Compression adaptative des CNNs par réduction de la dimensionnalité

LUCAS FERNANDEZ BRILLET[1,2], STEPHANE MANCINI[2], SEBASTIEN CLEYET-MERLE[1], MARINA NICOLAS[1]

[1] Imaging Division, STMicroelectronics Grenoble SAS, France
[2] Univ. Grenoble Alpes, CNRS, Grenoble INP, TIMA, Grenoble, France
[1] lucas.fernandezbrillet@st.com, sebastien.cleyet-merle@st.com, marina.nicolas@st.com
[2] stephane.mancini@univ-grenoble-alpes.fr

Résumé – La complexité computationnelle des réseaux de neurones à base de convolution (CNNs) rend leur intégration dans des systèmes embarqués complexe. Des méthodes permettant la simplification de ces algorithmes présentent alors un grand intérêt. Dans cet article, nous proposons une nouvelle méthode de compression des CNN fondée sur l'application d'une analyse en composantes principales (PCA) couche par couche et montrons l'intérêt d'une phase de réentrainement additionnelle. Grâce à cette méthode, il est possible de trouver des compromis paramétrables entre la taille et la performance du réseau, comme par exemple, une réduction par 2 du nombre de paramètres en échange d'une perte en performance de l'ordre de 2%. Nous considérons aussi la compatibilité de cette méthode avec d'autres méthodes de compression bien connues dans l'état de l'art.

Summary – Computational complexity of CNNs makes its integration in embedded systems a challenging task. In this paper, we propose a new CNN compression method based on the application of PCA in a layer-wise fashion, and show the benefits of an additional fine-tuning step. Through this method, it is possible to reach very flexible trade-offs, such as a x2 reduction in the number of parameters for an accuracy drop inferior to 2%.

1 Introduction

Les réseaux de neurones à base de convolution (CNN) sont récemment devenus la solution de l'état de l'art pour des problématiques de vision par ordinateur et ont le potentiel de remplacer des algorithmes plus traditionnels. Les capteurs d'image intelligents peuvent énormément bénéficier de cette capacité dans leurs traitements embarqués.

Cependant, cette transition vers les CNNs requiert une adaptation importante des algorithmes et architectures matérielles de traitement. Dans un contexte très contraint en surface et consommation, les transferts de données doivent être évités et les coefficients et données intermédiaires doivent rentrer dans la mémoire de la puce.

Dans cet article, nous proposons une nouvelle méthode pour la compression des CNNs. Cette méthode est fondée sur l'application d'une analyse en composantes principales (PCA) aux différentes couches du CNN. Nous montrons aussi l'intérêt d'une phase additionnelle de réentrainement. Avec cette méthode, il est possible de trouver des compromis paramétrables entre la taille et la performance du modèle, comme par exemple une réduction de 50% pour une perte de 2%.

L'article commence par une revue de l'état de l'art des techniques de compression appliquées aux CNNs. Nous appliquons en suite la PCA sur plusieurs réseaux et bases de données et montrons le bénéfice d'une phase additionnelle de réentrainement.

2 Travaux connexes

Les différentes méthodes adaptant les modèles CNN à des implémentations matérielles peuvent être classées selon les catégories suivantes:

La quantification : elle permet la réduction du nombre de bits utilisés pour représenter les données. Les bénéfices sont un coût réduit en termes de stockage et de calcul des opérateurs matériels [1].

Le pruning : il réduit le nombre de paramètres en remplaçant les valeurs faibles par des zéros. Les gains en mémoire sont obtenus en codant les positions des coefficients non nuls. Cependant, des blocs matériels additionnels sont nécessaires [2,3].

Le codage entropique : il s'agit d'une méthode de compression sans perte dans laquelle les symboles plus fréquents sont encodés par des codes plus courts [4].

La distillation : plusieurs travaux [5] montrent qu'il est possible de transférer les connaissances d'un modèle complexe vers un modèle plus simple avec une meilleure performance que dans le cas où le modèle simple est entrainé directement.

La transformation de matrices : plusieurs travaux montrent [6,7] qu'il est possible de trouver des représentations alternatives mieux adaptés aux CNNs. La technique proposée dans cet article fait partie de cette catégorie.

Ces méthodes constituent des approches complémentaires visant la simplification de complexité computationnelle des CNNs et sont toutes largement compatibles entre elles.

3 Compression des CNNs par PCA

L'analyse en composantes principales est une technique de réduction de la dimensionnalité. Elle est utilisée dans notre cas pour décomposer un ensemble de filtres appris en une combinaison linéaire de filtres de base et des coordonnées dans cette base. La compression consiste à choisir un sous-ensemble de la base PCA pour ensuite reconstruire une approximation des filtres.

L'idée est alors d'appliquer cette transformation couche par couche, afin de stocker une quantité réduite de paramètres, en échange de quelques calculs additionnels, comme montré sur la Figure 1.

Plus concrètement, pour une couche de convolution constituée de N filtres appris X_i, PCA permet de trouver une base orthonormée P de l'espace formé par ces filtres en diagonalisant la matrice centrée de covariance empirique $C_X \in M_{d\times d}$, donnée par $C_X = X_c^T X_c$, où :

- $X = [X_0, X_1, \dots, X_{N-1}] \in M_{N\times d}$ est une matrice construite en regroupant empilant les N filtres $X_i \in \mathbb{R}^d$ dans la couche,
- $X_c = X - \mu$ où $\mu = \frac{1}{N}\sum_{i=0}^{N-1} X_i$ et $\mu \in \mathbb{R}^d$.
- $d = f_h \times f_w \times f_c$ est la taille de chacun des N filtres 3-D aplatis,

Le but étant de trouver P tel que $Y_c = X_c P$, Y_c étant une nouvelle représentation des filtres telle que C_Y soit diagonale (la covariance est alors nulle après cette transformation).

C_X étant une matrice symétrique, elle peut être diagonalisée sous la forme $C_X = PDP^T$, où P est orthogonale, ses colonnes étant formées par les vecteurs propres $\{v_d\}$ de C_X.

Les lignes de Y_c peuvent être considérées comme les coordonnées $\{\lambda_d^i\}$ dans la base orthogonale P et il est alors possible d'écrire les filtres originaux comme $X_i = \sum_{k=0}^{d-1} \lambda_k^i v_k + \mu$, où de manière équivalente $X = Y_c P^{-1} + \mu$.

PCA étant une transformation orthogonale, la quantité d'énergie (norme de Frobenius de la matrice de transformation) est préservée par PCA et est contenue dans les premières valeurs propres. Cette propriété permet de réduire la dimensionnalité en gardant uniquement les vecteurs propres associés aux valeurs propres les plus grandes.

En d'autres termes il est possible d'approcher un filtre original quelconque, avec une précision arbitraire, en gardant un sous-ensemble Q de vecteurs propres et leurs coordonnées associées par $\hat{X}_i = \sum_{k=0}^{Q-1} \lambda_k^i v_k + \mu$, ou de manière équivalente $\hat{X} = Y_{cQ}(P^{-1})_Q + \mu$.

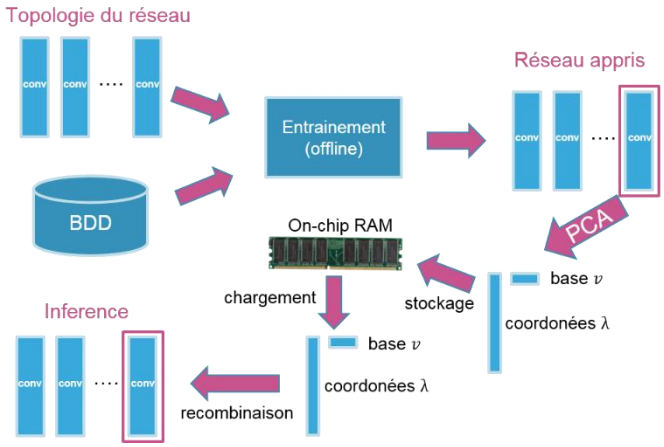

Fig. 1: Schéma de la méthode.

3.1 Reconstruction naïve des filtres

3.1.1 Principe

Comme présenté dans la section précédente, l'application directe de la PCA permet la décomposition de l'ensemble des filtres d'une couche en un ensemble de filtres de base avec leurs coordonnées dans cette base. Cela permet la reconstruction des filtres originaux avec une erreur quadratique moyenne décroissant avec le nombre de composantes principales qui sont gardées lors de la reconstruction. Par exemple, dans la figure 2, une couche comportant 128 filtres est reconstruite à partir de ses 11 premières composantes principales avec une erreur quadratique moyenne de 6.83 x 10^{-5}.

Les gains en mémoire sont obtenus en stockant un sous-ensemble des filtres de la base P_Q et leurs coordonnées correspondantes Y_{cQ}. Lors de l'inférence, les filtres sont reconstruits par un produit matriciel avant la convolution.

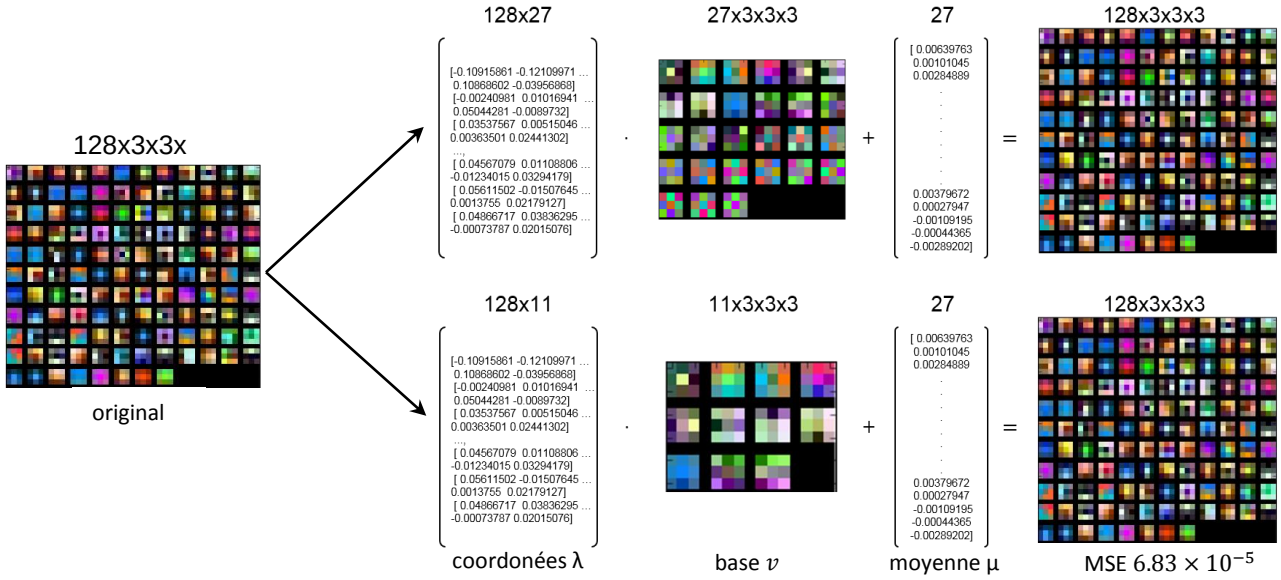

Fig. 2: Décomposition par PCA de la première couche du réseau VGG-16 sur CIFAR-10. La figure supérieure montre une reconstruction parfaite des filtres originaux en gardant tous les filtres de base. La figure inférieure montre une approximation obtenue en gardant uniquement les 11 premiers filtres de base.

3.1.2 Experimentation

Dans les expériences ci-dessous, les filtres d'origine sont remplacés par leur version reconstruite pour plusieurs seuils de reconstruction et la performance du réseau est mesurée pour plusieurs réseaux de l'état de l'art entrainés sur deux bases de données différentes.

Les Tableaux 1 & 2 montrent les compromis entre la taille des paramètres et la performance du réseau obtenus grâce à la PCA, pour CIFAR-10 & 100 [8].

Le tableau 1 montre qu'il est possible, par exemple, de réduire la taille d'un CNN d'un facteur 1.9 pour une perte de seulement 0.8 % d'erreur lorsque l'on conserve une base correspondant à 70% de l'énergie.

À partir du Tableau 1, on peut aussi remarquer que la taille initiale du modèle a un impact sur le taux de compression. Typiquement, les modèles plus efficaces comme ResNet-32[9] subissent une dégradation de performance plus rapide que les réseaux plutôt surdimensionnés tels que VGG-16[10]. Par exemple, une compression à 90% d'énergie pour le premier et à 70% d'énergie pour le deuxième, induisent une même erreur relative (-0.81%), alors que les facteurs de gain en mémoire sont de 1.07 et 1.94 respectivement.

Net	E (%)	Orig	90	80	70	60	50
VGG-16	P (M)	7.331	6.485	4.965	**3.778**	2.821	2.061
	G (x)	1.00	1.13	1.48	**1.94**	2.60	3.56
	A (%)	92.20	92.31	92.27	**91.45**	90.21	86.97
	ER (%)	0.0	0.0	0.0	**0. 81**	2.16	5.67
ResNet-110	P (M)	1.723	1.548	**1.26**	1.028	0.825	0.653
	G (x)	1.00	1.11	**1.37**	1.68	2.09	2.64
	A (%)	92.91	92.61	**91.48**	89.75	86.44	80.34
	ER (%)	0.0	0.32	**1.54**	3.40	6.96	13.53
ResNet-32	P (M)	0.453	**0.423**	0.35	0.288	0.232	0.185
	G (x)	1.00	**1.07**	1.29	1.57	1.95	2.45
	A (%)	92.04	**91.29**	89.51	86.75	81.02	67.04
	ER (%)	0.0	**0.81**	2.75	5.75	11.97	27.16

Tab 1 : Application directe de PCA pour CIFAR-10. P: nombre de paramètres, G:gain en mémoire, A: accuracy (taux de détection), ER: erreur relative.

La complexité de la tâche est aussi un facteur important de la dégradation de performance et, pour le même seuil d'énergie, l'erreur relative est supérieure et le gain mémoire plus faible dans le cas de CIFAR-100 (Tableau 2) que pour CIFAR-10 (Tableau 1).

Net	E (%)	Orig	90	85	80	75	70	60
RN 110	P (M)	1.723	1.585	1.445	**1.318**	1.2	1.09	0.894
	G (x)	1.00	1.09	1.19	**1.31**	1.44	1.58	1.93
	A (%)	71.3	68.3	63.3	**55.1**	43.2	32.2	8.4
	ER (%)	0.0	4.21	11.22	**22.72**	39.41	54.84	88.22
RN 32	P (M)	0.453	**0.435**	0.4	0.367	0.336	0.308	0.255
	G (x)	1.00	**1.04**	1.13	1.23	1.35	1.47	1.78
	A (%)	68.1	**56.9**	43.2	29.1	14.9	4.2	1.6
	ER (%)	0.0	**16.45**	36.56	57.27	78.12	93.83	97.65

Tab 2 : Application directe de PCA pour CIFAR-100.

Ces résultats montrent que cette technique est efficace pour identifier les paramètres importants du réseau et permet une approche adaptative pour trouver des compromis entre taille du modèle et performance. Cependant, ces résultats montrent aussi une dégradation rapide de performance, en particulier si l'on applique cette méthode indifféremment sur toutes les couches. Il semble que ceci, ainsi que l'accumulation d'erreurs des couches successives, soit la principale cause de dégradation de la performance. Nous présentons des solutions à ce problème dans la section qui suit.

3.2 Reconstruction des filtres ré-entrainés

Dans cette section, nous explorons la possibilité de combiner PCA et réentrainement pour réduire la dégradation de performance. L'idée est d'ajuster légèrement l'ensemble des coordonnées dans chaque couche à l'aide d'une étape de réentrainement, le point de départ étant les valeurs de coordonnées calculées par la PCA. Les expériences reportées ci-dessous montrent que l'étape de réentrainement permet une amélioration importante de la performance quel que soit le taux de compression par la PCA.

Pour atteindre ce but, la reconstruction des filtres est forcée de manière explicite dans le graphe de TensorFlow[11], comme le montre la Figure 3.

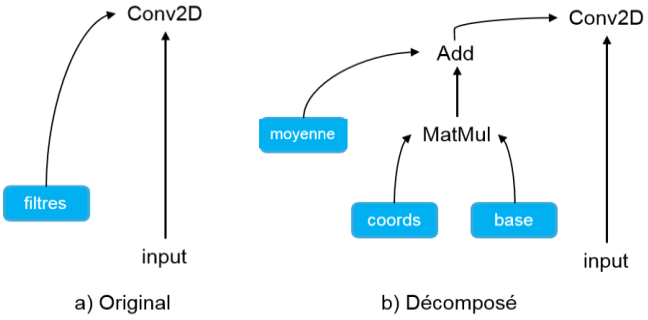

Fig. 3: Décomposition par PCA sous TensorFlow à droite, comparé avec l'original (gauche).

Chaque couche du nouveau réseau est formée par la base PCA, les coordonnées de reconstruction et la moyenne. Ces valeurs sont initialisées à partir de valeurs calculées par PCA. Comme montré sur la Figure 3.b, seules les coordonnées peuvent être entrainées, les autres paramètres restant fixés.

3.2.1 CIFAR-10

Les figures 4 et 5 montrent le bénéfice du réentrainement en comparaison avec l'application directe de la PCA pour ResNet-32 et ResNet-110 respectivement sur CIFAR- 10 (Tableau 1).

Par exemple, dans la Figure 4, le CNN comportant 0.46 Mparams à l'origine peut être compressé à 0.23 Mparams (50% en taux de compression) avec une précision de 89.97% grâce au réentrainement, au lieu d'une précision à 81.02% sans réentrainement.

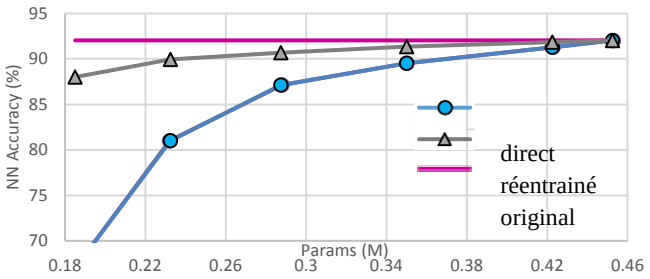

Fig. 4: Bénéfice de l'étape de réentrainement après PCA dans le cas de ResNet-32 sur CIFAR-10.

En comparaison avec la Figure 4, la Figure 5 montre que le réentrainement améliore encore le taux de détection lorsque la taille du modèle d'origine est plus grande, alors même que PCA altère moins le réseau original. Cependant, comme le montre la figure 4, on obtient des modèles finales plus efficaces lorsque le réseau de départ est déjà assez petit.

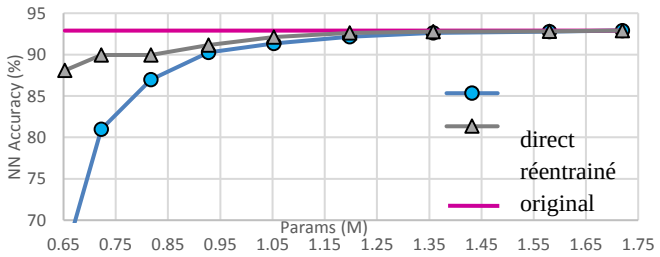

Fig. 5: Bénéfice de l'étape de réentrainement après PCA dans le cas de ResNet-110 sur CIFAR-10.

3.2.2 CIFAR-100

La même expérience est maintenant effectuée dans le cas de CIFAR-100. Les résultats sont reportés dans le Tableau 3. Ces résultats montrent un clair intérêt de l'étape de réentrainement. Par exemple, cette étape permet d'obtenir une précision de 70.06% pour un seuil d'énergie de 80% au lieu de 55.1% sans réentrainement.

Net	E(%)	100	90	85	80	70	60
RN110	P(M)	1.723	1.585	1.445	**1.318**	1.090	0.894
	A(%)	71.3	68.3	63.3	**55.1**	32.2	8.4
	A-R(%)	71.3	71.27	70.61	**70.06**	68.7	-
RN32	P (M)	0.453	**0.435**	0.400	0.367	0.308	0.255
	A(%)	68.1	**56.9**	43.2	29.1	4.2	1.6
	A-R(%)	68.1	**67.34**	-	66.76	65.75	64.13

Tab 3 : Bénéfice du réentrainement (A-R) dans le cas de CIFAR-100 par rapport à l'application directe (A).

La complexité de la tâche à effectuer par le réseau semble toujours être un facteur déterminant. Ainsi le même réseau (ResNet-32), pour un même seuil d'énergie (60%), permet d'obtenir un taux de compression par 2 en échange d'une perte en performance de 2.07% dans le cas de CIFAR-10, alors qu'une compression par 1.78 pour une perte de 3.97% est obtenue dans le cas de CIFAR-100.

4 Perspectives

Une avantage de la méthode proposée est qu'elle est uniquement composée de transformations linéaires couche par couche, ce qui élimine le besoin de HW additionnel, comme dans le cas du pruning [3].

Par ailleurs, la méthode permet une réduction directe du nombre d'opérations, et est aussi compatible avec la quantification et d'autres méthodes d'accélération [12].

De plus, des expériences préliminaires montrent la compatibilité avec le pruning. Ces directions seront explorées dans les travaux à venir.

5 Conclusion

Dans ce travail nous avons appliqué une analyse en composante principales aux différentes couches d'un réseau de neurones.

Nous avons montré que ceci constitue une méthode efficace pour la sélection des paramètres importants dans un CNN, et que cela permet de réduire le nombre de paramètres à stocker pour une dégradation faible de la performance de détection.

Nous avons appliqué avec succès la méthode proposée à des architectures complétement convolutionelles de la littérature considérées efficaces et montré qu'il est possible d'atteindre des taux de compression plus élevés en ajoutant une étape de réentraînement.

6 Références

[1] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations", *The Journal of Machine Learning Research 18.1*, pp. 6869-6898, 2017.

[2] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M.A. Horowitz, and W.J. Dally, "EIE: efficient inference engine on compressed deep neural network", *Proceedings of the 43rd International Symposium on Computer Architecture*, IEEE Press, pp. 243–254, 2016.

[3] S. Zhang, Z. Du, L. Zhang, H. Lan, S. Liu, L. Li, Q. Guo, T. Chen, and Y. Chen. "Cambricon-X: An Accelerator for Sparse Neural Networks" *Proceedings of the International Symposium on Microarchitecture (MICRO)*, 2016.

[4] S. Han, H. Mao, and W.J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding", *4th International Conference on Learning Representations*, 2016.

[5] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network", *Deep Learning and Representation Learning Workshop*, NIPS, 2014.

[6] E. Denton, W. Zaremba, J. Bruna, Y. Lecun, and R. Fergus, "Exploiting linear structure within convolutional neural networks for efficient evaluation", *Advances in Neural Information Processing Systems*, pp. 1269-1277, 2014.

[7] O. Rippel, J. Snoek, and R. Adams, "Spectral representations for convolutional neural networks", *Advances in Neural Information Processing Systems*, pp. 2449-2457, 2015.

[8] A. Krizhevsky, and G. Hinton, "Learning multiple layers of features from tiny images", *(Technical Report) University of Toronto*, 2009.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770-778, 2016.

[10] K. Simonyan, and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", *International Conference on Learning Representations*, 2015.

[11] M. Abadi, A. Agarwal, et al., "Tensorflow: Large-scale machine learning on heterogeneous distributed systems", *12th Symposium on Operating Systems Design and Implementation.* 2016.

[12] A. Lavin, and S. Gray, "Fast algorithms for convolutional neural networks", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2016.