier leur robustesse. Nos premiers résultats suggèrent que la quantification des modèles IA conserve la robustesse mais rend la tâche de vérification légèrement plus complexe.

Mots-clés

Embarqué, Quantification, Robustesse, Vérification

Abstract

Facing an increasing demand for embedded AI in critical systems, we focus in this work on the robustness of AI models after their quantization. To do so, we train and quantize both robust and non-robust AI models using the AIDGE platform and we verify their robustness with the PyRAT tool. Our preliminary results suggest that quantizing AI models preserves the robustness while making the robustness verification process slightly harder.

Keywords

Embedded, Quantization, Robustness, Verification.

1 Introduction

Avec la perméation de l'IA dans des domaines de plus en plus critiques, des efforts conséquents ont été déployés pour réussir à caractériser et augmenter la robustesse des composants IA : de façon empirique, comme au travers de l'entraînement adverse, mais aussi grâce à des méthodes plus formelles, apportant des garanties mathématiques fortes de robustesse. Cependant, et bien que des avancées très encourageantes ont eu lieu dans ce domaine, l'objet d'étude reste une représentation haut-niveau du modèle IA, non conforme nécessairement avec son implémentation finale. Dans le domaine embarqué, un tel modèle, souvent de taille importante, ne peut pas être utilisé tel quel et doit passer par des étapes d'optimisation comme la quantification afin de répondre à des contraintes fortes d'implémentation, de puissance de calcul, ou de mémoire. Ce modèle quantifié garde-t-il les propriétés de robustesse préalablement établies ? Empiriquement, des éléments de réponses ont déjà été proposés [5] mais qu'en est-il d'une robustesse plus formelle ? Pour répondre à cela, nous proposons d'entrainer des modèles que nous pouvons certifier être robustes puis de les quantifier en utilisant la plateforme AIDGE [6]. Nous mesurons ensuite l'impact de la quantification sur la robustesse des modèles grâce à l'outil PyRAT [4].

2 Robustesse & Quantification

2.1 Robustesse face aux perturbations

Étant donné un problème de classification, un modèle N est dit ϵ-robuste pour une entrée x si aucune autre classification ne peut exister dans la boule centrée en x de rayon ϵ, i.e.,

$$\forall x',\ \|x - x'\|_\infty \leq \epsilon\,,\ \ \arg\max N(x) = \arg\max N(x').$$

Le paramètre ϵ représente alors une intensité maximale de perturbation autorisée sur les entrées d'un modèle. Cette perturbation peut être d'origine naturelle (*e.g.,* luminosité, flou) ou d'attaques adversariales.

2.2 Entraînement

Pour entraîner un modèle afin qu'il soit robuste face aux perturbations, nous utilisons la méthode d'entraînement par propagation de bornes d'intervalles (IBP) [2]. Cette méthode combine une fonction de coût classique avec une fonction de coût robuste obtenue en calculant l'espace des sorties atteignables du modèle à partir d'une perturbation ϵ sur les entrées grâce à l'arithmétique des intervalles.

Nous avons implémenté cette méthode dans la plateforme AIDGE [6] spécialisée dans l'embarquabilité de l'IA. Cette méthode permet, sans modifier l'architecture du modèle, d'obtenir des modèles plus robustes et plus facilement vérifiables formellement que l'entrainement classique.

2.3 Quantification

Afin de diminuer la consommation mémoire d'un modèle, nous le compressons en réduisant le nombre de bits utilisés pour définir les paramètres du modèle : c'est la quantification. Pour que le modèle quantifié reste fidèle au modèle original, il est nécessaire de transformer et d'optimiser la représentation des nouveaux paramètres quantifiés. C'est ce qui est effectué par la méthode de quantification après entraînement (PTQ) [3] dans la plateforme AIDGE.

3 Vérification

Afin de vérifier la robustesse d'un modèle, nous utilisons PyRAT [4], un logiciel à l'état de l'art de la vérification formelle de réseaux de neurones. PyRAT utilise des méthodes de sur-approximation ensemblistes afin de prouver des propriétés de sûreté et de robustesse. Pour une entrée et un ϵ donné, PyRAT permet de prouver qu'un modèle est ϵ-robuste (Robuste) ou qu'un contre-exemple existe (Non-Robuste). Si la précision de l'analyse n'est pas suffisante, PyRAT peut également ne pas réussir à conclure (Inconnu).

4 Expérimentation

4.1 Mise en place

Nous entraînons 2 réseaux de neurones simples sur MNIST avec la même architecture de 3 couches de 100 neurones et des fonctions d'activation ReLU. Le premier est entrainée de façon classique et le second utilise un entrainement robuste IBP avec $\epsilon = 4 / 256$. Les modèles sont entrainés pour avoir une précision similaire 96-97%. La quantification est calibrée sur 1000 images de l'ensemble d'entraînement et nous utilisons PyRAT pour évaluer la robustesse formelle de ces modèles sur 1000 images de l'ensemble de test face à deux ϵ différent, 4/256 et 8/256.

4.2 Résultats

Les résultats de la vérification sont présentés dans les tables 1 et 2. Tout d'abord, nous pouvons voir sur les modèles non quantifiés que nous arrivons à vérifier respectivement 92.6% et 84.3% de robustesse sur les modèles robustes conte 13% et 0%, démontrant l'intérêt de l'entainement IBP. PyRAT est également capable de trouver un plus grand nombre de contre-exemples sur les modèles classiques.
Après la quantification, cette tendance est conservée malgré une légère diminitution de robustesse des modèles. En même temps, la proportion de résultat Inconnu augmente également sur les modèles quantifiés, qui semblent alors plus compliqués à vérifier ou à falsifier. De manière générale, il semble néanmoins que les modèles quantifiés conservent la robustesse pour laquelle ils ont été entrainés.

	Prec.	Robustesse ($\epsilon = 4/256$)		
		Robuste	Non Robuste	Inconnu
Classique	**97.4 %**	13.0 %	23.7 %	63.3 %
Robuste	96.2 %	**92.6 %**	**6.8 %**	**0.6 %**
Classique 8 bits	**97.5 %**	0.0 %	**3.8 %**	96.2 %
Robuste 8 bits	96.0 %	**82.9 %**	4.4 %	**12.7 %**

TABLE 1 – Précision et Robustesse d'un réseau classique vs robuste et leur quantification pour $\epsilon = 4 / 256$

5 Discussion & Perspectives

Ces travaux préliminaires montrent qu'il est possible d'entrainer des modèles formellement certifiés robustes à des perturbations même pour des cibles embarquées avec des contraintes fortes comme l'utilisation d'entier 8 bits. L'intégration de l'entrainement IBP à la plateforme AIDGE et son utilisation en combinaison avec PyRAT offre un moyen accessible et efficace de réaliser cela.
L'extension de ces travaux pour considérer des cas d'usages plus réalistes que MNIST tels que l'ACAS-Xu, permettra de démontrer l'utilité de ces résultats de manière plus large. Cette démarche et la robustesse formellement prouvée obtenue pourront alors être mise en avant dans des démarches de certification d'un composant IA [1].
Ce travail a été financé par le projet DeepGreen ANR-23-DEGR-0001.

	Prec.	Robustesse ($\epsilon = 8/256$)		
		Robuste	Non Robuste	Inconnu
Classique	**97.4 %**	0.0 %	68.4 %	31.6 %
Robuste	96.2 %	**84.3 %**	**11.3 %**	**4.4 %**
Classique 8 bits	**97.5 %**	0.0 %	**3.6 %**	96.4 %
Robuste 8 bits	96.0 %	**38.7 %**	4.5 %	**56.8 %**

TABLE 2 – Précision et Robustesse d'un réseau classique vs robuste et leur quantification pour $\epsilon = 8 / 256$

Références

[1] Christophe Gabreau, Marie-Charlotte Teulières, Eric Jenn, Augustin Lemesle, Dumitru Potop-Butucaru, Floris Thiant, Lucas Fischer, and Mariem Turki. A study of an acas-xu exact implementation using ed-324/arp6983. In *12th European Congress Embedded Real Time Systems-ERTS 2024*, 2024.

[2] S. Gowal, K. Dvijotham, R. Stanforth, R. Bunel, C. Qin, J. Uesato, R. Arandjelovic, T. Mann, and P. Kohli. Scalable verified training for provably robust image classification. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4841–4850, 2019.

[3] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713, 2018.

[4] Augustin Lemesle, Julien Lehmann, Tristan Le Gall, and Zakaria Chihani. Verifying neural networks with pyrat. In *International Static Analysis Symposium*, pages 11–33. Springer, 2025.

[5] Qun Li, Yuan Meng, Chen Tang, Jiacheng Jiang, and Zhi Wang. Investigating the impact of quantization on adversarial robustness, 2024.

[6] Cyril Moineau, Maxence Naud, Iryna De Albuquerque Silva, Axel Farrugia, and Olivier Bichler. Aidge : A comprehensive and traceable library for deploying standalone embedded deep neural networks. In *ERTS 2026-13th European Congress of Embedded Real Time Systems*, 2026.