Tolérance aux Fautes des CNNs Quantifiés et Élagués

Wilfread GUILLEME[1] Angeliki KRITIKAKOU[1, 2] Youri HELEN[3] Cedric KILLIAN[4] Daniel CHILLET[1]

[1]Université de Rennes, Inria, IRISA, 263 Av. Général Leclerc, 35000 Rennes, France

[2]Institut Universitaire de France, 103 Boulevard Saint-Michel, 75005 Paris, France

[3]DGA MI, 136 La Roche Marguerite, 35170 Bruz, France

[4]Université Jean Monnet Saint-Etienne, CNRS, Laboratoire Hubert Curien UMR 5516, F-42023 Saint-Étienne, France

Résumé – Les réseaux de neurones, en particulier ceux utilisés dans des applications critiques telles que la conduite autonome, nécessitent une fiabilité optimale. Bien que ces algorithmes soient intrinsèquement résilients, des événements singuliers peuvent affecter à la fois le matériel et le logiciel. Ces événements sont dus à des interactions avec des rayonnements cosmiques et radiatifs, qui entraînent des erreurs au niveau des composants électroniques. Dans ce contexte, un réseau de neurones convolutifs (CNN) peut produire des calculs erronés, augmentant le risque de prédictions incorrectes. Afin de répondre à ce défi, nous avons développé une méthode d'injection de fautes spécifiquement conçue pour des CNNs dédiés au domaine des systèmes embarqués. Pour implémenter efficacement ces réseaux, leur empreinte matérielle doit être la plus faible possible et leur déploiement passe généralement par une phase de quantification et d'élagage des paramètres. Par ailleurs, la méthode proposée tient compte de la direction des bitflips simulés, permettant de concevoir des solutions de durcissement matériel adaptées à la fiabilité requise par l'application ciblée.

Abstract – Neural networks, particularly those used in critical applications like autonomous driving, require high reliability. Although inherently resilient, these algorithms remain vulnerable to singular events affecting hardware and software, often caused by cosmic or radiative rays that induce errors in electronic components. Consequently, Convolutional Neural Networks (CNNs) may produce erroneous computations, increasing the risk of incorrect predictions. To address this, we developed a fault injection method tailored for CNNs in embedded systems. Efficient deployment requires minimizing their hardware footprint, typically achieved through quantization and pruning. Additionally, our method accounts for the direction of simulated bitflips, enabling hardware hardening solutions adapted to the application's reliability needs.

1 Introduction

Les réseaux de neurones convolutifs (CNNs) sont aujourd'hui omniprésents dans divers domaines, allant de la reconnaissance d'images à la conduite autonome. Leur déploiement sur des systèmes embarqués pose cependant des défis en termes d'intégration, de consommation énergétique et de rapidité d'exécution. Afin de réduire le coût matériel des CNNs, plusieurs approches peuvent être envisagées.

L'une des solutions consiste à entraîner le modèle en prenant en compte la quantification en virgule fixe des paramètres, ce qui permet de préserver la précision des prédictions tout en optimisant l'utilisation des ressources matérielles. Généralement, les CNNs sont entraînés en utilisant un format en virgule flottante sur 32 bits pour stocker les paramètres. En adoptant un format de données en virgule fixe, avec une précision réduite et un nombre de bits plus faible, il est possible de diminuer significativement les ressources nécessaires, notamment pour le stockage des paramètres. De plus, ce format de données réduit la complexité des opérateurs mathématiques, rendant leur implémentation matérielle plus facile et moins coûteuse que celle des opérateurs en virgule flottante. Pour mettre en œuvre cette technique, la librairie Brevitas [1], issue d'un projet de recherche open source, est particulièrement efficace.

Une autre optimisation repose sur l'élagage des poids, qui consiste à supprimer les poids les moins significatifs du modèle afin de réduire le nombre de calculs nécessaires lors de l'inférence, notamment en évitant les multiplications inutiles avec des valeurs proches de zéro. Cette approche permet non seulement d'accélérer les calculs, mais aussi de réduire la consommation d'énergie et l'espace mémoire requis pour stocker les paramètres. Une méthode courante pour encourager la sparsité est l'utilisation d'une régularisation L1 lors de l'entraînement sur l'ensemble du modèle. Cette technique peut être mise en œuvre à l'aide de la librairie Prune [2], qui permet de réduire la taille du modèle en supprimant les paramètres les moins pertinents. Cette réduction allège ainsi les calculs, rendant le modèle plus rapide et moins gourmand en ressources.

Cet article examine la fiabilité d'un CNN, allégé par quantification en virgule fixe et par élagage appliqué aux poids faibles, face aux fautes causées par des événements liés au rayonnement cosmique et radiatif. Ces événements peuvent générer des erreurs lorsqu'une particule interagit avec un composant électronique. Bien que ces algorithmes présentent une résilience intrinsèque aux fautes, il est pertinent d'évaluer leur sensibilité. Pour ce faire, un framework interne, SFI4NN, dédié à l'injection de fautes, est utilisé. Ce framework permet de simuler des bitflips en tenant compte de leur direction (0 vers 1 ou 1 vers 0), de leur position dans le CNN et de leur impact sur la prédiction du modèle. Des observations montrent que les CNNs réagissent différemment selon la direction des fautes. Ce constat ouvre la possibilité de concevoir une solution de protection matérielle plus légère que la triplication, offrant une alternative plus efficace en termes de coût matériel. Une expérience est réalisée pour évaluer la sensibilité des couches et des bits d'un CNN AlexNet [3] quantifié en 8 bits. Enfin, une étude est menée pour évaluer l'effet de l'élagage sur la fiabilité de cette architecture de CNN.

2 État de l'art

La fiabilité des CNNs en présence de fautes est un sujet de recherche actif. Deux approches principales émergent dans ce domaine : d'une part, l'évaluation de la résilience des modèles de CNN face aux fautes, et d'autre part, le développement de solutions matérielles visant à améliorer leur fiabilité.

Concernant la résilience des CNN, des travaux existants évaluent la tolérance aux fautes affectant les paramètres du modèle [4]. Avec la taille croissante des CNN, une injection de fautes exhaustive devient impraticable. Pour surmonter cette limitation, les auteurs de [5] ont proposé une méthode d'injection de fautes par approche statistique permettant d'évaluer efficacement la robustesse des CNNs tout en réduisant considérablement le temps de simulation. Cette approche limite le nombre de fautes testées par couche et par bit, réduisant ainsi la charge computationnelle par rapport aux méthodes exhaustives. Toutefois, cette étude se concentre uniquement sur les représentations en virgule flottante. Pour mieux comprendre les vulnérabilités liées aux différentes représentations de données, l'analyse menée dans [6] examine les représentations en virgule fixe et en virgule flottante. En virgule fixe, les bits de poids fort sont les plus critiques, tandis qu'en virgule flottante, ce sont les bits de l'exposant qui sont les plus sensibles. Bien que certaines études aient révélé des asymétries liées à la direction des bitflips [7], leur impact sur les classifications des CNNs reste à approfondir.

Les approches classiques de protection matérielle contre les fautes reposent généralement sur la redondance matérielle ou informationnelle. Dans [6], le renforcement sélectif par triplication des bits de poids fort des paramètres est proposé pour améliorer la tolérance aux fautes. L'utilisation de codes de correction d'erreurs pour protéger ces mêmes bits est explorée dans [8]. D'autres stratégies existent, comme MOZART+[9], qui est une solution spécifiquement conçue pour les CNNs. Cette solution, intégrée aux architectures de flux de données stationnaires en sortie, utilise un multiplexage permettant de désactiver l'élément de traitement défectueux tout en maintenant le fonctionnement global du système.

3 Méthodologie

3.1 Cadre d'injection de faute

Comme indiqué précédemment, une approche exhaustive pour étudier la résilience des CNNs n'est généralement pas possible compte tenu de leur taille, une approche statistique sera donc privilégiée [10]. En effet, l'injection de fautes statistiques (SFI) permet d'estimer l'impact des fautes sur un CNN, en réduisant significativement le nombre de tests nécessaires sans affecter la précision de l'étude. L'approche SFI, comme celle présentée dans [5], cible les couches de petite taille et les bits de poids fort, car ils influencent davantage le comportement du réseau. Cette méthode, décrite par l'équation (1), évalue la robustesse d'un CNN face aux fautes. Une marge d'erreur et un niveau de confiance définissent la précision souhaitée. Plus celle-ci est élevée, plus le nombre de fautes injectées augmente, ce qui prolonge la durée de simulation.

$$n(i,l) = \frac{N(i,l)}{1 + e^2 \cdot \frac{N(i,l)-1}{t^2 \cdot p(i) * (1-p(i))}} \tag{1}$$

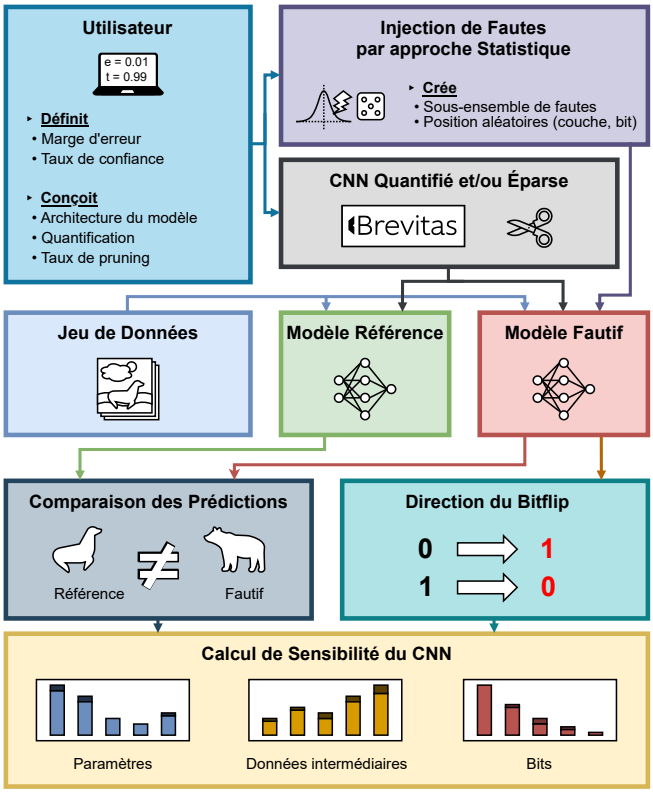


FIGURE 1 : Cadre d'injection de fautes SFI4NN.

$N(i,l)$ représente le nombre total de fautes pouvant être injectées dans le CNN, en fonction du bit i et de la couche l. Les paramètres e et t correspondent respectivement à la marge d'erreur et au niveau de confiance. Enfin, p indique la probabilité qu'une erreur survienne à chaque position de bit. Le nombre de fautes injectées augmente pour les bits de poids fort, car ils ont un impact plus important sur la résilience du CNN et sont donc plus susceptibles de provoquer des erreurs de prédiction. Ces probabilités, qui déterminent les taux d'injection de fautes selon les couches et les bits, sont calculées selon l'équation (2).

$$\forall i \in I \quad p(i) = p_{\text{min}} + \frac{(D(i) - D_{\text{min}})(p_{\text{max}} - p_{\text{min}})}{(D_{\text{max}} - D_{\text{min}})} \tag{2}$$

$D(i)$ représente la distance d'un bitflip sur une valeur en fonction du bit sélectionné. Pour une faute injectée sur le bit de poids faible d'une donnée encodée en entier signé, D_{min} est toujours égal à 1. Tandis que D_{max} dépend de la longueur des données I. Par exemple, pour une donnée en entier signé sur 8 bits, un bitflip sur le bit de poids fort donne une distance de 128. Enfin, p_{max} est fixé à 0.5, tandis que p_{min} dépend de la taille des données I via la relation $p_{\text{min}} = \frac{1}{2^I}$.

Le framework SFI4NN a été développé pour adapter cette approche d'injection de fautes, comme illustré à la figure 1. L'utilisateur définit la qualité des résultats souhaités en fixant la marge d'erreur et le niveau de confiance. L'entraînement utilise les bibliothèques Brevitas et Prune pour générer un CNN optimisé. ONNXRuntime exécute simultanément deux CNNs : un modèle de référence et un modèle fautif, sur lequel des fautes sont simulées. Une divergence entre leurs classifications indique que la faute simulée a un impact critique.

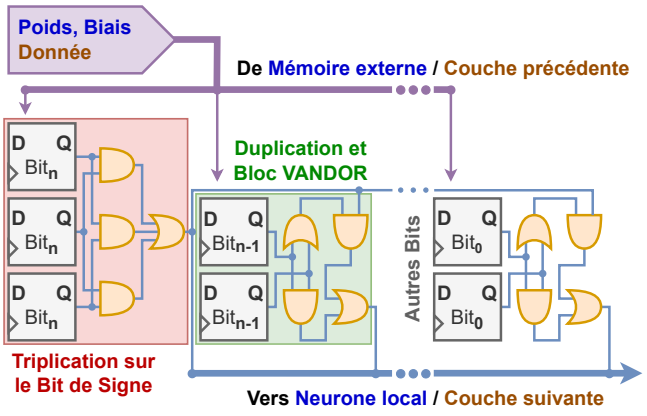


FIGURE 2 : Solution de durcissement matériel VANDOR.

3.2 Solution de protection matérielle

L'implémentation matérielle pour protéger le CNN, issue de l'article [11], est illustrée à la figure 2. Cette méthode de protection, nommée VANDOR, peut s'appliquer à tous les paramètres et données intermédiaires d'un CNN. Elle repose sur la triplication du bit de signe qui influence directement le mécanisme de protection des autres bits, qui ne sont que dupliqués. Une faute est détectée lorsque l'état logique du bit d'origine stocké dans une bascule D diffère de celui du bit dupliqué. Le processus de décision pour déterminer la valeur utilisée dans les calculs du CNN est géré par un bloc de vote composé de portes logiques. Lorsque le bit de signe est à '0' (valeur positive), le bloc de vote privilégie l'état logique '0' du bit fautif, fonctionnant comme une porte AND. À l'inverse, lorsque le bit signe est à '1' (valeur négative), le bloc de vote favorise l'état logique '1', agissant alors comme une porte OR. Dans les deux cas, la décision rapproche la valeur erronée vers zéro grâce à la représentation en complément à deux. Cette stratégie de protection repose sur l'observation que les fautes rapprochant la valeur erronée vers zéro ont un impact moindre sur les prédictions d'un CNN.

4 Expérience et résultats

4.1 Modèle et jeu de données utilisé

L'architecture AlexNet étudiée, composée de cinq couches convolutionnelles (CONV) suivies de trois couches entièrement connectées (FC), est détaillée dans le Tableau 1. Chaque couche CONV applique des filtres convolutifs générant, par exemple, 64 canaux dans la première couche. Des opérations de MaxPooling réduisent la dimensionnalité spatiale des données intermédiaires. La fonction d'activation ReLU est systématiquement utilisée après chaque couche pour introduire de la non-linéarité. Les deux premières couches FC comptent 4 096 neurones chacune, tandis que la dernière couche FC contient 10 neurones correspondant aux classes du jeu de données CIFAR-10. Ce dernier est composé de 60 000 images couleur de 32×32 pixels, il se divise en 50 000 images pour l'apprentissage et 10 000 pour le test. La classification finale repose sur le neurone présentant la valeur de sortie la plus élevée. L'architecture mobilise environ 28,5 millions de paramètres et traite 289 994 données intermédiaires au total.

TABLE 1 : Architecture détaillée du CNN AlexNet étudié.

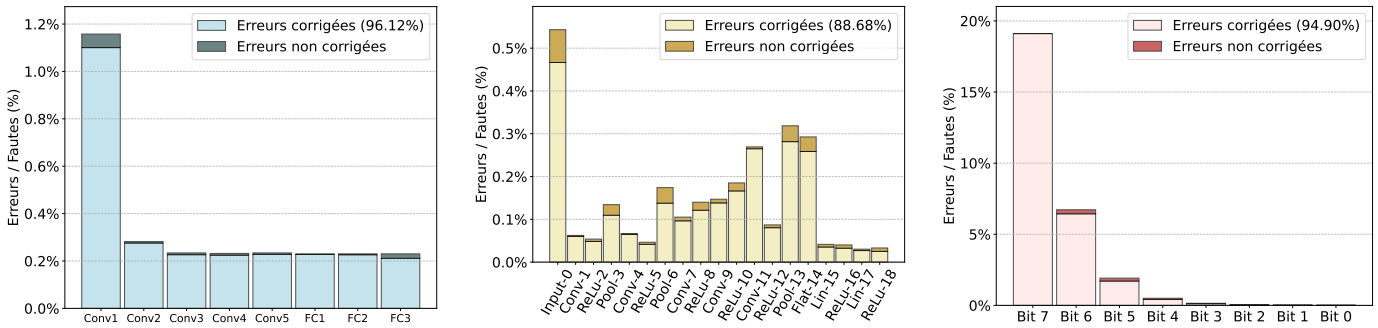
Type de couche		Nombre de données	Format	Nombre de paramètres
INPUT	Input-0	3 072	[3, 32, 32]	0
CONV-1	Conv-1	57 600	[64, 30, 30]	1 728
	ReLu-2	57 600	[64, 30, 30]	0
	Pool-3	14 400	[64, 15, 15]	0
CONV-2	Conv-4	32 448	[192, 13, 13]	110 592
	ReLu-5	32 448	[192, 13, 13]	0
	Pool-6	6 912	[192, 6, 6]	0
CONV-3	Conv-7	13 824	[384, 6, 6]	663 552
	ReLu-8	13 824	[384, 6, 6]	0
CONV-4	Conv-9	9 216	[256, 6, 6]	884 736
	ReLu-10	9 216	[256, 6, 6]	0
CONV-5	Conv-11	9 216	[256, 6, 6]	589 824
	ReLu-12	9 216	[256, 6, 6]	0
	Pool-13	2 304	[256, 3, 3]	0
FC-1	Flat-14	2 304	[2 304]	0
	Lin-15	4 096	[4 096]	9 437 184
	ReLu-16	4 096	[4 096]	0
FC-2	Lin-17	4 096	[4 096]	16 777 216
	ReLu-18	4 096	[4 096]	0
FC-3	Lin-19	10	[10]	40 960
Total		**289 994**		**28 505 792**

4.2 Sensibilité d'un modèle quantifié

Le modèle AlexNet a été quantifié sur 8 bits, atteignant une précision de plus de 84 % sur le jeu de données CIFAR-10. Une campagne d'injection de fautes a été menée sur les 10 000 images de test en utilisant l'approche SFI avec une marge d'erreur de 10 % et un niveau de confiance de 90 %. Les résultats sont illustrés à la Figure 3. La partie 3a montre la sensibilité des couches contenant les paramètres du modèle. La première couche de convolution est la plus sensible, avec plus de 1 % de risque d'erreur de classification. VANDOR offre une protection de 96 %, confirmant son efficacité. Pour les données intermédiaires, l'étude SFI révèle une variation notable de sensibilité entre les couches, voir Figure 3b, avec un taux de protection de 88 % en moyenne avec la solution VANDOR. Enfin, la Figure 3c montre la sensibilité des bits en tenant compte des poids et des données intermédiaires. Les bits de poids fort sont les plus vulnérables. En moyenne, VANDOR protège 91 % des fautes dans le CNN étudié.

4.3 Sensibilité en fonction du taux d'élagage

Cette étude analyse l'impact de l'élagage des poids sur la tolérance aux fautes d'un modèle CNN. Dix modèles AlexNet identiques ont été entraînés sur 100 epochs, avec des taux de suppression des connexions variant de 0 % à 60 %. L'élagage L1 uniforme a été appliqué aux couches CONV et FC aux epochs 20, 40, 60 et 80. Le Tableau 2 présente le nombre de paramètres forcés à zéro à l'issue de l'entraînement selon ces taux et le nombre de fautes injectées par approche statistique. Les résultats, illustrés à la Figure 4, montrent qu'une réduction des poids diminue la performance de classification, de 84 % à 74 % pour les modèles les plus élagués de cette étude. Concernant la tolérance aux fautes, la sensibilité du CNN augmente avec la perte de connexions. L'efficacité de la solution matérielle VANDOR décroît progressivement, atteignant 96 % de fautes protégées sur un modèle quantifié non élagué, puis 74 % avec une forte réduction des paramètres.



(a) Paramètres (b) Données intermédiaires (c) Position des bits

FIGURE 3 : Sensibilité du CNN et protection de VANDOR selon les paramètres, les données intermédiaires et la position des bits.

TABLE 2 : Caractéristiques de l'étude du taux d'élagage

Taux d'élagage (%)	0	5	10	20	30	40	45	50	55	60
Poids à zéro (%)	6.66	32.17	35.88	64.46	75.99	87.04	90.85	93.75	95.90	97.44
Classification (%)	84.05	84.68	84.47	84.20	84.05	82.56	82.13	80.67	76.92	74.33
Fautes injectées SFI (millions)	14.69	14.67	14.65	14.62	14.56	14.42	14.33	14.17	14.02	13.80

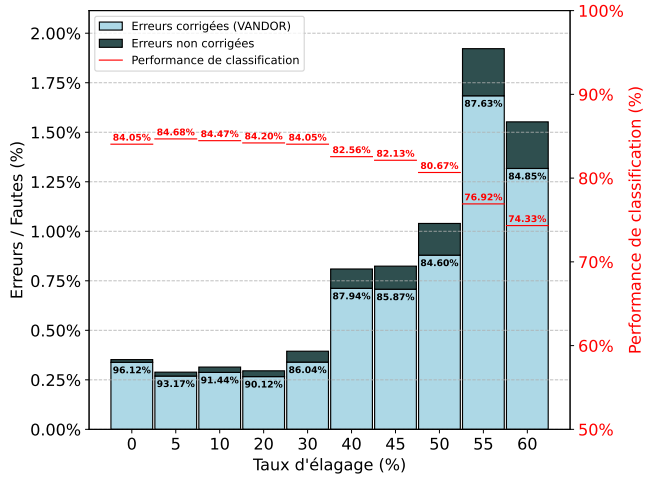


FIGURE 4 : Sensibilité des CNNs selon le taux d'élagage.

5 Conclusion

Cet article présente une méthode d'injection de fautes basée sur une approche statistique, appliquée aux paramètres, aux données intermédiaires et à la position des bits d'un CNN quantifié en virgule fixe et élagué. En se basant sur l'observation que les fautes rapprochant la valeur erronée vers zéro provoquent moins d'erreurs de prédiction, la solution matérielle VANDOR a été présentée. Par la suite, deux études ont été réalisées sur l'architecture CNN AlexNet. La première analyse concerne l'évaluation de sensibilité d'un modèle quantifié. La seconde étude concerne la sensibilité des CNNs en fonction du taux d'élagage. Les résultats ont montré une tendance nette, plus l'élagage des connexions est important, plus la sensibilité du modèle aux fautes augmente. Ainsi, il devient essentiel de trouver un équilibre entre la réduction des ressources et la fiabilité du modèle. VANDOR constitue une réponse efficace pour protéger ces algorithmes, bien que son efficacité diminue à mesure que l'élagage s'intensifie.

Références

[1] Alessandro P. Xilinx/brevitas, 2023.

[2] M. Paganini and J. Forde. Streamlining tensor and network pruning in pytorch. *arXiv preprint arXiv :2004.13770*, 2020.

[3] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. 2012.

[4] A. Bosio, P. Bernardi, A. Ruospo, and E. Sanchez. A reliability analysis of a deep neural network. 2019.

[5] A. Ruospo, G. Gavarini, C. De Sio, J. Guerrero, L. Sterpone, S. Reorda, E. Sanchez, R. Mariani, J. Aribido, and J. Athavale. Assessing convolutional neural networks reliability through statistical fault injections. 2023.

[6] G. Li, S. Hari, M. Sullivan, T. Tsai, K. Pattabiraman, J. Emer, and S. Keckler. Understanding error propagation in deep learning neural network (dnn) accelerators and applications. 2017.

[7] Y. Zhang, H. Itsuji, T. Uezono, T. Toba, and M. Hashimoto. Estimating vulnerability of all model parameters in dnn with a small number of fault injections. 2022.

[8] M. Traiola, A. Kritikakou, and O. Sentieys. hardnning : a machine-learning-based framework for fault tolerance assessment and protection of dnns. 2023.

[9] S. Burel, A. Evans, and L. Anghel. Mozart+ : Masking outputs with zeros for improved architectural robustness and testing of dnn accelerators. 2022.

[10] R. Leveugle, A. Calvez, P. Maistri, and P. Vanhauwaert. Statistical fault injection : Quantified error and confidence. 2009.

[11] W. Guillemé, A. Kritikakou, Y. Helen, C. Killian, and D. Chillet. Vandor : Mitigating seus into quantized neural networks. 2024.