Compression des réseaux de neurones profonds à base de quantification uniforme et non-uniforme

Diana RESMERITA[1,2], Rodrigo CABRAL FARIAS[1], Benoit Dupont DE DINECHIN[2], Lionel FILLATRE[1]

[1]Université Côte d'Azur, CNRS, I3S
2000 route des Lucioles, Les algorithmes, 06903 Sophia Antipolis, France

[2]Kalray
180 Avenue de l'Europe, 38330 Montbonnot-Saint-Martin, France

{diana.resmerita, cabral, lionel.fillatre}@i3s.unice.fr
{diana.resmerita, benoit.dinechin}@kalray.eu

Résumé – Les réseaux de neurones convolutifs ont des millions de paramètres et sont très coûteux en calculs, ce qui limite le déploiement de ces modèles sur des systèmes embarqués. Il est donc impératif de les compresser. De nombreux travaux ont proposé des techniques de compression telles que l'élagage, la quantification et le calcul matriciel optimisé. Cet article s'intéresse aux stratégies de quantification qui s'adaptent aux spécificités de chaque couche. Il propose une étude numérique sur l'efficacité des quantifications uniforme et non-uniforme.

Abstract – Convolutional neural networks have millions of parameters and are computationally expensive. Hence, there are some limitations when deploying these models on embedded systems. It is therefore crucial to compress them. Several works have considered compression techniques such as pruning, quantization and efficient matrix multiplications. We are interested in quantification strategies that adapt to the specificities of each layer. This article proposes a numerical study on the efficiency of uniform and non-uniform quantification.

1 Introduction

Les réseaux de neurones convolutifs (CNNs) sont reconnus comme étant la référence pour la classification, la détection et la segmentation des images. Mais, les architectures des réseaux sont de plus en plus complexes et cela provoque une augmentation de la taille mémoire et de la complexité de calcul. De nos jours, les CNNs sont très demandés dans des applications embarquées telles que la détection d'objets pour les voitures autonomes et la vidéosurveillance. Les systèmes embarqués ont généralement des capacités de calcul plutôt faibles et sont limités en mémoire et en consommation d'énergie. Lorsque le CNN est utilisé pour classifier de nouvelles données, il est indispensable de réduire les coûts de communication, d'économiser l'énergie et d'assurer un temps de réponse court.

Plusieurs méthodes ont été proposées pour compresser les réseaux de neurones comme la quantification [1, 2, 3, 4, 5], l'élagage [6, 7, 5] et la construction de nouveaux réseaux plus efficaces [8, 9]. L'élagage nécessite généralement d'utiliser des bibliothèques de calculs dédiées à l'algèbre linéaire creuse. La construction de nouveaux réseaux ne permet pas de compresser un réseau existant mais plutôt de reproduire des résultats équivalents avec une architecture différente mais plus économique. Cet article s'intéresse à la quantification parce qu'elle permet de réduire la taille de réseaux préalablement entraînés sans avoir à utiliser de bibliothèques de calculs particulières. On identifie deux tendances dans le domaine de quantification : i) la quantification par faible précision, souvent utilisée dans l'industrie et ii) la quantification classique, qu'elle soit uniforme ou non-uniforme. La quantification non-uniforme est a priori plus efficace mais elle nécessite la gestion d'un dictionnaire. La quantification uniforme est plus facile à utiliser. C'est pourquoi cet article propose de comparer les deux méthodes de quantification sur un réseau de neurones déjà entraîné en utilisant une quantification scalaire appliquée à chaque couche.

L'évaluation des quantificateurs est faite en regardant la qualité de la classification et deux types de distorsions. La première distorsion concerne les poids du réseau tandis que la seconde s'intéresse à la distribution statistique en sortie du CNN. On souhaite mettre en évidence quelle méthode apporte les meilleurs résultats. Enfin et surtout, on étudie l'impact de l'erreur pour chaque couche du réseau. L'objectif de ce travail est de répondre à plusieurs questions. Comment peut-on choisir la meilleure méthode à appliquer pour réduire la complexité d'un réseau de neurones sans perte de performance ? Comment savoir si un réseau a été bien compressé ? En particulier, quels sont les critères d'évaluation appropriés ?

Ce document est organisé de la manière suivante. La section 2 introduit les réseaux de neurones profonds. La section 3 introduit les méthodes de quantification utilisées. La section 4 décrit les expérimentations numériques effectuées sur la base d'images MNIST. Enfin, la section 5 conclut l'article.

2 Réseau de neurones profonds

Un réseau de neurones pour la classification de C classes est une fonction $h : \mathbb{R}^{n_0} \to \mathrm{S}_C$ composée de neurones formels organisés en couches [10]. L'ensemble $\mathrm{S}_C = \{\mathbf{p} \in \mathbb{R}^C \mid p_i \geq 0, \sum_{i=1}^{C} p_i = 1\}$ désigne le C-simplexe. Autrement dit, étant donné un vecteur $\mathbf{x} = (x_1, ..., x_{n_0})$, un réseau de neurones génère une distribution discrète $h(\mathbf{x}) = (h_1(\mathbf{x}), \ldots, h_C(\mathbf{x}))$ tel que $h_i = h_i(\mathbf{x})$ est la probabilité que $\mathbf{x}$ appartiennent à la classe $i \in \{1, \ldots, C\}$. La structure d'un réseau est généralement divisée en 3 parties : une couche d'entrée, des couches cachées et une couche de sortie. Une couche est un ensemble de neurones n'ayant pas de connexion entre eux. On note h^k la couche k d'un réseau. Pour un réseau avec L couches cachées, h^0 est la couche d'entrée, h^{L+1} est la couche de sortie et h^k avec $1 \leq k \leq L$ représente les couches cachées. L'entrée de chaque couche est pondérée par une matrice de poids W_k à laquelle on additionne un biais $\boldsymbol{\beta}_k$. Dans chaque couche, une fonction non-linéaire σ_k, appelé fonction d'activation, est appliquée à chaque combinaison pondérée. Les différentes couches sont formalisées de la façon suivante :

$$\mathbf{z}_0 = h^0(\mathbf{x}) = \mathbf{x}, \tag{1}$$

$$\mathbf{z}_k = h^k(\mathbf{x}) = \sigma_k(W_k h^{k-1}(\mathbf{x}) + \boldsymbol{\beta}_k),\ 1 \leq k \leq L+1, \tag{2}$$

$$W_k \in \mathbb{R}^{n_k \times n_{k-1}}, \boldsymbol{\beta}_k \in \mathbb{R}^{n_k}, \tag{3}$$

avec la convention $n_{L+1} = C$. Généralement, la couche de sortie utilise une fonction d'activation différente des couches cachées. Dans le cas d'une classification avec C classes, cette fonction d'activation, appelée *softmax*, est définie par $\mathrm{softmax}(\mathbf{u}) = (e^{u_1}, \ldots, e^{u_C}) / (\sum_{i=1}^{C} e^{u_i}),\ \forall \mathbf{u} \in \mathbb{R}^C$. La sortie du réseau est noté $\mathbf{p}(\mathbf{x}) = h_\theta(\mathbf{x})$ où la notation h_θ souligne que le réseau dépend du vecteur de paramètres

$$\theta = (\theta_1, ..., \theta_{L+1}),\ \text{ avec } \theta_k = (W_k, \boldsymbol{\beta}_k), \tag{4}$$

composé de $|\theta| = \sum_{k=1}^{L+1}(n_k n_{k-1} + n_k)$ paramètres réels. Dans cet article, seuls deux types de couches sont analysés : 1) les couches entièrement connectées (FC) associées à une matrice pleine W_k et 2) les couches convolutives (CONV) associées à une matrice de Toeplitz W_k.

Les paramètres θ sont estimés dans la phase d'apprentissage sur un ensemble d'apprentissage. Cet ensemble d'apprentissage $(X, \mathbf{y})$ est composé de N échantillons $\mathbf{x}^{(i)}$ et de N vecteurs d'étiquettes $\mathbf{y}^{(i)}$ tel que $y_c^{(i)} = 1$ si $\mathbf{x}^{(i)}$ est associée la classe c (les autres composantes de $\mathbf{y}^{(i)}$ sont nulles). L'apprentissage estime θ par minimisation d'une fonction de perte, l'entropie croisée [10], qui va indirectement maximiser la justesse du réseau, appelée *accuracy* (ACC), définie par

$$ACC(h_\theta, X, \mathbf{y}) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}\left\{ s\Big(h_\theta(\mathbf{x}^{(i)})\Big) \neq s\Big(\mathbf{y}^{(i)}\Big) \right\}, \tag{5}$$

où $s(\mathbf{z})$ donne l'indice de la classe avec la plus forte probabilité dans le vecteur $\mathbf{z} \in \mathrm{S}_C$: $s(\mathbf{z}) = \arg\max_{1 \leq i \leq C} z_i$. La justesse est ensuite évaluée sur des bases de test afin de vérifier l'erreur de généralisation du réseau.

3 Compression des réseaux

3.1 Généralités sur la quantification

La quantification scalaire est peu complexe à mettre en oeuvre. Soit w une composante de θ un paramètre d'un réseau de neurones entraîné. On défini Q_b^t un quantificateur scalaire de taille $R = 2^b$ où b est le nombre de bits et $t \in \{U, NU\}$ est le type de quantification (U pour Uniforme et NU pour Non-Uniforme). Le quantificateur Q_b^t transforme une valeur continue de l'intervalle $\mathcal{D} = [\theta_{\min}, \theta_{\max}] \subset \mathbb{R}$ en une valeur discrète $\mathcal{A} = \{a_1, a_2, ..., a_R\} \subset \mathbb{R}$. Le quantificateur partitionne l'intervalle $\mathcal{D}$ en plusieurs sous-intervalles $\mathcal{D}_i = [d_i, d_{i+1}[$, pour $1 \leq i \leq R$, tels que $\hat{w} = Q_b^t(w) = a_i$ ssi $w \in [d_i, d_{i+1}[$. Le pas de quantification $\Delta_i = d_{i+1} - d_i$ correspond à la largeur de l'intervalle $\mathcal{D}_i$. La quantification introduit une erreur, appelée distorsion, définie par : $\varepsilon(w) = Q_b^t(w) - w$.

Quantification uniforme Le quantificateur uniforme est défini par un pas constant et des niveaux de quantification a_i qui représentent les points centraux des intervalles quantifiés. Pour quantifier les paramètres θ_k d'une couche k, avec b bits, on calcule $\Delta = (\theta_{\max} - \theta_{\min})/2^b$. Pour chaque valeur $w \in \theta_k$, on calcule l'index, $i_w = \lfloor w/\Delta \rfloor$. Si on connaît i_w, on peut en déduire facilement la valeur quantifiée avec $\widehat{w} = Q_b^U(w) = \Delta i_w + \Delta/2$. L'avantage de cette méthode est qu'elle ne requiert qu'une multiplication, par conséquent, elle est plus adaptée aux matériels informatiques. La quantification uniforme n'est a priori optimale que pour une distribution de données uniforme, ce qui n'est pas le cas des paramètres d'un réseau de neurones. Cependant, les résultats dans la littérature montrent que les réseaux de neurones sont robustes aux erreurs d'approximation [11], ce qui est confirmée par la section 4.

Quantification non-uniforme Un algorithme de quantification non-uniforme bien connu est l'algorithme de Lloyd [12], également appelé k-means. La méthode de Lloyd, noté Q_b^{NU}, est définie par des intervalles $\mathcal{D}_i$ de taille variables et les centroïdes de ces intervalles qui sont les niveaux de quantification a_i. Cette quantification s'adapte à la distribution des données. Cependant, sa mise en oeuvre nécessite l'utilisation d'une table de correspondances, ce qui rend cette approche moins adaptée au matériel informatique que la quantification uniforme.

3.2 Compromis débit-distorsion

L'objectif de ce travail est de proposer une stratégie de compression pour le réseau de neurones en entier. Plus précisément, il s'agit d'approcher la fonction h_θ par une fonction $h_{\hat{\theta}}$, où $\hat{\theta} = F(\theta)$ est une forme compressée de θ. On propose d'utiliser un quantificateur différent par couche de neurones. On note $Q_{b_k}^{t_k}$ le quantificateur appliqué aux paramètres de la couche h^k où $1 \leq k \leq L+1$. La couche quantifiée $\hat{h}^k$ s'écrit :

$$\hat{h}^k(\mathbf{x}) = \sigma_k(Q_{b_k}^{t_k}(W_k) h^{k-1}(\mathbf{x}) + Q_{b_k}^{t_k}(\boldsymbol{\beta}_k)). \tag{6}$$

On cherche une séquence $(Q_{b_1}^{t_1}, ..., Q_{b_{L+1}}^{t_{L+1}})$ pour quantifier le réseau entier, afin de trouver un compromis entre la taille mémoire occupée par les poids du réseau (débit) et la qualité (distorsion) de la classification. La justesse du réseau (5) doit très peu être modifiée. On s'intéresse à deux mesures de distorsion complémentaires [13] : l'erreur quadratique moyenne (EQM) et la divergence de Kullback-Leibler (KL). L'EQM entre θ et $\hat{\theta}$,

$$\mathrm{EQM}(\theta, \hat{\theta}) = \frac{1}{|\theta|} \sum_{w \in \theta} \left(w - Q_b^t(w)\right)^2, \tag{7}$$

mesure la distance entre les paramètres avant et après quantification. La divergence KL entre la sortie du réseau initial $\mathbf{p}$ et la sortie du réseau compressé $\hat{\mathbf{p}}$,

$$\mathrm{KL}(\mathbf{p}, \hat{\mathbf{p}}) = \sum_{c=1}^{C} p_c \log\left(\frac{p_c}{\hat{p}_c}\right), \tag{8}$$

mesure l'écart entre les décisions du réseau de neurones avant et après la quantification.

4 Expérimentations numériques

Cadre expérimental Pour les expériences numériques, on utilise un réseau CNN entraîné sur les données MNIST [10]. MNIST contient des images de chiffres (10 classes) de taille 28x28 pixels en noir et blanc. L'entraînement est fait sur 60×10^3 exemples et le test sur 10×10^3 exemples. La Figure 1

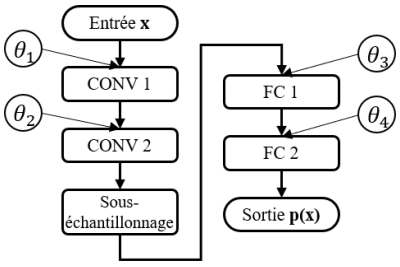

FIGURE 1 – Architecture du réseau CNN.

donne l'architecture du réseau et la Table 1 résume les informations sur les couches. La couche de sous-échantillonage n'apparaît pas dans le tableau car elle ne peut pas être quantifiée. On affiche la taille des entrées et sorties, le nombre de paramètres de chaque couche et la taille mémoire utilisée. Les variables θ_1, θ_2, θ_3, θ_4 représentent les paramètres et regroupent les poids W_k et les biais $\boldsymbol{\beta}_k$ de chaque couche tels que définis dans (4). En pratique, les valeurs de θ sont représenté en format *float32*.

TABLE 1 – Paramètres du réseau CNN entraîné sur MNIST.

Couches	Entrée	Sortie	Paramètres	Taille
CONV 1	28x28x1	26x26x32	320	1,28 KB
CONV 2	26x26x32	24x24x64	18.496	73,9 KB
FC 1	9.216	128	1.179.776	4,71 MB
FC 2	128	10	1.290	5,16 KB
Total	-	-	1.199.882	4,79 MB

Pour chaque expérience, on calcule le taux de compression

$$\tau = \frac{\text{Taille initiale}}{\text{Taille après compression}}, \tag{9}$$

la justesse (5) sur la base de test, l'EQM (7) et la divergence KL (8). La justesse initiale du réseau non-compressé est $99,16\%$.

4.1 Interprétation des résultats

Quantification couche par couche : On applique la quantification sur une seule couche isolée et les autres couches restent configurées avec les paramètres initiaux non-compressés. On s'intéresse aux résultats obtenus pour 1 à 6 bits, car de 7 à 32 bits la perte de justesse est négligeable. La Figure 2 décrit la justesse, la Figure 3 l'EQM et la Figure 4 la divergence KL.

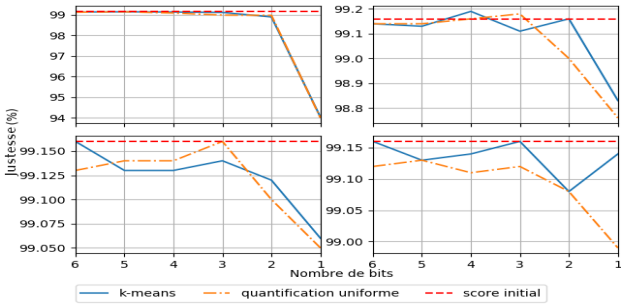

FIGURE 2 – Justesse : CONV 1 (haut gauche), CONV 2 (haut droite), FC 1 (bas gauche), FC 2 (bas droite).

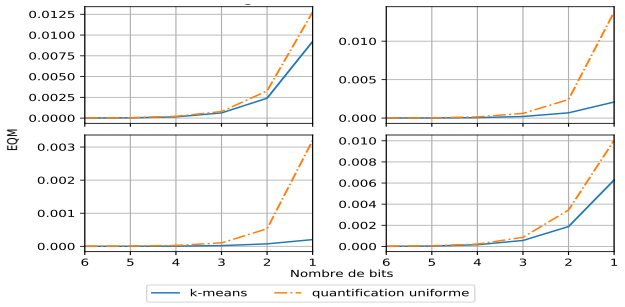

FIGURE 3 – EQM : CONV 1 (haut gauche), CONV 2 (haut droite), FC 1 (bas gauche), FC 2 (bas droite).

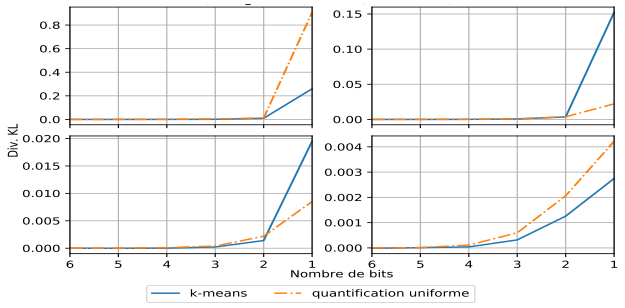

FIGURE 4 – Divergence KL : CONV 1 (haut gauche), CONV 2 (haut droite), FC 1 (bas gauche), FC 2 (bas droite).

On constate que la quantification non-uniforme fonctionne mieux car elle s'adapte globalement mieux à la distribution

des données. Il existe cependant certains cas où la quantification uniforme a de meilleures performances. Globalement, la divergence KL et l'EQM sont plus grandes pour la première couche que pour les autres. Le rôle de la première couche est de construire une bonne base de descripteurs pour tout le réseau. Elle est donc davantage sensible à la quantification.

TABLE 2 – Statistiques de compression pour chaque couche.

Couche	Méthode	ACC	Taille	τ_i	τ
CONV 1	Q_3^U	98,89 %	132B	9,69	1,0001
	Q_3^{NU}	99,11 %	150B	8,53	1,0001
CONV 2	Q_3^U	99,00 %	6,94KB	10,64	1,014
	Q_2^{NU}	99,16 %	4.64KB	15.92	1,014
FC 1	Q_1^U	99,05 %	147,48KB	31,93	21,025
	Q_1^{NU}	99,06 %	147,47KB	31,93	21,026
FC 2	Q_1^U	98,99 %	173B	29,82	1,001
	Q_1^{NU}	99,14 %	169B	30,53	1,001

La Table 2 contient des statistiques de compression pour chaque couche : la méthode utilisée, la justesse après compression, le nombre de bits de quantification, la taille de la couche après compression, le taux de compression τ_i de la couche et le taux de compression τ du réseau entier. Pour chaque couche et pour chaque méthode de compression, on indique seulement un résultat, celui qui nous semble avoir le meilleur taux de compression pour un niveau de justesse presque inchangé.

Quantification adaptative du réseau entier : La dernière évaluation est faite sur le réseau entier en appliquant un quantificateur différent (même type mais nombre de bits différent) pour chaque couche. Dans les expériences précédentes, on a observé que, pour 5 ou 6 bits, l'erreur est négligeable. Pour cette évaluation résumée dans la Table 3, on a donc appliqué une quantification entre 1 et 4 bits pour chaque couche. On a testé toutes les combinaisons possibles et on a extrait 2 cas pour chaque méthode : le premier cas donne la meilleure justesse et le second cas présente un bon compromis entre la justesse et le taux de compression. La quantification uniforme reste tout à fait performante comparée à la quantification non uniforme.

5 Conclusions et perspectives

Cet article compare deux méthodes de quantification scalaire, uniforme et non-uniforme, en vue de réduire le stockage en mémoire des paramètres. La quantification uniforme semble donner de meilleurs résultats tout en étant plus facilement utilisable d'un point de vue informatique. Quelque soit la méthode utilisée, la sortie du réseau est peu affectée par une quantification grossière des poids (2-4 bits). Des évaluations numériques sur le réseau VGG [14] ont été réalisées. Les conclusions restent globalement les mêmes. Après compression, les dernières couches du réseau ont des pertes peu significatives, contrairement aux premières couches qui sont très sensibles à l'erreur de quantification. Nos prochains travaux s'intéresseront à l'analyse théorique des phénomènes observés.

TABLE 3 – Statistiques de compression pour le réseau avec des quantificateurs adaptés à chaque couche.

Méthode	b_1 CONV 1	b_2 CONV 2	b_3 FC 1	b_4 FC 2	ACC	τ
$Q_{b_k}^U$	4	3	3	4	99,18%	10,66
	3	2	2	2	98,80%	16
$Q_{b_k}^{NU}$	4	4	4	3	99,16%	8
	4	2	2	2	98,33%	15,98

Références

[1] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. G. Howard, H. Adam, and D. Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *IEEE CVPR 2018*, pages 2704–2713, 2018.

[2] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan. Deep learning with limited numerical precision. In *Proceedings of the 32nd ICML*, volume 37, pages 1737–1746, 2015.

[3] P. Gysel. Ristretto : Hardware-oriented approximation of convolutional neural networks. *CoRR*, 2016.

[4] Y. Gong, L. Liu, M. Yang, and L. D. Bourdev. Compressing deep convolutional networks using vector quantization. *CoRR*, 2014.

[5] S. Han, H. Mao, and W. J. Dally. Deep compression : Compressing deep neural network with pruning, trained quantization and huffman coding. In Y. Bengio and Y. LeCun, editors, *4th ICLR*, 2016.

[6] S. Anwar, K. Hwang, and W. Sung. Structured pruning of deep convolutional neural networks. *JETC*, 13(3) :32 :1–32 :18, 2017.

[7] H. Mao, S. Han, J. Pool, W. Li, X. Liu, Yu Wang, and W. J. Dally. Exploring the regularity of sparse structure in convolutional neural networks. *CoRR*, 2017.

[8] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks. In *NIPS 29*, pages 4107–4115, 2016.

[9] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net : ImageNet classification using binary convolutional neural networks. In *Comput. Vis. ECCV*, pages 525–542. Springer, 2016.

[10] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[11] P. Merolla, R. Appuswamy, J. V. Arthur, S. K. Esser, and D. S. Modha. Deep neural networks are robust to weight binarization and other non-linear distortions. *CoRR*, 2016.

[12] A. Gersho and R. M. Gray. *Vector quantization and signal compression*. Springer Science & Business Media, 2012.

[13] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, New York, 2006.

[14] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd ICLR*, 2015.