Robustesse des réseaux de neurones profonds aux défaillances mémoire

Ghouthi BOUKLI HACENE[1,2], François LEDUC-PRIMEAU[3], Amal BEN SOUSSIA[2] Vincent GRIPON[1,2] François GAGNON[4]

[1]Université de Montréal, MILA

[2]IMT Atlantique, Lab-STICC

[3]Département de génie électrique, École Polytechnique de Montréal

[4]Département de génie électrique, École de technologie supérieure de Montréal

Résumé – Les Réseaux de Neurones Profonds (RNPs) reposent sur un grand nombre de paramètres et de calculs, de sorte que leur implémentation matérielle sur des systèmes contraints en ressources représente un défi. Dans cet article, nous nous intéressons à la possibilité de réduire la tension d'alimentation des mémoires utilisées dans un tel système pour diminuer la consommation énergétique. Nous analysons la robustesse de certains RNPs aux conséquences de ces réductions, et proposons une solution pour maintenir un haut niveau de précision. Nos expériences montrent clairement l'intérêt d'un système opérant dans un régime défectueux afin de réduire la consommation d'énergie, sans entraîner des pertes en précision.

Abstract – Because deep neural networks (DNNs) rely on a large number of parameters and computations, their implementation in energy-constrained systems is challenging. In this paper, we investigate the solution of reducing the supply voltage of the memories used in the system, which results in bit-cell faults. We explore the robustness of state-of-the-art DNN architectures towards such defects and propose a regularizer meant to mitigate their effects on accuracy. Our experiments clearly demonstrate the interest of operating the system in a faulty regime to save energy without reducing accuracy.

1 Introduction

Durant la dernière décennie, les Réseaux de Neurones Profonds (RNPs) sont devenus l'état de l'art dans plusieurs domaines tels que la vision par ordinateur [13] ou le langage naturel [10]. Cependant, pour atteindre une bonne précision, les RNPs utilisent un grand nombre de paramètres et une complexité de calcul considérable, et donc entraînent une consommation d'énergie trop élevée pour être directement utilisables sur des systèmes embarqués. Puisqu'un accès à une mémoire externe est très coûteux en énergie, il est nécessaire de stocker toutes les données sur une mémoire interne. La conséquence est que l'énergie consommée par la mémoire interne représente 30% à 60% de l'énergie totale [11]. Afin de diminuer cette consommation, il est possible de réduire la tension d'alimentation de la mémoire, mais ceci entraîne l'apparition d'erreurs sur les bits stockés [3]. Si de telles erreurs peuvent être dramatiques dans certains cas, la précision des RNPs peut être maintenue à condition d'utiliser des mécanismes adéquats.

Les RNPs disposent d'une certaine tolérance aux erreurs introduites [19, 9], ce qui a entraîné de nombreuses contributions cherchant à démontrer l'intérêt de réduire la tension d'alimentation. Par exemple, quand des erreurs au niveau des bits sont détectées, un masque binaire peut être utilisé afin de s'assurer que les erreurs réduisent la valeurs des poids erronés, ce qui a pour effet de limiter l'impact des erreurs sur la précision [15, 20]. Dans [11], les auteurs proposent de modifier le processus d'entraînement en prenant en compte les erreurs pouvant affecter les bits stockés dans la mémoire. Dans [14, 21] les auteurs considèrent le cas où un RNP est entraîné afin de réduire les effets d'erreurs à des emplacements connus.

D'autres types de méthodes visant à compresser les RNPs et à réduire leur nombre de paramètres et leur complexité ont été introduites. Elles sont de trois types principaux. Le premier utilise la quantification sur un nombre de bits réduit afin de coder les valeurs des poids et des activations [2, 17]. Le deuxième utilise des techniques d'élagage afin d'enlever certaines connexions jugées peu utiles à la précision du RNP [4]. Finalement, le troisième utilise la factorisation des poids afin d'utiliser une seule valeur partagée par plusieurs poids [12, 7].

Dans cette contribution, nous introduisons une régularisation qui, lorsqu'elle est mise en place pendant la phase d'entraînement, permet de considérablement accroître la robustesse des RNPs aux pertes de précision susceptibles d'apparaître lorsque la tension d'alimentation est réduite.

2 Description de la méthode

Dans notre étude, nous tenons compte uniquement de l'énergie consommée par les accès mémoire, et nous considérons que

l'énergie totale consommée pour calculer une décision du RNP est proportionelle a celle consommée par ces accès mémoire. Ainsi, nous introduisons une énergie de base E_0 proportionnelle à la quantité de paramètres nécessaires au calcul d'une décision du RNP, soit la somme entre le nombre de poids et le nombre de dimensions des vecteurs d'activation dans le réseau.

Afin de réduire la consommation d'énergie, nous proposons de réduire la tension d'alimentation, ce qui entraînera une défaillance de la mémoire introduisant des erreurs sur les bits stockés. Dans le but d'étudier le comportement de l'implémentation des RNPs subissant des défaillances mémoires, il nous faut trouver une relation permettant d'exprimer la probabilité p qu'un bit soit erroné en fonction de l'énergie consommée par les accès mémoire. Nous notons par $0 < \eta < 1$ la consommation d'énergie normalisée par rapport à la consommation d'énergie d'une mémoire fiable, de sorte que la consommation d'énergie totale devient ηE_0.

En utilisant les données publiées dans [3, Fig.7], nous considérons que p et η sont reliés par une fonction exponentielle qui s'exprime comme suit :

$$p(\eta) = e^{-a\eta} \, . \tag{1}$$

Pour fixer la valeur de a, nous nous appuyons sur les données d'énergie présentées dans [1, Fig.1], et à l'étude de la fiabilité de la mémoire pour un CMOS 65nm avec un $V_{\mathrm{DD}} \in \{0.5, 1.1\}$ introduite dans [3, Fig.7]. En minimisant la somme du carré des erreurs relatives, nous aboutissons à $a = 12.8$. Dans notre étude, nous considérons le cas où une erreur introduite au niveau d'un bit peut être détectée, et nous proposons d'utiliser le Masquage de Bit (MB), une technique de compensation introduite dans [15]. Le modèle MB utilisé peut être défini comme suit : si une erreur est détectée au niveau du bit de signe, la valeur correspondante est remplacée par zéro. Si par ailleurs, une erreur est détectée au niveau d'un autre bit, sa valeur est remplacée par la valeur du bit de signe, de façon à faire converger les valeurs dont certains bits sont erronés vers zéro.

Dans cette contribution, nous cherchons à utiliser un modèle d'erreur correspondant à la technique MB durant la phase d'apprentissage afin de rendre le RNP plus robuste à ces défauts et ainsi éviter une baisse significative de la précision. Pour ce faire, nous cherchons à approcher le modèle MB par un modèle plus simple étant moins gourmand en calculs. Puisque le modèle MB fait en sorte que les valeurs affectées par une erreur tendent toujours vers zéro, nous proposons de l'approcher par un modèle d'*effacement*, dans lequel chaque valeur a une probabilité p_e d'être transformée en zéro. Durant la phase d'apprentissage, le modèle d'effacement se comporte de la même manière qu'un *dropout* [18], a l'exception qu'il est utilisé pour augmenter la robustesse du RNP et non pas pour limiter les effets du sur-apprentissage.

3 Résultats

Afin d'évaluer la méthode proposée, nous utilisons la base de données CIFAR10 qui comporte des petites images en couleur de taille 32×32 pixels. Premièrement, nous comparons 4 architectures de RNPs, à savoir PreActResNet18 [6], MobileNetV2 [16], SENet18 [8] et ResNet18 [5], qui sont toutes des architectures modernes ayant une bonne précision sur CIFAR10. La Table 1 montre le nombre de paramètres (poids) et activations utilisés pour que le RNP calcule une décision, ainsi que la précision de chaque architecture sur la base de donnée.

TABLE 1 – Nombre des accès mémoire et précision atteinte pour chaque architecture.

Architecture	Paramètres	Activations	Précision
PreActResNet18 [6]	11.2×10^6	0.55×10^6	94.87%
MobileNetV2 [16]	2.30×10^6	1.53×10^6	93.80%
SENet18 [8]	11.3×10^6	0.86×10^6	94.77%
ResNet18 [5]	11.2×10^6	0.56×10^6	94.86%

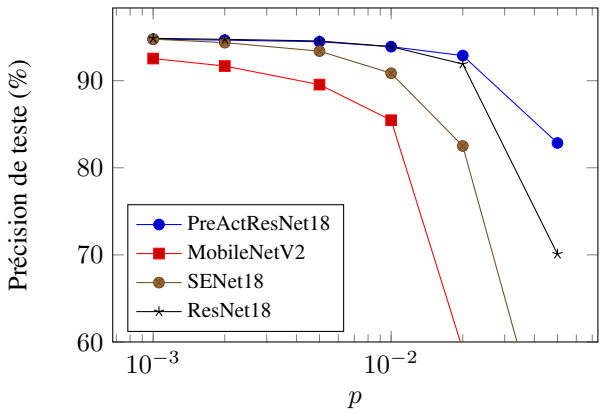

FIGURE 1 – Évolution de la précision de chaque architecture en fonction de la probabilité de défaillance dans le modèle MB.

Dans la figure 1, nous comparons la robustesse des architectures citées précédemment dans le scénario où les paramètres et les activations sont affectées par le modèle MB. Nous observons que certaines architectures sont plus robustes que d'autres, et que cette différence de robustesse ne dépend pas uniquement du nombre de paramètres.

Dans la fgure 2, nous traçons des courbes de précision en fonction de l'énergie ηE_0, où E_0 représente la somme du nombre de paramètres et des activations introduits dans la Table 1. La probabilité de faute p est obtenue à partir de l'énergie normalisée η en utilisant l'équation (1). Nous constatons que PreActResNet18 présente un bon compromis entre la précision, la consommation d'énergie et la robustesse aux défauts MB. Par conséquent, nous utilisons uniquement cette architecture dans les expériences restantes.

Dans la prochaine évaluation, nous nous intéressons à identifier la robustesse relative de différentes parties du RNP en utilisant le modèle MB. Pour cela, nous introduisons le modèle MB uniquement sur une portion du réseau. Sachant que PreActResNet18 se compose de 4 blocs séquentiels (comportant des couches convolutives et des couches de normalisation des lots),

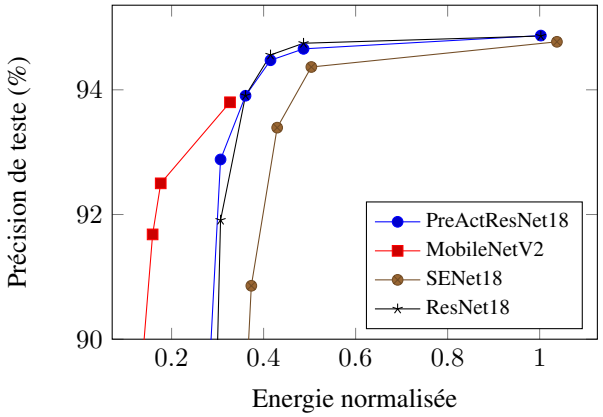

FIGURE 2 – Évolution de la précision de chaque architecture en fonction de la quantité d'énergie consommée pour une décision, obtenue en faisant varier la proportion de fautes dans le modèle MB.

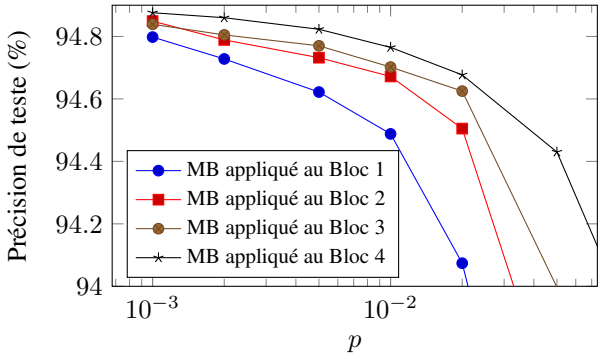

FIGURE 3 – Évolution de la précision pour l'architecture PreActResNet18 en fonction de la proportion de défaillances dans le modèle MB, et des parties (blocs) du réseau affectées.

nous appliquons le modèle MB à un bloc à la fois. Nous voyons figure 3 que l'effet du MB n'est pas le même pour chaque bloc. En effet, nous remarquons que la robustesse augmente avec la profondeur du réseau, de sorte qu'exploiter la différence de robustesse entre les couches peut s'avérer un moyen pertinent afin de réduire davantage la consommation d'énergie tout en gardant une précision acceptable. Nous appellerons ce procédé "Faute Diff" dans les évaluations restantes. Nous remarquons également à la figure 3 que le point correspondant à la précision maximale de 94.8% est obtenu pour $p_{\mathrm{B4}} = 5p_{\mathrm{B3}} = 5p_{\mathrm{B2}} = 10p_{\mathrm{B1}}$, où $p_{\mathrm{B}i}$ représente la probabilité d'erreur assignée au bloc i. Par conséquent, nous utiliserons cette configuration lors de l'introduction de "Faute Diff" dans la figure 5.

Afin d'introduire la régularisation durant la phase d'apprentissage, nous devons d'abord trouver les paramètres du modèle d'effacement approchant au mieux le modèle MB. Pour ce faire, nous proposons d'appliquer les deux modèles sur le réseau et de comparer leur effets. La figure 4 montre les résultats d'une telle évaluation. Puisque PreActResNet18 contient $20\times$ plus de paramètres que d'activations, nous considérons pour notre analyse que le cas où les deux modèles sont appliqués uniquement sur les paramètres. Nous observons que généralement la même précision est obtenue en appliquant les deux modèles pour $p_e = 2p$, et donc nous adopterons cette relation afin d'approcher le modèle MB en utilisant le modèle d'effacement.

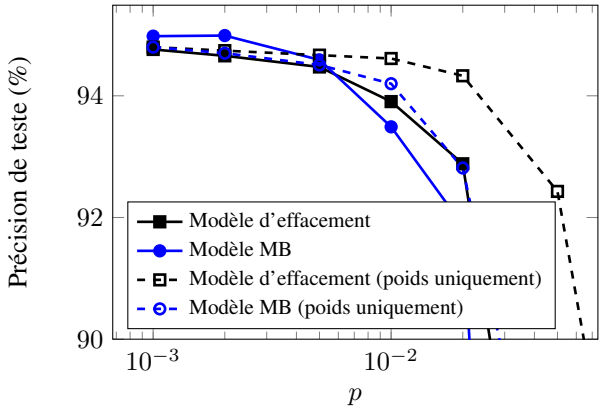

FIGURE 4 – Évolution de la précision en fonction de la proportion de défaillances mémoire pour plusieurs modèles.

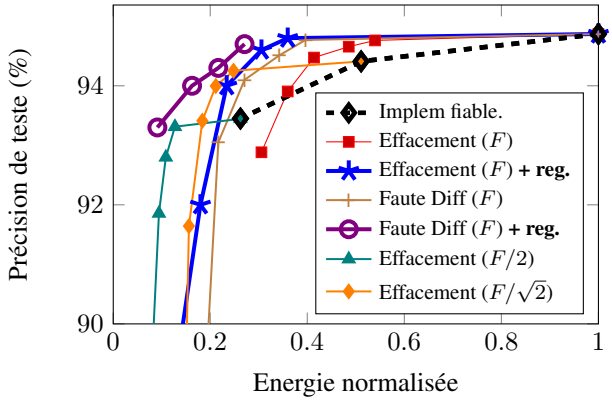

FIGURE 5 – Évolution de la précision en fonction de la consommation d'énergie pour plusieurs des méthodes introduites dans cet article.

Pour la dernière évaluation, nous introduisons la régularisation consistant à appliquer le modèle d'effacement durant la phase d'apprentissage. Nous considérons également l'effet du nombre de paramètres dans l'architecture sur la précision. Sachant que le nombre de paramètres pour chaque couche convolutive dépend linéairement du nombre de canaux d'entrée et de sortie, une façon simple de réduire le nombre de paramètres consiste à réduire le nombre de canaux d'entrée et/ou de sortie. Nous introduisons donc deux variantes de PreActResNet où le nombre de paramètres est respectivement $F/2$ et $F/\sqrt{2}$, où F représente le nombre original de paramètres. Ces deux variantes serviront de base de comparaison.

La figure 4 montre que l'utilisation de la régularisation permet d'atteindre une meilleure précision en consommant moins d'énergie. De plus, la combinaison de cette régularisation avec la "Faute Diff" entraîne un gain supplémentaire en précision à une consommation d'énergie égale. Nous remarquons aussi qu'il est préférable d'entraîner un RNP a être robuste aux erreurs pour réduire l'énergie plutôt que juste réduire le nombre de paramètres.

4 Conclusion

Dans cette contribution, nous avons exploré la possibilité d'exploiter la tolérance aux erreurs des RNP dans le but de réduire la consommation d'énergie. Nous avons montré qu'il est possible d'atteindre un meilleur compromis entre la précision et l'énergie en combinant une réduction de la tension d'alimentation avec une régularisation pendant l'apprentissage ayant pour objectif d'accroître la robustesse des architectures aux défauts induits par cette réduction. Une étude pour évaluer une éventuelle combinaison entre cette méthode et des techniques d'élagage, de quantification ou encore de factorisation peut faire l'objet d'un travail futur.

Références

[1] G. Chen, D. Sylvester, D. Blaauw, and T. Mudge. Yield-driven near-threshold SRAM design. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 18(11) :1590–1598, Nov 2010.

[2] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect : Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015.

[3] R.G. Dreslinski, M. Wieckowski, D Blaauw, D Sylvester, and T. Mudge. Near-threshold computing : Reclaiming Moore's law through energy efficient integrated circuits. *Proc. of the IEEE*, 98(2) :253–266, Feb. 2010.

[4] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[7] Lu Hou, Quanming Yao, and James T Kwok. Loss-aware binarization of deep networks. *arXiv preprint arXiv :1611.01600*, 2016.

[8] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. *arXiv preprint arXiv :1709.01507*, 7, 2017.

[9] X. Jiao, M. Luo, J. Lin, and R. K. Gupta. An assessment of vulnerability of hardware neural networks to dynamic voltage and temperature variations. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 945–950, Nov 2017.

[10] Diksha Khurana, Aditya Koli, Kiran Khatter, and Sukhdev Singh. Natural language processing : State of the art, current trends and challenges. 08 2017.

[11] S. Kim, P. Howe, T. Moreau, A. Alaghi, L. Ceze, and V. S. Sathe. Energy-efficient neural network acceleration in the presence of bit-level memory errors. *IEEE Trans. on Circuits and Systems I : Regular Papers*, pages 1–14, 2018.

[12] Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin. Compression of deep convolutional neural networks for fast and low power mobile applications. *arXiv preprint arXiv :1511.06530*, 2015.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[14] C. Liu, M. Hu, J. P. Strachan, and H. Li. Rescuing memristor-based neuromorphic design with high defects. In *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, June 2017.

[15] Brandon Reagen, Paul Whatmough, Robert Adolf, Saketh Rama, Hyunkwang Lee, Sae Kyu Lee, José Miguel Hernández-Lobato, Gu-Yeon Wei, and David Brooks. Minerva : Enabling low-power, highly-accurate deep neural network accelerators. In *Proc. 43rd Int. Symp. on Computer Architecture (ISCA'16)*, pages 267–278, Piscataway, NJ, USA, 2016. IEEE Press.

[16] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2 : Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[17] Guillaume Soulié, Vincent Gripon, and Maëlys Robert. Compression of deep neural networks on the fly. In *International Conference on Artificial Neural Networks*, pages 153–160. Springer, 2016.

[18] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout : a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1) :1929–1958, 2014.

[19] Jean-Charles Vialatte and François Leduc-Primeau. A study of deep learning robustness against computation failures. In *Proc. 9th Int. Conf. on Advanced Cognitive Technologies and Applications*, Feb. 2017.

[20] P. N. Whatmough, S. K. Lee, H. Lee, S. Rama, D. Brooks, and G. Wei. A 28nm soc with a 1.2ghz 568nj/prediction sparse deep-neural-network engine with $>$0.1 timing error rate tolerance for IoT applications. In *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 242–243, Feb 2017.

[21] L. Xia, M. Liu, X. Ning, K. Chakrabarty, and Y. Wang. Fault-tolerant training enabled by on-line fault detection for RRAM-based neural computing systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 1–1, 2018.