Compression de réseaux convolutifs par utilisation d'un terme de clarté l_1/l_2 sur les noyaux

Anthony Berthelier[1] Yongzhe Yan[1] Thierry Chateau[1] Christophe Blanc[1]
Stefan Duffner [2] Christophe Garcia [2]

[1] Université Clermont Auvergne, France
[2] Université de Lyon, CNRS, INSA Lyon, LIRIS, UMR5205, Lyon, France

anthony.berthelier@etu.uca.fr

1 Introduction

Depuis l'avènement des réseaux de neurones profonds (DNNs) et des réseaux de neurones convolutifs (CNNs) en particulier, les méthodes d'apprentissage profond sont devenues des standards pour la plupart des tâches d'analyses d'images (reconnaissance faciale, segmentation sémantique, détection d'objets...). Afin d'obtenir de telles performances, ces modèles ont besoin d'importantes capacités de calcul et mémorielles afin de pouvoir gérer les millions de paramètres qu'ils contiennent. Ces pré-requis sont un frein pour le fonctionnement de ces modèles sur des systèmes embarqués par exemple.
Par conséquent, de nombreuses études ont été menées afin de compresser ces réseaux (knowledge distillation, précision numérique...). Cependant les méthodes les plus élaborées sont celles permettant de réduire leur redondance. Parmi celles-ci, les méthodes de quantification et d'élagage (pruning) sont les plus utilisées [1]. Néanmoins, la plupart d'entre-elles conduisent vers des architectures manquant de structures et créant de trop nombreuses connexions irrégulières entre les paramètres, cela ayant pour conséquence une mauvaise gestion de la mémoire. Il est aussi important de noter que la majorité de ces méthodes s'appliquent en dehors de l'apprentissage initial du modèle, nécessitant de lourdes opérations de fine-tuning.
Dans cet article, nous proposons d'ajouter à la fonction de *loss* un terme de clarté (sparsity). Cette clarté s'applique sur l'ensemble constitué des normes des poids de chaque noyau de convolution et est définie comme étant le nombre de poids d'un réseau mis à zéro. Ainsi, motivés par (1) la redondance des paramètres dans un réseau profond [2] ; (2) le fait que de nombreuses méthodes de clarté existent [3] mais que (3) très peu d'entre elles sont utilisées pour supprimer des poids durant l'apprentissage [4], nous proposons une méthode de clarté permettant d'apprendre durant l'apprentissage une structure compressée de CNNs en utilisant une norme de régularisation l_1/l_2. Plus précisément, la formule générale pour introduire de la clarté dans un groupe est la norme l_1/l_q avec $1 < q \leq \infty$. Cependant nous ne nous intéressons dans ce papier qu'à la norme l_1/l_2 pour deux raisons. Premièrement, cette norme est la plus simple à implémenter parmi l'ensemble des normes possibles. Deuxièmement, nous supposons que l'utilisation de la norme l_1, améliorant les performances, l'interprétabilité et la clarté des modèles, couplée avec la norme l_2 permet d'obtenir des structures de réseaux plus stables.
Dans cet article, la norme l_1/l_2 est uniquement utilisée au niveau des filtres de CNNs pour en compresser leur structure. Nous ne nous intéressons ainsi qu'à la régularisation et la compression des couches convolutives dans un CNN. Nous montrons que notre méthode permet d'augmenter la clarté de leurs paramètres sans baisses significatives de leurs précisions.

2 Méthode d'apprentissage avec la norme l_1/l_2

Nous supposons que $\boldsymbol{W}$ représente l'ensemble des poids d'un DNN. Les poids d'une couche convolutive forment un tenseur 4D $\boldsymbol{W}_C^{(l)} \in R^{O_l \times I_l \times Kh_l \times Kw_l}$, où I_l représente le nombre de canaux en entrée ; O_l est le nombre de canaux en sortie ; et Kh_l et Kw_l sont les dimensions en hauteur et largeur des filtres. Nous définissons aussi $\boldsymbol{W}_C$ comme l'ensemble des poids de toutes les couches convolutives d'un CNN. A partir de ces définitions, nous proposons la fonction de coût suivante :

$$E(\boldsymbol{W}) = E_D(\boldsymbol{W}) + \lambda . \frac{L_1(\boldsymbol{W}_C)}{L_2(\boldsymbol{W}_C)} \tag{1}$$

$E_D(\boldsymbol{W})$ est une fonction de coût classique sur les données ; λ est un coefficient de régularisation et $L_1(.)$ et $L_2(.)$ sont respectivement les normes l_1 et l_2 appliquées sur l'ensemble des couches convolutives. Le ratio de ces deux fonctions constitue notre terme de régularisation permettant d'introduire de la clarté dans les paramètres. Ces deux fonctions se définissent de la façon suivante : $L_1(\boldsymbol{W}_C) = \sum N(\boldsymbol{W}_C)$ et $L_2(\boldsymbol{W}_C) = \sqrt{\sum N(\boldsymbol{W}_C)^2}$. Dans ces deux équations, la fonction $N(.)$ est in-

troduite. Comme précisé précédemment, les poids d'une couche convolutive sont dans un tenseur 4D. Ainsi, dans une couche convolutive l ($1 \leq l \leq L$), nous considérons qu'il y a O_l filtres et que chacun de ces filtres a une dimension $I_l \times Kh_l \times Kw_l$. Pour la couche l, la fonction $N(\boldsymbol{W}_C^{(l)})$ a pour but de construire un vecteur composé de O_l valeurs, chacune étant la norme d'un des filtres O_l. Cette norme est le résultat de la somme des valeurs absolues de chaque élément de la matrice 3D de taille $I_l \times Kh_l \times Kw_l$. Ces opérations peuvent être formulées de la manière suivante : $N(\boldsymbol{W}_C^{(l)}) = \begin{bmatrix} n_{(1)} & \dots & n_{(k)} & \dots & n_{(O_l)} \end{bmatrix}$ avec $n_{(k)} = \sum_{i=1}^{I_l} \sum_{j=1}^{Kh_l} \sum_{z=1}^{Kw_l} \mid x_{(k)ijz} \mid$. Une fois que tous les vecteurs $N(\boldsymbol{W}_C^{(l)})$ sont construits pour chacune des L couches convolutives, nous les concaténons dans un vecteur $N(\boldsymbol{W}_C) = ||_{l=1}^{L} N(\boldsymbol{W}_C^{(l)})$. Les fonctions $L_1(.)$ et $L_2(.)$ calculent les normes l_1 et l_2 sur ce vecteur 1D. La norme l_1/l_2 fait converger vers zéro certains paramètres, néanmoins, un seuil est aussi défini pour que ces valeurs soient réellement mises à zéro si leur valeur absolue passe en dessous de ce seuil.

3 Premières expérimentations sur *LeNet*

Nous expérimentons notre méthode avec *LeNet* [5] sur le jeu de données MNIST afin de pouvoir nous comparer à la méthode SSL [4], basée sur la régularisation Lasso et elle aussi réduisant le nombre de filtres pendant l'apprentissage. Les résultats obtenues sont consignés dans la TABLE 1 (a). Nos tests montrent que notre méthode est capable d'obtenir la précision du modèle de référence tout en supprimant la majorité des filtres présents dans les deux couches convolutives (les couches 1 et 2 passent respectivement de 20 à 4 filtres et de 50 à 12 filtres). Comparée à la méthode SSL, notre méthode est capable d'obtenir une précision similaire tout en réduisant à zéro plus de filtres. Enfin nous voulions aussi prouver l'efficacité de notre méthode face à des normes l_1 et l_2 classiques, elles aussi appliquées sur les filtres. Nous en arrivons aux mêmes conclusions. Ces deux normes sont capables d'atteindre le niveau de précision de référence mais ne sont pas capables de supprimer autant de filtres que notre méthode. Afin de tester notre méthode sur un problème de classification plus difficile, nous avons répété ces tests (à l'exception de SSL, non implémentée) sur le jeu de données CIFAR-10 (TABLE 1 (b)). Nous remarquons que bien qu'aucune des méthodes n'arrivent à atteindre le niveau de précision de la référence, notre méthode est celle supprimant le plus de filtres tout en ayant la plus haute précision. Le trade-off entre suppression de filtres et précision du modèle dépends de λ, fixé manuellement mais d'autres méthodes sont à explorer. Ce travail se poursuit désormais en appliquant cette méthode sur des réseaux plus complexes telles que des CNNs plus profonds, des réseaux entièrement convolutionnels et pour effectuer des tâches qui vont au-delà de la classification. La généralisation de notre méthode à la norme l_1/l_q est aussi envisagée.

Références

[1] S. Han, H. Mao, and W. J. Dally, "Deep Compression - Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," *ICLR*, 2016.

[2] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," *BMVC*, 2014.

[3] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski, "Optimization with sparsity-inducing penalties," *Found. Trends Mach. Learn.*, 2012.

[4] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," *NIPS*, 2016.

[5] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, 1998.

Méthode	λ	Erreur	Filtres Conv1	Filtres Conv2
Référence	-	0.9%	20	50
l_1	-	0.9%	7	21
l_2	-	0.9%	5	32
SSL 1	-	0.8%	5	19
SSL 2	-	1.0%	3	12
l_1/l_2	0.005	1.3%	2	9
l_1/l_2	0.001	0.9%	4	12

(a) Expérimentations avec *LeNet* sur MNIST

Méthode	λ	Erreur	Filtres Conv1	Filtres Conv2
Référence	-	28.4%	20	50
l_1	-	29.8%	13	42
l_2	-	31.2%	8	43
l_1/l_2	0.005	50.9%	2	8
l_1/l_2	0.001	30.1%	11	31
l_1/l_2	0.0005	29.4%	9	41

(b) Expérimentations avec *LeNet* sur CIFAR-10

TABLE 1 – Erreur et nombre de filtres pour chacune des deux couches convolutive de *LeNet* sur (a) MNIST et (b) CIFAR-10 testé avec différentes méthodes de régularisation. Référence est le modèle *LeNet* entraîné sans contraintes. l_1 et l_2 sont les meilleurs résultats trouvés en utilisant ces normes. SSL est la méthode tirée de [4]. l_1/l_2 est notre méthode avec différents λ.