Élagage de réseaux profond de neurones par dégradation sélective des pondérations

Hugo TESSIER[1, 2], Vincent GRIPON[2], Mathieu LÉONARDON[2], Matthieu ARZEL[2], Thomas HANNAGAN[1], David BERTRAND[1]

[1]Stellantis, Centre Technique Vélizy
Vélizy-Villacoublay 78140, France

[2]IMT Atlantique, Lab-STICC
UMR CNRS 6285, F-29238, France

hugo.tessier@stellantis.com, vincent.gripon@imt-atlantique.fr,
mathieu.leonardon@imt-atlantique.fr, matthieu.arzel@imt-atlantique.fr,
thomas.hannagan@stellantis.com, david.bertrand@stellantis.com

Résumé – Les réseaux de neurones profonds sont le standard incontournable de l'apprentissage automatique. Cependant, pour atteindre les meilleures performances, ils requièrent des millions de paramètres entraînables, résultant en des architectures lourdes en calculs et en mémoire, et donc peu adaptées à certains contextes applicatifs comme l'embarqué. L'élagage des paramètres pendant l'entraînement est une méthodologie fréquemment mise en œuvre pour réduire ces coûts, mais il induit de nouveaux problèmes : effondrement soudain des performances à fort taux d'élagage, discontinuités entre les phases de l'entraînement... Dans ce papier nous introduisons la Pénalisation Sélective des Pondérations (PSP), une méthode inspirée du lissage lagrangien et permettant un élagage progressif et continu pendant l'entraînement. Nous montrons sur des jeux de données standards la capacité de cette méthode à atteindre les meilleures performances, notamment aux plus forts taux d'élagage.

Abstract – Deep neural networks are the standard in machine learning. However, to achieve the best performance, they require millions of trainable parameters, resulting in computationally and memory intensive architectures, and therefore not well suited to certain application contexts such as embedded systems. Parameter pruning during training is a frequently used methodology to reduce these costs, but it induces new problems: sudden performance collapse at high pruning rates, discontinuities between training phases... In this paper we introduce Selective Weight Decay (SWD), a method inspired by Lagrangian smoothing and allowing a progressive and continuous pruning during training. We show on standard datasets the ability of this method to achieve the best performances, especially at the highest pruning rates.

1 Introduction

Les réseaux profonds de neurones sont devenus l'état de l'art dans de nombreux domaines comme la vision par ordinateur, notamment depuis qu'ils ont démontré leur efficacité lors de compétitions telles qu'ImageNet en 2012. Toutefois, leur coût en terme de mémoire ou de calcul peut être prohibitif sur matériel embarqué [1]. De fait, le domaine de la compression est devenu essentiel pour réduire ces coûts. Il comporte plusieurs familles de techniques, dont la quantification [2] ou la distillation [5].

La méthode à laquelle nous nous intéressons est l'élagage, consistant à éliminer des parties peu utiles d'un réseau pour en réduire le coût, bien qu'elle fût originellement développée pour en améliorer la généralisation [7]. Les parties éliminées peuvent être aussi bien des connexions éparses [4] que des neurones entiers [8], cet aspect étant important pour savoir quel gain espérer. En effet, un élagage « non structuré » n'est pas forcément exploitable, du fait de la difficulté d'accélérer le produit de matrices creuses, là où un élagage « structuré » l'est de sorte à produire des architectures réduites facilement exploitables. Leur importance est déterminée par une métrique telle que la norme des pondérations [4] ou celle de leur gradient [10]. Le retrait lui-même des parties ciblées peut être itératif [4], intégrer un principe de repousse des connexions [3] ou encore se faire par le biais d'une régularisation définie par inférence variationnelle [9] – la littérature sur le domaine présente de très nombreuses méthodes différentes.

Parmi tous les problèmes auxquels est confronté le domaine de l'élagage, l'un des principaux est la cohabitation entre le besoin du processus d'apprentissage d'être laissé suffisamment libre pour trouver une solution pertinente [6] et la contrainte dure que fait peser l'élagage sur la valeur des pondérations. Notre méthode, la PSP, résout ce problème en relaxant la contrainte de l'élagage sous la forme d'une pénalisation quadratique sélective dont l'importance croît au court de l'entraînement – en cela, nous nous inspirons de la méthode du lissage lagrangien [11].

Requérant peu d'hyperparamètres, la PSP obtient des résultats convaincants sur différentes tâches, notamment dans le cas de taux d'élagage très agressifs pour lesquels les autres méthodes de référence réduisent les performances des réseaux à l'aléatoire. Dans les parties suivantes, nous allons décrire pré-

cisément le fonctionnement de la PSP de même que plusieurs expériences démontrant son efficacité ainsi que la validité de son approche pour élaguer les connexions d'un réseau de neurones. Notre code source est disponible à l'adresse suivante : `https://github.com/HugoTessier-lab/SWD`

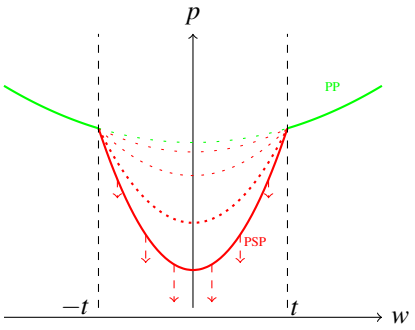


Figure 1 – En abscisse : la valeur des pondérations du réseau, en ordonnée : la pénalité soit de la pénalisation classique des pondérations (PP) ou de notre méthode sélective (PSP). La pente accrue de la PSP en dessous d'un seuil t contraint les pondérations plus fortement vers 0, ce qui les élague à terme. Ce seuil t est adapté à tout moment pour cibler la proportion de pondérations définie par le taux d'élagage cible.

2 Méthode

La Pénalité Sélective des Pondérations (PSP) peut se décrire sous la forme d'un terme de pénalité additionnel dans la fonction objectif d'un réseau $\mathcal{N}$ au cours de son entraînement sur les données $\mathcal{D}$:

$$\mathcal{L}(\mathbf{w}) = \underbrace{\sum_{(x,y)\in\mathcal{D}} \mathcal{E}(\mathcal{N}(x,\mathbf{w}),y)}_{Err} + \underbrace{\mu \|\mathbf{w}\|_2}_{PP} + \underbrace{a\mu \|\mathbf{w}^*\|_2}_{PSP} \quad (1)$$

avec $\mathcal{E}$ la fonction d'erreur, μ le coefficient déterminant l'importance de la pénalité quadratique des pondérations (PP), une méthode classique de régularisation, et a définissant l'importance de notre pénalité sélective sur $\mathbf{w}^*$, le sous-ensemble des pondérations ciblées. Nous avons choisi la norme $\mathcal{L}_2$ pour notre pénalité (la même que pour la régularisation) puisqu'elle constitue une relaxation dérivable pertinente de la norme $\mathcal{L}_0$ classiquement employée pour modéliser la contrainte que fait peser l'élagage sur les pondérations $\mathbf{w}$ du réseau.

La valeur du coefficient a croît au cours de l'entraînement, valant a_{min} au début de celui-ci et a_{max} à la fin (une fois que l'on a atteint l'itération s_{final} de l'entraînement) :

$$a(s) = a_{min}\left(\frac{a_{max}}{a_{min}}\right)^{\frac{s}{s_{final}}} \quad (2)$$

La croissance exponentielle du a permettant de ne pas trop pénaliser l'entraînement tout en assurant l'élimination des connexions au terme de celui-ci, pourvu que a_{max} soit suffisamment grand.

Le choix du $\mathbf{w}^*$ est tout à fait libre, le principe de la PSP n'en dépendant pas, mais dans le cadre de notre contribution, nous avons choisi deux cas d'application : l'élagage des pondérations individuelles de moindre norme [4] ou l'élagage de filtres de convolution sur la base de leur coefficient multiplicatif dans la couche de normalisation de lot suivante [8]. Nous avons ainsi deux exemples d'élagage respectivement non-structuré et structuré. À noter que la proportion de pondérations contenues dans $\mathbf{w}^*$ dépend de l'objectif final d'élagage et que ce $\mathbf{w}^*$ doit être régulièrement mis à jour – nous le recalculons à chaque étape s d'entraînement (pour un coût de calcul peu significatif).

3 Expériences

3.1 Conditions générales d'entraînement

Chaque série d'expérience a été menée sous les mêmes conditions (hormis mention explicite) en utilisant la même initialisation. Nous avons mené les expériences sous Pytorch, en utilisant l'optimiseur SGD, avec un taux d'apprentissage de 1e−1 (décroissant par palier de puissance 10 tous les 100 cycles d'apprentissage) et un moment de 0.9. Nous utilisons l'initialisation par défaut de Pytorch.

3.2 Méthodes de référence

Élagage non-structuré : Han et al.[4] Cette méthode consiste à éliminer, après l'entraînement, un taux croissant des pondérations de moindre norme en 5 itérations entrecoupées de quelques cycles d'entraînement au plus bas taux d'apprentissage.

Élagage structuré : Liu et al.[8] Cette méthode consiste à éliminer, en une fois après l'entraînement, les filtres de convolution dont le coefficient multiplicatif, situé dans la couche suivante de normalisation par lot, est de moindre norme. Afin de faciliter l'élagage, la norme des pondérations de ces couches est pénalisée par une norme $\mathcal{L}_1$ douce.

Rembobinage du taux d'apprentissage (LRR) : Renda et al.[12] Cette méthode effectue un réentraînement complet post-élagage, ce qui peut considérablement améliorer les performances. Comme il s'effectue en réinitialisant le taux d'apprentissage à sa valeur d'origine, cette méthode est appelée « rembobinage ». Du fait du coût accru de l'entraînement, quand nous combinons cette technique avec celle de Han et. al., nous ne procédons plus de manière itérative mais en une fois.

3.3 Résultats sur ImageNet ILSVRC2012

La table 1 permet de comparer les performances obtenues par la PSP et les autres méthodes de référence, pour le réseau ResNet-50 entraîné sur ImageNet ILSVRC2012. La précision du réseau non-élagué est de 75.7 en top-1 et 92.8 en top-5. Le coefficient de la pénalité quadratique μ vaut 1e−4, les réseaux ont été entraînés durant 90 cycles et chaque itération d'élagage fut ensuivie de 5 cycles (excepté la dernière qui est suivie par 15

Cible	Han et. al.[4]		+LRR[12]		PSP	
	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
50	74.9	92.2	58.4	82.1	**75.0**	**92.2**
90	71.1	90.5	54.6	79.6	**73.1**	**91.3**
97.5	47.2	73.2	34.8	61.54	**67.8**	**88.4**

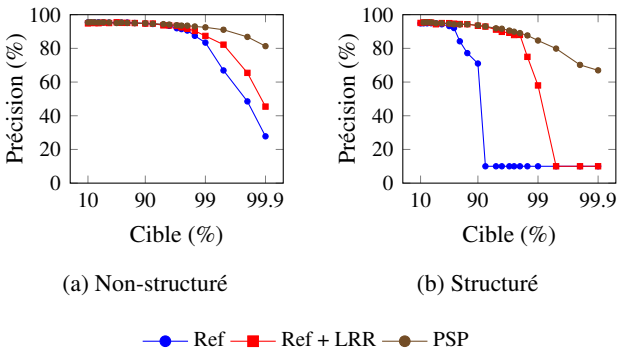
Structuré

Cible	Liu et. al.[8]		+LRR[12]		PSP	
	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
10	**74.7**	**92.2**	56.1	80.7	74.2	91.9
25	73.4	91.6	51.1	77.1	**73.5**	**91.5**
50	63.6	85.7	40.0	66.2	**69.0**	**88.8**
80	0.1	0.5	0.1	0.5	**69.0**	**88.7**

Table 1 – Résultats de ResNet-50 sur ImageNet ILSVRC2012. Pour différentes taux cible d'élagage et différentes méthodes, nous présentons les performance en précision top-1 et top-5. Toutes les valeurs sont en pourcent.

cycles). Le coefficient de la pénalité $\mathscr{L}_1$ de Liu et. al. [8] est de $\lambda = 1e{-}5$. Les valeurs de a_{min} et a_{max} de la PSP valent $1e{-}1$ et 1e5 dans le cas non-structuré ou 1e1 et $a_{max} = 1e4$ dans le cas structuré. Les réseaux élagués par PSP ont été ré-entraînés pendant 30 cycles.

3.4 Résultats sur CIFAR-10



(a) Non-structuré (b) Structuré

Figure 2 – Comparaison de la précision obtenue par une méthode de référence (Han et. al. [4] ou Liu et. al. [8]), la même méthode adjointe au rembobinage (LRR) et notre méthode (PSP), pour plusieurs taux cible d'élagage (abscisse logarithmique), sur le réseau ResNet-20 entraîné sur CIFAR-10.

La figure 2 compare les résultats de notre méthode à ceux des méthodes de référence sur le réseau ResNet-20 (avec une largeur initiale de 64 filtres de convolution) entraîné sur CIFAR-10. Les réseaux ont tous été entraînés durant 300 cycles et chaque ré-entraînement (hors rembobinage) a duré 15 cycles (excepté le dernier qui en dure 50). Les réseaux élagués par PSP n'ont pas été réentraînés. Le coefficient μ vaut $5e{-}4$, la pénalité $\mathscr{L}_1$ douce est à $1e{-}4$. Les coefficients de PSP sont fixés comme suit : $a_{min} = 1e{-}1$ et $a_{max} = 1e5$ en non-structuré ; $a_{min} = 1e2$ et $a_{max} = 1e7$ en structuré.

3.5 Étude des hyperparamètres

$a_{début}$	1e4	1e3	1e2	1e1	1e0	1e−1	1e−2	1e−3	1e−4	1e−5
a_{fin}	**Précision post-élimination (%)**									
1e1	94.44	94.08	93.92	94.21	94.54	91.00	83.89	53.68	71.88	67.07
1e2	94.52	94.13	93.91	94.00	94.65	95.15	94.55	93.99	92.40	89.87
1e3	94.57	94.23	93.50	94.00	94.72	94.96	95.29	95.27	94.81	94.72
1e4	94.61	94.17	93.85	94.49	94.50	94.73	95.37	95.29	95.14	95.22
1e5	94.37	94.45	93.54	94.39	94.39	94.78	95.24	95.07	95.30	95.19
	Dégradation par l'élimination (%)									
1e1	-0.1	0	-0.01	0	0.06	-4.33	-11.49	-41.72	-23.53	-28.36
1e2	0.06	-0.09	-0.01	-0.05	0.1	-0.01	-0.45	-0.99	-2.37	-4.74
1e3	-0.01	0.03	-0.01	0	0.02	-0.01	-0.06	0.05	-0.01	0.11
1e4	-0.02	-0.07	0.01	0.01	0.02	-0.03	0.02	0.05	-0.02	0.08
1e5	0.01	0.04	0.02	0	0	-0.01	-0.02	-0.01	0.01	-0.10

Table 2 – Performance finale et dégradation introduite par l'élimination terminale des pondérations, dans le cas de la PSP appliquée pour l'élimination non-structurée de 90% des pondérations d'un ResNet-20 entraîné sur CIFAR-10.

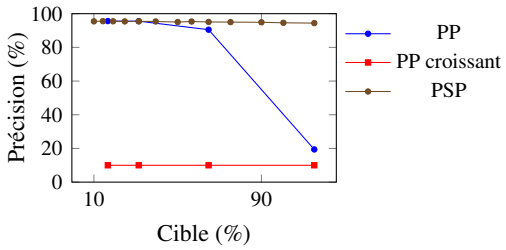


Figure 3 – Précisions top-1 d'un ResNet-20 entraîné sur CIFAR-10 et élagué à différents taux cible d'élagage (abscisse logarithmique), lorsque le réseau est soumis soit uniquement à une régularisation classique (PP), à une pénalisation globale croissant (PP croissant) ou à la PSP elle-même.

La table 2 montre, sur un réseau ResNet-20 entraîné sur CIFAR-10, comment les valeurs de début et de fin de a influent sur les performances de la PSP à un même taux d'élagage de 90%. Parmi les couples de valeurs présentés, on peut comparer le cas d'un a croissant, constant ou décroissant. Nous présentons aussi bien la précision post-élimination (c'est à dire après que les pondérations ciblées par PSP aient été définitivement retirées du réseau) que la dégradation apportée par cette élimination : si la PSP est suffisante pour éliminer les pondérations, alors cette élimination ne devrait pas avoir d'impact. Cette étude permet donc de vérifier la validité de la PSP comme méthode d'élimination.

La figure 3 permet d'étudier l'importance d'appliquer la PSP sur un ensemble retreint $\mathbf{w}^*$ des pondérations du réseau. Dans le cas d'une pénalité croissante non-sélective, toutes les pondérations du réseau se retrouvent éliminées, tandis que la PSP permet au contraire de meilleures performances que dans le cas où

seule une régularisation classique est appliquée. Ces résultats ont aussi été obtenus sur ResNet-20 entraîné sur CIFAR-10.

4 Discussion

Les résultats de la table 1 et de la figure 2 montrent l'efficacité de la PSP pour l'élagage non-structuré et structuré, sur plusieurs jeux de données différents. La PSP se distingue nettement dans le cas de taux d'élagage très importants, toutes les méthodes ici comparées tendant à se confondre pour les taux les plus bas. La PSP se montre capable d'éviter une destruction catastrophique (par le phénomène d'effondrement des couches [13]) des réseaux élagués que les autres méthodes semblent provoquer dans le cas d'élagage structuré à très haut taux. Ce dernier point tend à confirmer qu'une méthode d'élagage progressive, continue et capable d'ajuster au fur et à mesure l'ensemble des pondérations ciblées est plus à même de sélectionner les parties pertinentes à éliminer sans infliger de dégâts critiques susceptibles de réduire les performances du réseau à l'aléatoire.

Quant aux résultats de la table 2, ils permettent de démontrer deux choses : 1) les meilleurs résultats sont obtenus pour un a croissant depuis une valeur basse vers une valeur très haute (ce qui confirme notre justification de la section 2) et 2) une valeur suffisamment haute de a_{max} est nécessaire pour que l'élimination terminale des pondérations ciblées soit transparente dans son impact sur les performances du réseau, ce qui montre qu'elles étaient bel et bien éliminés car d'impact nul sur la fonction du réseau.

5 Conclusion

Nous avons proposé une méthode d'élimination des connexions d'un réseau pouvant être appliquée à n'importe quelle structure selon n'importe quel critère dans le cadre de l'élagage de réseaux profonds de neurones. Cette méthode est la Pénalisation Sélective des Pondérations (PSP) et permet une élimination progressive, continue et adaptative des connexions au cours de l'entraînement. Nos résultats ont montré que cette méthode permet des gains très nets en performance, par rapport à nos méthodes de référence, pour les forts taux d'élagage. Cette méthode nécessite l'introduction de nouveaux hyperparamètres, mais nos expériences suggèrent une assez faible sensibilité des performances à ces derniers, participant à faire de PSP une méthode que nous pensons adaptée à de nombreux contextes applicatifs.

References

[1] Ghouthi Boukli Hacene. *Processing and learning deep neural networks on chip*. PhD thesis, Ecole nationale supérieure Mines-Télécom Atlantique Bretagne Pays de la Loire, 2019.

[2] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3123–3131. Curran Associates, Inc., 2015.

[3] Tim Dettmers and Luke Zettlemoyer. Sparse networks from scratch: Faster training without losing performance. *arXiv*, pages arXiv–1907, 2019.

[4] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1135–1143. Curran Associates, Inc., 2015.

[5] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *stat*, 1050:9, 2015.

[6] Yann Le Cun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[7] Yann Le Cun, John S. Denker, and Sara A. Solla. *Optimal Brain Damage*, page 598–605. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[8] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[9] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *International Conference on Machine Learning*, pages 2498–2507. PMLR, 2017.

[10] P Molchanov, S Tyree, T Karras, T Aila, and J Kautz. Pruning convolutional neural networks for resource efficient inference. In *5th International Conference on Learning Representations, ICLR 2017-Conference Track Proceedings*, 2019.

[11] Walter Murray and Kien-Ming Ng. Algorithm for nonlinear optimization problems with binary variables. *Computational Optimization and Applications*, 47:257–288, 10 2010.

[12] Alex Renda, Jonathan Frankle, and Michael Carbin. Comparing rewinding and fine-tuning in neural network pruning. *arXiv preprint arXiv:2003.02389*, 2020.

[13] Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. *Advances in Neural Information Processing Systems*, 33, 2020.