Compression des réseaux de neurones convolutifs : l'apport de la décomposition CP régularisée

Rima KHOUJA[1,2], Konstantin USEVICH[2], Marianne CLAUSEL[1], Sebastian MIRON[2]

[1]Université de Lorraine, CNRS, IECL, Nancy, France

[2]Université de Lorraine, CNRS, CRAN, Nancy, France.

prenom.nom@univ-lorraine.fr

Résumé – Les réseaux de neurones convolutifs (CNNs) sont devenus un outil clé dans le domaine de la vision par ordinateur avec l'avènement d'outils de calcul efficaces, notamment les unités de traitement graphique (GPU). L'un des problèmes des CNN est leur grand nombre de paramètres. Ceci induit en effet des coûts de stockage et de calcul importants, ce qui notamment a un impact environnemental non-négligeable. Dans ce papier, nous considérons des réseaux de neurones unidimensionnels et nous proposons une technique de compression basée sur une approximation des tenseurs liés aux couches de convolution. Notre méthode étend les travaux de (Lebedev et al, ICLR 2015), qui a proposé d'utiliser la décomposition CP (CPD) pour compresser les tenseurs convolutionels. L'avantage particulier de la CPD est qu'elle décompose la convolution multidimensionnelle en opérations unidimensionnelles plus simples qui permettent de gagner en temps d'inférence. Nous proposons d'améliorer cet algorithme en introduisant un terme de régularisation. Le problème d'optimisation correspondant est ensuite résolu de manière efficace via un algorithme des moindres carrés alternés.

Abstract – Convolutional neural networks (CNNs) have become a key tool in the field of computer vision with the advent of efficient computational tools, notably graphics processing units (GPUs). One of the problems with CNNs is their large number of parameters. This in fact induces significant storage and calculation costs, which in particular has a non-negligible environmental impact. In this paper, we consider one-dimensional neural networks and we propose a compression technique based on an approximation of the tensors related to the convolution layers. Our method extends the work of (Lebedev et al, ICLR 2015), who proposed to use CP decomposition (CPD) to compress convolutional tensors. The particular advantage of CPD is that it decomposes the multidimensional convolution into simpler one-dimensional operations which save inference time. We propose to improve this algorithm by introducing a regularization term. The corresponding optimization problem is then efficiently solved via an alternating least squares algorithm.

1 Introduction

Les réseaux de neurones convolutifs (CNNs) ont été largement utilisés dans le domaine de la vision par ordinateur [1, 2] grâce aux outils de calcul efficaces tels que les unités de traitement graphique (GPU). Cependant, les couches convolutionnelles des CNNs ont un impact énergétique important sur le coût de stockage et le temps de calcul. Afin de réduire cet impact, des approches d'IA dites "frugales" se sont intéressées à la compression des réseaux de neurones, proposant des méthodes de compression sans perte de performance.

Notre travail s'inscrit dans la lignée de ceux de [3], où les auteurs ont utilisé la décomposition canonique polyadique (CPD) pour le tenseur des poids, appelé tenseur du noyau. Cette décomposition permet de représenter un tenseur d'ordre supérieur comme une combinaison linéaire de tenseurs de rang un [4]. Le tenseur du noyau, de dimension trois pour la convolution unidimensionnelle et quatre pour la convolution bidimensionnelle, peut ainsi être décomposé en opérations plus simples, ce qui est crucial.

Bien que l'idée d'approximation de rang faible pour la convolution soit présente depuis longtemps dans le traitement de l'image [5, Chapitre 24] et ait été utilisée dans l'apprentissage de dictionnaires séparables [6] ainsi que dans les architectures de type MobileNet [7], l'utilisation de la décomposition tensorielle CP pour compresser la couche de convolution dans les réseaux de neurones convolutifs (CNN) en apprentissage profond a été introduite pour la première fois dans [3]. Cependant, les approches de compression du tenseur du noyau convolutif via CPD utilisent des méthodes d'approximation tensorielle classiques qui ne tiennent pas compte de l'opération de convolution elle-même, ce qui peut conduire à des résultats contre-intuitifs sur le plan applicatif. Dans certains cas, une grande erreur d'approximation du tenseur peut être observée tandis que les performances du modèle compressé restent proches de celles du modèle initial, et vice-versa. Pour résoudre ce problème, des approches coûteuses, notamment avec l'augmentation du rang d'approximation, telles que le *fine-tuning* par *back-propagation*, ont été proposées.

Dans notre travail, nous proposons une approche différente en régularisant le problème initial d'approximation tensorielle. Cette régularisation vise à minimiser la distance entre les sorties de la couche de convolution originale et celles de la couche de convolution compressée par CPD. Nous nous concentrons sur la convolution unidimensionnelle, qui trouve des applications intéressantes dans le domaine du traitement automatique du langage naturel (NLP) [8]. L'idée de pondérer l'approximation en utilisant une métrique dépendant des données a également été explorée dans d'autres travaux récents pour réduire la complexité des réseaux de neurones [9], y com-

pris pour la compression des couches convolutives [10].

Étant donné les dimensions importantes du tenseur du noyau, mettre à jour toutes les variables en une seule fois nécessiterait de résoudre un système linéaire complexe. Pour simplifier l'implémentation, nous développons une méthode des moindres carrés alternés permettant de mettre à jour les variables séquentiellement. Nous présentons une représentation matricielle de la convolution unidimensionnelle compressée via la décomposition CP et exploitons cette représentation pour obtenir des formules explicites pour le calcul des mises à jour. Cela simplifie l'implémentation de la méthode des moindres carrés alternés et permet d'optimiser efficacement le processus.

2 Notations et préliminaires

Pour faciliter la lecture, nous utilisons respectivement des lettres minuscules et majuscules en gras pour représenter les vecteurs ($\boldsymbol{a}$) et les matrices ($\boldsymbol{A}$). Un élément i (resp. (i,j)) d'un vecteur $\boldsymbol{a}$ (resp. d'une matrice $\boldsymbol{A}$) est noté a_i (resp. $a_{i,j}$). La *i-ième* ligne (resp. colonne) dans $\boldsymbol{A}$ est notée $A_{i,:}$ (resp. $A_{:,i}$). On utilise la norme de Frobenius et on la note par $\|.\|$. Les symboles $\otimes$, $\boxtimes$ et $\odot$ designent respectivement le produit extérieur, le produit de Kronecker et le produit de Khatri-Rao.

La couche de convolution unidimensionnelle (*convolution-1D*) est un cas particulier de la convolution bidimensionnelle classique, où l'entrée n'a qu'une seule dimension spatiale (largeur ou hauteur). Supposons que la dimension spatiale de l'entrée correspond à la largeur et elle est de taille X_u ; supposons également que l'entrée est constituée de S canaux arrangés verticalement. On peut représenter cette entrée sous forme d'une matrice notée $\boldsymbol{U} \in \mathbb{R}^{S\times X_u}$. Supposons que l'opérateur de convolution-1D est constitué de T filtres contenant chacun S canaux de largeur d ($d < X_u$). On stocke les poids de cet opérateur dans un tableau tridimensionnel qu'on peut assimiler à un *tenseur* [4] (généralisation des matrices en dimension supérieure à deux). Ce tenseur des poids noté $\boldsymbol{\mathcal{K}}$ (appelé tenseur du noyau) est de dimension $S \times d \times T$. La sortie est une matrice notée $\boldsymbol{V}$ dans $\mathbb{R}^{T\times X_v}$ de T canaux (nombre des filtres dans $\boldsymbol{\mathcal{K}}$) et de largeur $X_v := X_u - d + 1$ (voir Figure 1). Un élément v_{t,x_v} de $\boldsymbol{V}$ est donné par :

$$v_{t,x_v} = \sum_{i=x_v-\delta}^{x_v+\delta} \sum_{s=1}^{S} k_{s,i-x_v+\delta,t}\, u_{s,i}, \tag{1}$$

où δ désigne la demi-largeur et est égal à $d/2$ si d est pair et $(d-1)/2$ si d est impair.

La décomposition CP exprime le tenseur sous forme d'une combinaison linéaire des tenseurs simples de rang un (voir par exemple [4]). Le rang de tenseur est défini comme le plus petit nombre de termes de rang un nécessaires dans cette décomposition. Contrairement au cas matriciel, en général, il n'existe pas de moyen direct pour déterminer le rang tensoriel permettant d'obtenir une décomposition exacte. C'est pourquoi, dans le contexte de compression du tenseur de noyau, nous cherchons à approcher la décomposition tensorielle CP avec un rang faible.

Ainsi, une approximation de rang R de notre tenseur du noyau tridimensionnel $\boldsymbol{\mathcal{K}}$ peut s'écrire sous la forme suivante :

$$\boldsymbol{\mathcal{K}} \approx \sum_{i=1}^{R} K^S_{:,i} \otimes K^X_{:,i} \otimes K^T_{:,i}, \tag{2}$$

où $\boldsymbol{K^S} \in \mathbb{R}^{S\times R}$, $\boldsymbol{K^X} \in \mathbb{R}^{d\times R}$ et $\boldsymbol{K^T} \in \mathbb{R}^{T\times R}$ sont les trois matrices de facteurs de la décomposition. Un élément du tenseur s'exprime sous la forme :

$$k_{l,m,n} = \sum_{i=1}^{R} k^S_{l,i} k^X_{m,i} k^T_{n,i}. \tag{3}$$

En utilisant (3) dans (1) on peut montrer que la convolution-1D se décompose en trois simples opérations selon

1. $u^S_{r,i} := \sum_{s=1}^{S} k^S_{s,r}\, u_{s,i}$,
2. $u^{S,X}_{r,x_v} := \sum_{i=x_v-\delta}^{x_v+\delta} k^X_{i-x_v+\delta+1,r}\, u^S_{r,i}$,
3. $v^R_{t,x_v} := \sum_{r=1}^{R} k^t_{t,r}\, u^{S,X}_{r,x_v}$.

Remarquons, à partir des trois opérations ci-dessus, que l'utilisation de la décomposition CP pour le tenseur du noyau permet de réduire à la fois le nombre de paramètres et le coût de calcul (c'est-à-dire le nombre de multiplications+additions) par pixel en sortie, passant de $S \times d \times T$ (voir (1)) à $R(S+d+T)$. Cela permet d'accélérer les opérations, en particulier lorsque R est petit.

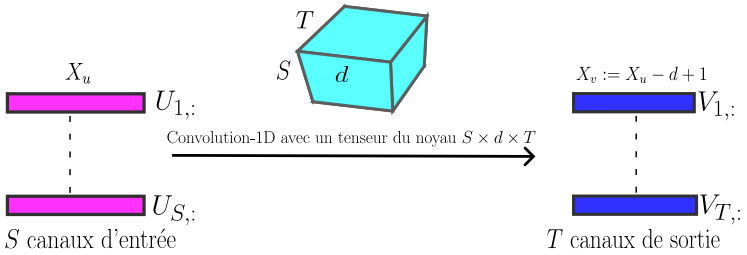


FIG. 1: La convolution-1D.

3 Représentation matricielle de la convolution-1D compressée via CPD

Il est connu que l'opération de la convolution-1D peut être représentée sous forme de multiplication matricielle à l'aide de matrices de Toeplitz. Ainsi, dans cette section, nous introduisons la représentation matricielle de l'opération de convolution-1D compressée via CPD en termes des trois matrices facteurs $\boldsymbol{K^S}$, $\boldsymbol{K^X}$ et $\boldsymbol{K^T}$ comme suit :

$$\boldsymbol{q^R} = \boldsymbol{M_1 M_2 M_3 p}, \tag{4}$$

où

- $\boldsymbol{M_1} = \boldsymbol{K^T} \boxtimes \boldsymbol{I_{X_v}}$,
- $\boldsymbol{M_2} = \mathrm{diag}(\mathcal{T}(K^X_{:,i})_{1\le i\le R})$, tel que $\mathcal{T}(K^X_{:,i})$ est la matrice de Toeplitz de taille $X_u \times X_v$ donnée par la *i-ème* ligne de $\boldsymbol{K^X}$,

- $\boldsymbol{M_3} = (\boldsymbol{K^S})^\mathsf{T} \boxtimes \boldsymbol{I_{X_u}}$.

Le vecteur $\boldsymbol{p}$ est obtenu en empilant les représentations vectorielles de chaque canal de l'entrée $\boldsymbol{U}$, les uns après les autres, comme suit :

$$\boldsymbol{p} = \begin{bmatrix} u_{1,1} & \dots & u_{1,X_u} & \dots & u_{S,1} & \dots & u_{S,X_u} \end{bmatrix}^\mathsf{T}.$$

Le vecteur $\boldsymbol{q^R}$ est la vectorisation de la sortie $\boldsymbol{V^R}$ de la couche de convolution compressée via la CPD, selon :

$$\boldsymbol{q^R} = \begin{bmatrix} v^R_{1,1} & \dots & v^R_{1,X_v} & \dots & v^R_{T,1} & \dots & v^R_{T,X_v} \end{bmatrix}^\mathsf{T}.$$

L'expression (4) montre explicitement comment l'opération de la convolution-1D compressée via la CPD se décompose en trois opérations plus simples et séparables, où chaque matrice facteur apparaît dans l'une de ces trois opérations.

4 Problème d'approximation CP régularisé pour la convolution 1D

Dans cette section, nous proposons d'ajouter un terme de régularisation au problème d'optimisation tensorielle standard. Ce terme vise à minimiser l'erreur entre la couche de convolution originale et la couche de convolution compressée. L'objectif de cette régularisation est d'obtenir une décomposition CP approchée plus stable par rapport à l'opération de convolution. Cela permet de trouver une décomposition CP qui ne réduit pas seulement l'erreur d'approximation tensorielle, mais qui réduit également l'erreur entre les sorties de la couche de convolution originale et celles de la couche compressée. Par conséquent, cela contribue à réduire la baisse de précision du modèle CNN compressé par rapport au modèle original, évitant ainsi l'étape habituelle de *fine-tuning* utilisée pour remédier à ce problème.

Soit $F_1(\boldsymbol{K^S}, \boldsymbol{K^X}, \boldsymbol{K^T}) := (\sum_{i=1}^R K^S_{i,:} \otimes K^X_{i,:} \otimes K^T_{i,:}) - \boldsymbol{\mathcal{K}}$ et $F_{2,j}(\boldsymbol{K^S}, \boldsymbol{K^X}, \boldsymbol{K^T}) := (\boldsymbol{M_1 M_2 M_3} - \boldsymbol{M})\boldsymbol{p}_j \quad \text{for} \quad j \in \{1, \dots, N\}$. Les $\boldsymbol{p}_j$ sont les vectorisations des entrées $\boldsymbol{U}_j$, *i.e.*, les points données utilisés pour ajuster notre approximation de $\boldsymbol{\mathcal{K}}$. On désigne par $\boldsymbol{M p}_j$ (resp. $\boldsymbol{M_1 M_2 M_3 p}_j$) la représentation matricielle de la convolution-1D de $\boldsymbol{U}_j$ par le tenseur du noyau $\boldsymbol{\mathcal{K}}$ (resp. par un CPD approché de $\boldsymbol{\mathcal{K}}$ définit par $(\boldsymbol{K^S}, \boldsymbol{K^X}, \boldsymbol{K^T})$ comme dans (4)). Le problème d'optimisation que nous proposons est alors le suivant :

$$\underset{\boldsymbol{K^S}, \boldsymbol{K^X}, \boldsymbol{K^T}}{\operatorname{argmin}} \ \lambda_1 f_1(\boldsymbol{K^S}, \boldsymbol{K^X}, \boldsymbol{K^T}) + \lambda_2 f_2(\boldsymbol{K^S}, \boldsymbol{K^X}, \boldsymbol{K^T}). \quad (5)$$

où :

- $f_1(\boldsymbol{K^S}, \boldsymbol{K^X}, \boldsymbol{K^T}) = \frac{1}{2}\|F_1\|^2$, cherche tout simplement à minimiser la distance entre le tenseur et sa décomposition CP de rang R.
- $f_2(\boldsymbol{K^S}, \boldsymbol{K^X}, \boldsymbol{K^T}) = \frac{1}{2}\sum_{j=1}^N \|F_{2,j}\|^2$, vise à minimiser l'erreur entre les sorties de la couche de convolution originale et celles de la couche compressée.
- $\lambda_1 = \frac{1}{\|\boldsymbol{\mathcal{K}}\|}$, $\lambda_2 = \frac{1}{\sum_{j=1}^N \|\boldsymbol{M p}_j\|}$.

Le tenseur du noyau a généralement des dimensions importantes, ce qui rend l'utilisation de méthodes telles que la méthode Quasi-Newton pour résoudre l'équation (5) coûteuse car cela nécessite la résolution d'un grand système linéaire. Afin d'éviter cela, nous choisissons d'appliquer une méthode des moindres carrés alternés, qui consiste à mettre à jour de manière séquentielle les 3 facteurs de la décomposition.

Soit $\boldsymbol{K}^{\boldsymbol{S}}_i, \boldsymbol{K}^{\boldsymbol{X}}_i, \boldsymbol{K}^{\boldsymbol{T}}_i$ (resp. $\boldsymbol{K}^{\boldsymbol{S}}_{i+1}, \boldsymbol{K}^{\boldsymbol{X}}_{i+1}, \boldsymbol{K}^{\boldsymbol{T}}_{i+1}$) les trois matrices facteurs à l'itération i (resp. $i+1$). On note $J^{F_1}_{S,i}$ (resp. $J^{F_{2,j}}_{S,i}$) le jacobien de F_1 (resp. de $F_{2,j}$) par rapport à $\boldsymbol{K}^{\boldsymbol{S}}_i$. De même, on note par $J^{F_1}_{X,i}$ (resp. $J^{F_{2,j}}_{X,i}$) le jacobien de F_1 (resp. de $F_{2,j}$) par rapport à $\boldsymbol{K}^{\boldsymbol{X}}_i$, et par $J^{F_1}_{T,i}$ (resp. $J^{F_{2,j}}_{T,i}$), le jacobien de F_1 (resp. de $F_{2,j}$) par rapport à $\boldsymbol{K}^{\boldsymbol{T}}_i$. La matrice $\boldsymbol{M_{1,i}}$ (resp. $\boldsymbol{M_{2,i}}$ et $\boldsymbol{M_{3,i}}$) désigne la matrice $\boldsymbol{M_1}$ (resp. $\boldsymbol{M_2}$ et $\boldsymbol{M_3}$) à l'itération i. Enfin on note par $\boldsymbol{b}_j$ la vectorisation de la sortie $\boldsymbol{V}_j$ de la convolution de $\boldsymbol{U}_j$ par $\boldsymbol{\mathcal{K}}$, pour $j \in \{1, \dots, N\}$.

Sans entrer dans les détails de calcul, les équations à résoudre pour trouver $\boldsymbol{K}^{\boldsymbol{S}}_{i+1}, \boldsymbol{K}^{\boldsymbol{X}}_{i+1}, \boldsymbol{K}^{\boldsymbol{T}}_{i+1}$ sont :

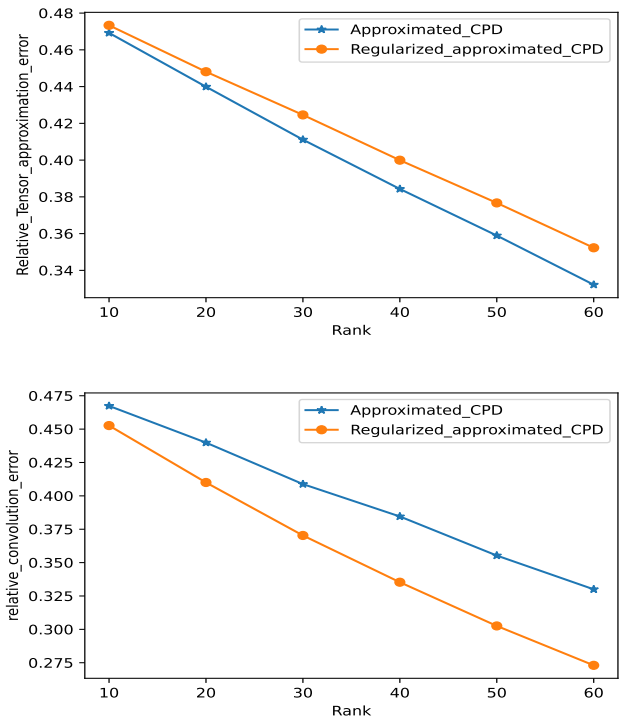
1. $[\lambda_1 (J^{F_1}_{S,i})^\mathsf{T} J^{F_1}_{S,i} + \lambda_2 \sum_{j=1}^N (J^{F_{2,j}}_{S,i})^\mathsf{T} J^{F_{2,j}}_{S,i}] \ \mathrm{vec}(\boldsymbol{K}^{\boldsymbol{S}}_{i+1}) = \lambda_1 (J^{F_1}_{S,i})^\mathsf{T} \mathrm{vec}(\boldsymbol{\mathcal{K}}_{(2)}) + \lambda_2 \sum_{j=1}^N (J^{F_{2,j}}_{S,i})^\mathsf{T} \boldsymbol{b}_j$,
où
   - $J^{F_1}_{S,i} = (\boldsymbol{K}^{\boldsymbol{T}}_i \odot \boldsymbol{K}^{\boldsymbol{X}}_{i+1}) \boxtimes \boldsymbol{I_S}$,
   - $J^{F_{2,j}}_{S,i} = \boldsymbol{M_{1,i} M_{2,i+1}} (\boldsymbol{I_R} \boxtimes \boldsymbol{U}_j^\mathsf{T})$.
2. $[\lambda_1 (J^{F_1}_{X,i})^\mathsf{T} J^{F_1}_{X,i} + \lambda_2 \sum_{j=1}^N (J^{F_{2,j}}_{X,i})^\mathsf{T} J^{F_{2,j}}_{X,i}] \ \mathrm{vec}(\boldsymbol{K}^{\boldsymbol{X}}_{i+1}) = \lambda_1 (J^{F_1}_{X,i})^\mathsf{T} \mathrm{vec}(\boldsymbol{\mathcal{K}}_{(1)}) + \lambda_2 \sum_{j=1}^N (J^{F_{2,j}}_{X,i})^\mathsf{T} \boldsymbol{b}_j$,
où
   - $J^{F_1}_{X,i} = (\boldsymbol{K}^{\boldsymbol{T}}_i \odot \boldsymbol{K}^{\boldsymbol{S}}_i) \boxtimes \boldsymbol{I_d}$
   - $J^{F_{2,j}}_{X,i} = \boldsymbol{M_{1,i}} \mathrm{diag}(\mathcal{H}(Q^{i,j}_{:,k}))_{1 \le k \le R}$,
   tel que $\boldsymbol{Q^{i,j}}$ est la matricisation du vecteur $\boldsymbol{M_{3,i}} \mathrm{vec}(\boldsymbol{U}_j)$ en une matrice de taille $X_u \times R$, et $\mathcal{H}(Q^{i,j}_{:,k})$ est la matrice de Hankel de taille $X_v \times d$ définie par la k-ième colonne de $\boldsymbol{Q^{i,j}}$.
3. $[\lambda_1 (J^{F_1}_{T,i})^\mathsf{T} J^{F_1}_{T,i} + \lambda_2 \sum_{j=1}^N (J^{F_{2,j}}_{T,i})^\mathsf{T} J^{F_{2,j}}_{T,i}] \ \mathrm{vec}(\boldsymbol{K}^{\boldsymbol{T}}_{i+1}) = \lambda_1 (J^{F_1}_{T,i})^\mathsf{T} \mathrm{vec}(\boldsymbol{\mathcal{K}}_{(3)}) + \lambda_2 \sum_{j=1}^N (J^{F_{2,j}}_{T,i})^\mathsf{T} \boldsymbol{b}_j$,
où
   - $J^{F_1}_{T,i} = (\boldsymbol{K}^{\boldsymbol{X}}_{i+1} \odot \boldsymbol{K}^{\boldsymbol{S}}_{i+1}) \boxtimes \boldsymbol{I_T}$,
   - $J^{F_{2,j}}_{T,i} = \boldsymbol{I_T} \boxtimes (\boldsymbol{M_{2,i+1} M_{3,i+1}} \mathrm{vec}(\boldsymbol{U}_j))$.

5 Exemple

Toutes les données dans cet exemple sont générées suivant une distribution normale standard. On considère une couche de convolution de $N = 36$ entrées, chacune contenant $S = 48$ canaux, chacun ayant une largeur de $X_u = 24$. Le tenseur de noyau $\boldsymbol{\mathcal{K}}$ utilisé pour cette convolution contient $T = 128$ filtres, chacun ayant S canaux de largeur $d = 9$. On compresse le tenseur $\boldsymbol{\mathcal{K}}$ en utilisant la décomposition CP calculée pour différents rangs d'approximation $R = [10, 20, 30, 40, 50, 60]$.

D'une part, nous considérons uniquement la partie f_1 de la fonction de coût f, c'est-à-dire en utilisant un algorithme d'approximation tensorielle standard. D'autre part, nous considérons f qui inclut le terme de régularisation lié à la convolution. Il convient de noter que le point initial est le même pour les deux algorithmes qui sont lancés pour les différents rangs pendant 10 itérations.
Le premier graphe montre l'erreur relative de l'approximation tensorielle par l'algorithme d'approximation standard (en bleu) et par l'algorithme d'approximation tensorielle régularisé (en orange). Le deuxième graphe montre l'erreur relative de la convolution en utilisant le tenseur approché donné par l'algorithme standard (en bleu) et celui donné par l'algorithme d'approximation tensorielle régularisée (en orange). On peut observer que le tenseur de noyau compressé en utilisant l'algorithme d'approximation régularisé conduit à une erreur relative de convolution plus faible que celle obtenue en utilisant une méthode d'approximation tensorielle classique. Cependant, on remarque une légère augmentation de l'erreur relative de l'approximation tensorielle, bien que celle-ci ne soit pas significative, pourvu que l'on puisse trouver un tenseur approché avec une erreur d'approximation tensorielle raisonnable, qui s'ajuste mieux à l'ensemble des données.

FIG. 2: Résultats des simulations pour exemple 5.



6 Conclusion

Nous proposons dans cette communication un algorithme d'approximation CPD régularisé pour compresser les couches de convolution 1D d'un réseau de neurones. L'originalité de l'approche consiste à ajouter à la fonction de coût classique un terme qui minimise l'erreur entre les sorties de la couche de convolution originale et celles de la couche compressée. Pour résoudre ce problème, nous avons développé un algorithme basé sur la méthode des moindres carrés alternés. Cet algorithme a donné des résultats prometteurs en termes de compromis entre compression du noyau et précision, tout en évitant le *fine-tuning* après la compression du modèle CNN.

References

[1] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, CA, USA, jun 2015, pp. 3431–3440, IEEE Computer Society.

[2] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun, Eds., 2015.

[3] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan V. Oseledets, and Victor S. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned cp-decomposition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun, Eds., 2015.

[4] T. Kolda and B. Bader, "Tensor decompositions and applications," *SIAM Review*, 2009.

[5] Steven W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*, California Technical Publishing, USA, 1997.

[6] Simon Hawe, Matthias Seibert, and Martin Kleinsteuber, "Separable dictionary learning," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 438–445.

[7] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman, "Speeding up convolutional neural networks with low rank expansions," *CoRR*, vol. abs/1405.3866, 2014.

[8] Yoon Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, Oct. 2014, pp. 1746–1751, Association for Computational Linguistics.

[9] Yassine Zniyed, Konstantin Usevich, Sebastian Miron, and David Brie, "Tensor-based framework for training flexible neural networks," 2021, arXiv:2106.13542.

[10] Pierre Stock, Armand Joulin, Rémi Gribonval, Benjamin Graham, and Hervé Jégou, "And the bit goes down: Revisiting the quantization of neural networks," 2020, arXiv:1907.05686.