Une approche tensorielle pour entrainer des réseaux de neurones flexibles*

Yassine ZNIYED[1], Konstantin USEVICH[2], Sebastian MIRON[2], David BRIE[2]

[1]Université de Toulon, Aix Marseille Université, CNRS, LIS, Toulon, France

[2]Université de Lorraine, CNRS, CRAN, F-54000 Nancy, France

yassine.zniyed@univ-tln.fr, {konstantin.usevich, sebastian.miron, david.brie}@univ-lorraine.fr

Résumé – Dans ce travail, nous nous concentrons sur la compression des réseaux de neurones (RNs), en considérant une approche basée sur les tenseurs. En particulier, nous nous intéressons à l'estimation de fonctions d'activation flexibles à l'aide de solutions basées sur des tenseurs. La méthode proposée est basée sur une factorisation couplée matrice-tenseur (FCMT), fusionnant les informations d'ordre 0 et d'ordre 1 des RNs. L'algorithme d'apprentissage proposé est basé sur une approche contrainte des moindres carrés alternés. Les fonctions d'activation (FAs) sont exprimées sous la forme d'une somme pondérée d'éléments de base prédéfinis. Notre méthode permet de compresser fortement des grandes couches des RNs, avec une très légère dégradation de la précision de la classification. Cela donne également un aperçu du fonctionnement des RNs, qui est un problème du plus haut intérêt dans ce domaine.

Abstract – In this work, we focus on the compression of Neural Networks (NNs), considering a tensor-based approach. In particular, we are interested in the estimation of flexible activation functions using tensor-based solutions. The proposed method is based on a coupled matrix-tensor factorization (CMTF), fusing the zeroth and first order information of the NN. The proposed learning algorithm is based on a constrained alternating least squares (ALS) approach. The activation functions (AFs) are expressed as a weighted sum of pre-defined basis elements. Our method allows to strongly compress large NN layers, with a very slight degradation of the classification accuracy. This also provides insights into the functioning of NNs, which is a problem of utmost interest in NN domain.

1 Introduction

Les réseaux de neurones (RNs) sont des outils puissants, qui ont la capacité d'apprendre et de modéliser des fonctions non linéaires complexes, tout en montrant de bonnes performances de prédiction. Avant d'entraîner un réseau de neurones, il faut décider (i) combien de neurones et de couches cachées utiliser ? et (ii) quelles fonctions d'activation (FAs) utiliser ? Ces choix contrôlent la capacité d'apprentissage du RN à partir des données. Les architectures des RNs modernes de pointe sont trop volumineuses et comportent de nombreux paramètres. Les exécuter dans des systèmes avec une capacité de calcul limitée est une tâche difficile. Pour cette raison, plusieurs travaux dans la littérature se sont intéressés à la compression de ces modèles.

Les approches tensorielles les plus populaires [1] pour la compression visent principalement à compresser les couches des réseaux convolutifs, qui peuvent être considérés comme des tenseurs. Par une décomposition polyadique canonique (CPD) [2], ils remplacent les convolutions multidimensionnelles par des convolutions unidimensionnelles. Une autre direction de recherche se concentre sur la relation entre les décompositions de tenseurs et les réseaux de neurones avec des unités de produit (au lieu d'unités de sommation) [3] ; ce type de représentations est cependant peu utilisé en pratique.

Dans ce travail, nous considérons une approche totalement différente. Tout en conservant la structure traditionnelle du réseau de neurones (poids linéaires + fonctions d'activation non linéaires), nous visons à ajouter de la flexibilité [4] aux FAs, par opposition aux FAs fixes utilisées classiquement. En particulier, les fonctions d'activation peuvent être différentes dans différents nœuds. Une telle architecture est particulièrement intéressante grâce à la théorie de l'identifiabilité (unicité) disponible dans le cas polynomial [5]. Les propriétés d'identifiabilité peuvent donner un aperçu du fonctionnement de ces RNs et aider à renforcer la stabilité de la représentation.

Contrairement aux méthodes existantes pour les FAs flexibles (FAFs) qui utilisent des techniques d'entraînement conventionnelles [4], nous utilisons un cadre original développé dans la communauté d'identification de systèmes non linéaires. Les travaux de [6] ont montré qu'une architecture avec une couche flexible cachée peut être identifiée comme une CPD d'un tenseur jacobien. Cependant, il n'est pas directement applicable dans la configuration d'apprentissage ; en particulier, il n'y a pas de moyen simple d'estimer les fonctions d'activation. Dans ce travail, nous proposons une nouvelle méthode de compression de réseaux de neurones pré-entraînés basée sur la factorisation couplée matrice-tenseur (FCMT). Cette technique fusionne les informations d'ordre 0 et 1 du RN, où les informations de premier ordre sont contenues dans un tenseur jacobien,

*Travail soutenu par l'ANR LeaFleT (ANR-19- CE23-0021).

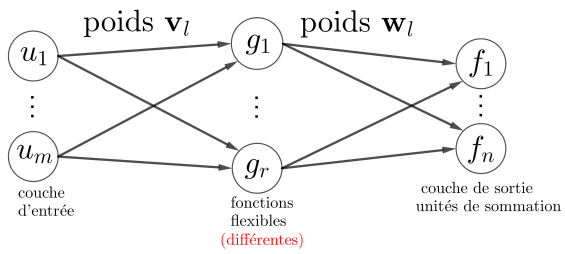

FIGURE 1 – Représentation graphique du bloc de base à une couche du RN flexible.

suivant une CPD contrainte.

Le nouveau RN flexible a des FAs différentes et flexibles dans chaque neurone. Les FAFs sont exprimées sous forme de somme pondérée de fonctions de base prédéfinies. Le but de ce modèle est d'augmenter l'expressivité du modèle pour pouvoir apprendre des fonctions avec un nombre de paramètres inférieur par rapport aux modèles classiques. Ce genre de résultat est susceptible d'accélérer l'exécution des réseaux de neurones et permettra leur utilisation sur des systèmes à faibles ressources de calcul.

2 Modèle de FAFs

Modèle flexible Le modèle proposé dans ce travail est un RN flexible, où chaque neurone a une FA différente et flexible. Les FAs sont exprimées sous la forme d'une combinaison linéaire de fonctions de base prédéfinies. Une illustration graphique d'un bloc de base à une couche de ce modèle est donnée dans la Figure 1. La fonction multivariée qui représente la relation entrée-sortie est notée $\boldsymbol{f}$. Cette fonction $\boldsymbol{f} : \mathbb{R}^m \to \mathbb{R}^n$ est donnée par :

$$\boldsymbol{f}(\boldsymbol{u}) = [f_1(\boldsymbol{u}) \cdots f_n(\boldsymbol{u})]^\top, \quad \text{with } \boldsymbol{u} = [u_1 \cdots u_m]^\top. \quad (1)$$

On dit que $\boldsymbol{f}$ admet un modèle RN flexible s'il peut s'exprimer sous la forme :

$$\boldsymbol{f}(\boldsymbol{u}) = \boldsymbol{W}\boldsymbol{g}(\boldsymbol{V}^\top \boldsymbol{u}), \quad (2)$$

où $\boldsymbol{V} \in \mathbb{R}^{m\times r}$, $\boldsymbol{W} \in \mathbb{R}^{n\times r}$ sont des matrices de transformation, $\boldsymbol{v}_\ell$ et $\boldsymbol{w}_\ell$, pour $1 \leq \ell \leq r$, sont respectivement les colonnes de $\boldsymbol{V}$ et $\boldsymbol{W}$, et $\boldsymbol{g} : \mathbb{R}^r \to \mathbb{R}^r$ est exprimée comme

$$\boldsymbol{g}(t_1, \ldots, t_r) = [g_1(t_1) \cdots g_r(t_r)]^\top, \quad (3)$$

avec $g_\ell : \mathbb{R} \to \mathbb{R}$ est une fonction non linéaire univariée.

Fonctions de base Les fonctions non linéaires univariées g_ℓ ($1 \leq \ell \leq r$) dans (2) sont une combinaison linéaire de fonctions de base prédéfinies $\{\phi_1, \ldots, \phi_d\}$:

$$g_\ell(t) = c_{0,\ell} + c_{1,\ell}\phi_1(t) + \cdots + c_{d,\ell}\phi_d(t). \quad (4)$$

Les fonctions ϕ_k, pour $1 \leq k \leq d$, dans (4) étant *a priori* choisies, les paramètres $c_{k,\ell}$, pour $0 \leq k \leq d$ et $1 \leq \ell \leq r$, sont les paramètres apprenables.

3 Décomposition d'une fonction multivariée basée sur un tenseur

Représenter une fonction non-linéaire $\boldsymbol{f}$ par une représentation plus simple, telle que (2) a été abordée dans plusieurs travaux, puisqu'elle permet d'approximer la fonction avec un nombre réduit de paramètres et peut donner un aperçu de la compréhension des mécanismes derrière la fonction non linéaire. Ces méthodes supposent que la fonction $\boldsymbol{f}$ est connue à travers une représentation paramétrique polynomiale, et ont été développées dans le contexte du problème de découplage pour les polynômes réels multivariés. L'une de ces méthodes est la méthode jacobienne [6] qui vise à décomposer $\boldsymbol{f}(\boldsymbol{u})$ selon (2) à partir de son information de premier ordre. Cette approche repose sur l'observation que la matrice jacobienne de $\boldsymbol{f}$, évaluée en un point donné $\boldsymbol{u}$ s'exprime par

$$\boldsymbol{J}_{\boldsymbol{f}}(\boldsymbol{u}) := \begin{bmatrix} \frac{\partial f_1}{\partial u_1}(\boldsymbol{u}) & \cdots & \frac{\partial f_1}{\partial u_m}(\boldsymbol{u}) \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial u_1}(\boldsymbol{u}) & \cdots & \frac{\partial f_n}{\partial u_m}(\boldsymbol{u}) \end{bmatrix} \quad (5)$$

$$= \boldsymbol{W} \cdot diag\big(g_1'(\boldsymbol{v}_1^\top \boldsymbol{u}) \cdots g_r'(\boldsymbol{v}_r^\top \boldsymbol{u})\big) \cdot \boldsymbol{V}^\top. \quad (6)$$

En empilant les matrices jacobiennes évaluées en N points d'échantillonnage différents $\boldsymbol{u}^{(j)} \in \mathbb{R}^m$, pour $1 \leq j \leq N$, tel que $\boldsymbol{\mathcal{J}}_{:,:,j} = \boldsymbol{J}_{\boldsymbol{f}}(\boldsymbol{u}^{(j)})$, nous construisons un tenseur jacobien d'ordre 3 $\boldsymbol{\mathcal{J}}$ de taille $n \times m \times N$ suivant une CPD exprimée comme

$$\boldsymbol{\mathcal{J}} = \sum_{\ell=1}^{r} \boldsymbol{w}_\ell \otimes \boldsymbol{v}_\ell \otimes \boldsymbol{h}_\ell, \quad (7)$$

où $\otimes$ dénote le produit tensoriel. De manière équivalente, on note $\boldsymbol{\mathcal{J}} = [|\boldsymbol{W}, \boldsymbol{V}, \boldsymbol{H}|]$. Comme montré dans [6, 7], les colonnes de $\boldsymbol{H} \in \mathbb{R}^{N\times r}$ ont la structure suivante

$$\boldsymbol{h}_\ell = [g_\ell'(\boldsymbol{v}_\ell^\top \boldsymbol{u}^{(1)}), \cdots, g_\ell'(\boldsymbol{v}_\ell^\top \boldsymbol{u}^{(N)})]^\top, \quad (8)$$

ce qui montre que le facteur matriciel $\boldsymbol{H}$ dépend de $\boldsymbol{V}$. Il convient de noter que l'approche originale de [6] est assez sensible à la non-unicité de la CPD et au bruit [8, 9], puisqu'une solution ALS relaxée est utilisée pour estimer les facteurs.

Suivant [8, 9], cela peut être attribué au fait que l'algorithme ne tient pas compte de la structure particulière du tenseur jacobien (8), *ie* la relation entre les colonnes de $\boldsymbol{H}$ et $\boldsymbol{V}$. Cela devient problématique, en particulier lorsque l'unicité de la CPD est perdue, auquel cas l'ALS relâché ne parviendra pas à récupérer des FAs significatives. Cela a motivé le développement d'algorithmes structurés pour le découplage de polynômes multivariés [8, 9]. De plus, il a été montré dans [7] que la prise en compte de la structure (8) conduit à des conditions d'unicité moins restrictives par rapport au cas non structuré, élargissant ainsi le domaine d'application de la méthode.

4 Problème d'apprentissage

Pour surmonter les inconvénients susmentionnés des méthodes existantes, nous proposons dans cette section un algorithme

pour la CPD structurée du tenseur jacobien. Il consiste à formuler la CPD structurée comme une factorisation couplée matrice-tenseur contrainte, utilisant conjointement les informations d'ordre 0 et 1.

Information d'ordre 0 Soit $\boldsymbol{F}$ une matrice de taille $n \times N$ dont les lignes sont les évaluations des n sorties du sous-réseau d'origine à N points d'échantillonnage différents $\boldsymbol{u}^{(j)} \in \mathbb{R}^m$:

$$\boldsymbol{F} = \begin{bmatrix} f_1(\boldsymbol{u}^{(1)}) & \cdots & f_1(\boldsymbol{u}^{(N)}) \\ \vdots & & \vdots \\ f_n(\boldsymbol{u}^{(1)}) & \cdots & f_n(\boldsymbol{u}^{(N)}) \end{bmatrix}. \tag{9}$$

En se basant sur (9) et (2), nous pouvons voir que $\boldsymbol{F}$ est sujet à une factorisation donnée par

$$\boldsymbol{F} = \begin{bmatrix} f_1(\boldsymbol{u}^{(1)}) & \cdots & f_1(\boldsymbol{u}^{(N)}) \\ \vdots & & \vdots \\ f_n(\boldsymbol{u}^{(1)}) & \cdots & f_n(\boldsymbol{u}^{(N)}) \end{bmatrix} \tag{10}$$

$$= \boldsymbol{W} \cdot \underbrace{\begin{bmatrix} g_1(\boldsymbol{v}_1^\top \boldsymbol{u}^{(1)}) & \cdots & g_1(\boldsymbol{v}_r^\top \boldsymbol{u}^{(N)}) \\ \vdots & & \vdots \\ g_r(\boldsymbol{v}_1^\top \boldsymbol{u}^{(1)}) & \cdots & g_r(\boldsymbol{v}_r^\top \boldsymbol{u}^{(N)}) \end{bmatrix}}_{\boldsymbol{Z}^\top}, \tag{11}$$

où $\boldsymbol{Z}$ est de taille $N \times r$. On peut voir à partir de (11) que la matrice $\boldsymbol{F}$ partage le même facteur $\boldsymbol{W}$ avec le tenseur $\boldsymbol{\mathcal{J}}$. Sur la base de cette observation, nous construisons un problème FCMT contraint, impliquant la matrice $\boldsymbol{F}$ et le tenseur $\boldsymbol{\mathcal{J}}$.

Algorithme d'apprentissage L'idée de l'approche basée sur la FCMT contrainte est de formuler l'apprentissage des paramètres, $\mathbf{W}$, $\mathbf{V}$ et $c_{k,\ell}$, pour $0 \leq k \leq d$ et $1 \leq \ell \leq r$, comme suit.

$$\min_{\mathbf{W},\mathbf{V},\mathbf{H},\mathbf{Z}} \left\| \boldsymbol{\mathcal{J}} - [|\mathbf{W}, \mathbf{V}, \mathbf{H}|] \right\|^2 + \lambda \cdot \left\| \mathbf{F} - \mathbf{W}\mathbf{Z}^T \right\|^2 \tag{12}$$

$$\text{t.q.} \quad h_{j,\ell} = c_{1,\ell}\phi_1'(\mathbf{v}_\ell^T \mathbf{u}^{(j)}) + \cdots + c_{d,\ell}\phi_d'(\mathbf{v}_\ell^T \mathbf{u}^{(j)}),$$

$$z_{j,l} = c_{0,l} + c_{1,\ell}\phi_1(\mathbf{v}_\ell^T \mathbf{u}^{(j)}) + \cdots + c_{d,\ell}\phi_d(\mathbf{v}_\ell^T \mathbf{u}^{(j)}).$$

Ces contraintes s'expriment de manière équivalente comme

$$\boldsymbol{h}_\ell = \boldsymbol{X}_\ell \cdot \boldsymbol{c}_\ell, \quad \boldsymbol{z}_\ell = \boldsymbol{Y}_\ell \cdot \boldsymbol{c}_\ell, \quad \text{for } 1 \leq \ell \leq r,$$

où les matrices $\boldsymbol{X}_\ell$ et $\boldsymbol{Y}_\ell$, de taille $N \times (d+1)$, sont données par

$$\boldsymbol{X}_\ell = \begin{bmatrix} 0 & \phi_1'(\boldsymbol{v}_\ell^\top \boldsymbol{u}^{(1)}) & \cdots & \phi_d'(\boldsymbol{v}_\ell^\top \boldsymbol{u}^{(1)}) \\ \vdots & \vdots & & \vdots \\ 0 & \phi_1'(\boldsymbol{v}_\ell^\top \boldsymbol{u}^{(N)}) & \cdots & \phi_d'(\boldsymbol{v}_\ell^\top \boldsymbol{u}^{(N)}) \end{bmatrix}, \tag{13}$$

$$\boldsymbol{Y}_\ell = \begin{bmatrix} 1 & \phi_1(\boldsymbol{v}_\ell^\top \boldsymbol{u}^{(1)}) & \cdots & \phi_d(\boldsymbol{v}_\ell^\top \boldsymbol{u}^{(1)}) \\ \vdots & \vdots & & \vdots \\ 1 & \phi_1(\boldsymbol{v}_\ell^\top \boldsymbol{u}^{(N)}) & \cdots & \phi_d(\boldsymbol{v}_\ell^\top \boldsymbol{u}^{(N)}) \end{bmatrix}, \tag{14}$$

et le ℓ-ième vecteur de coefficient $\boldsymbol{c}_\ell$ est défini comme $\boldsymbol{c}_\ell = \begin{bmatrix} c_{0,\ell} & c_{1,\ell} & \ldots & c_{d,\ell} \end{bmatrix}^\top$.

Pour résoudre le problème (12), nous proposons un algorithme des moindres carrés alternés à facteurs $\boldsymbol{H}$ et $\boldsymbol{Z}$ contraints, dans lequel nous mettons à jour les coefficients des FAs $g_1, \cdots, g_r$ selon (4). La procédure complète est donnée dans Algorithme 1, où $\odot$ dénote le produit de Khatri-Rao.

Algorithme 1 Algorithme d'apprentissage basé sur la FCMT

Entrée : Tenseur $\boldsymbol{\mathcal{J}}$ de dimenson $n \times m \times N$, matrice $\boldsymbol{F}$ de taille $n \times N$, les fonctions $\{\phi_1, \ldots, \phi_d\}$, paramètre d'ajustement λ, rang r.
Sortie : Facteurs $\boldsymbol{W}$, $\boldsymbol{V}$, $\boldsymbol{H}$ et $\boldsymbol{Z}$, et les coefficients $\boldsymbol{c}_\ell$.

Initialiser $\boldsymbol{V}$, $\boldsymbol{H}$, $\boldsymbol{Z}$
répéter
Mettre à jour $\boldsymbol{W}$ avec $\min_{\boldsymbol{W}} \left\| \text{unfold}_1 \boldsymbol{\mathcal{J}} - \boldsymbol{W}\left(\boldsymbol{H} \odot \boldsymbol{V}\right)^\top \right\|^2 + \lambda \cdot \left\| \boldsymbol{F} - \boldsymbol{W}\boldsymbol{Z}^\top \right\|^2$
Mettre à jour $\boldsymbol{V}$ avec $\min_{\boldsymbol{V}} \left\| \text{unfold}_2 \boldsymbol{\mathcal{J}} - \boldsymbol{V}\left(\boldsymbol{H} \odot \boldsymbol{W}\right)^\top \right\|^2$
Mettre à jour $\boldsymbol{H}$ avec $\min_{\boldsymbol{H}} \left\| \text{unfold}_3 \boldsymbol{\mathcal{J}} - \boldsymbol{H}\left(\boldsymbol{V} \odot \boldsymbol{W}\right)^\top \right\|^2$
Mettre à jour $\boldsymbol{Z}$ avec $\min_{\boldsymbol{Z}} \left\| \boldsymbol{F} - \boldsymbol{W}\boldsymbol{Z}^\top \right\|^2$
pour $\ell = 1 \ldots r$ **faire**
  Construire $\boldsymbol{X}_\ell$ selon (13).
  Construire $\boldsymbol{Y}_\ell$ selon (14).
fin pour
Calculer $\boldsymbol{c}$ comme $\min_{\boldsymbol{c}} \left\| \text{vec}(\boldsymbol{H}) - \begin{bmatrix} \boldsymbol{X}_1 & & 0 \\ & \ddots & \\ 0 & & \boldsymbol{X}_r \end{bmatrix} \cdot \boldsymbol{c} \right\|^2 + \lambda \cdot \left\| \text{vec}(\boldsymbol{Z}) - \begin{bmatrix} \boldsymbol{Y}_1 & & 0 \\ & \ddots & \\ 0 & & \boldsymbol{Y}_r \end{bmatrix} \cdot \boldsymbol{c} \right\|^2$
pour $\ell = 1 \ldots r$ **faire**
  Calculer $\boldsymbol{h}_\ell$ t.q : $\boldsymbol{h}_\ell = \boldsymbol{X}_\ell \cdot \boldsymbol{c}_\ell$
  Calculer $\boldsymbol{z}_\ell$ t.q : $\boldsymbol{z}_\ell = \boldsymbol{Y}_\ell \cdot \boldsymbol{c}_\ell$
fin pour
jusqu'à un test de convergence est satisfait ou le nombre maximal d'itérations est atteint
retourne $\boldsymbol{W}$, $\boldsymbol{V}$, $\boldsymbol{c}_\ell$.

5 Compression des réseaux de neurones pré-entraînés

Pour illustrer l'efficacité de l'apprentissage de réseaux de neurones flexibles basé sur des tenseurs, la compression de réseaux de neurones pré-formés est considérée dans cette section. L'un des principaux objectifs de la compression est de réduire la complexité d'un réseau de neurones afin de permettre son implémentation sur des systèmes embarqués avec des capaci-

tés de calcul et de mémoire limitées.

Processus de compression L'idée ici est de remplacer une ou plusieurs couches consécutives du réseau d'origine par une seule couche flexible. Ce processus est représenté graphiquement dans la Fig. 2. À gauche, la figure correspond à un RN avec deux couches cachées et des FAs fixes (ReLU dans cet exemple). Les deux couches cachées sont remplacées par une seule couche avec des FAs flexibles (sur le côté droit de la Fig. 2) ayant un nombre de neurones inférieur au réseau initial, *c'est-à-dire* le réseau compressé.

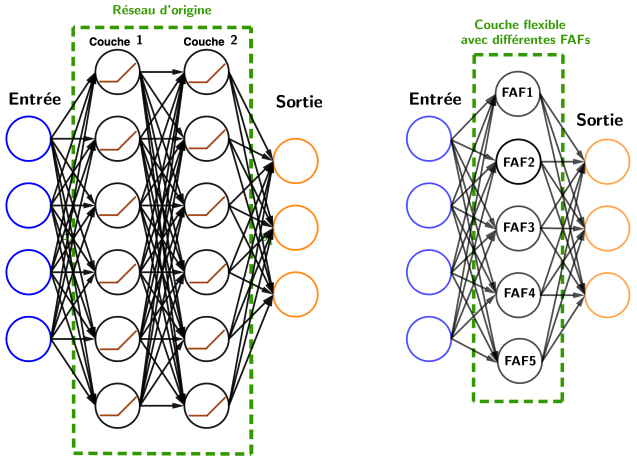

FIGURE 2 – Représentation graphique du processus de compression : (à gauche) le RN d'origine avec des FAs fixes - (à droite) le RN approximé avec des FAs flexibles.

Quelques résultats de simulation Dans ce travail, nous considérons un RN convolutif [10], qui comprend quatre couches convolutives utilisant une FA maxout. Le réseau a été formé pour classer images 24×24 de caractères en niveaux de gris en 36 classes différentes, correspondant à 10 chiffres et 26 lettres. Les EQMs normalisées sont définies selon EQMN tenseur $= \frac{||\hat{\mathcal{J}}-\mathcal{J}||^2}{||\mathcal{J}||^2}$, et, EQMN matrice $= \frac{||\hat{\boldsymbol{F}}-\boldsymbol{F}||^2}{||\boldsymbol{F}||^2}$, où $\hat{\mathcal{J}}$ and $\hat{\boldsymbol{F}}$ font respectivement référence au tenseur et à la matrice estimés. La `Perte de précision` est définie comme la différence entre la précision de classification du RN original et du RN approximé. Le `Taux de compression` est égal au rapport entre le nombre de paramètres des couches compressées et d'origine. Nous notons que DTC fait référence à une approche de décomposition tensorielle contrainte réalisée avec une correction de décalage.

6 Conclusion

Dans ce travail, nous avons proposé un cadre basé sur des tenseurs pour la compression de grands réseaux de neurones, en utilisant des fonctions d'activation flexibles. Les FAs sont

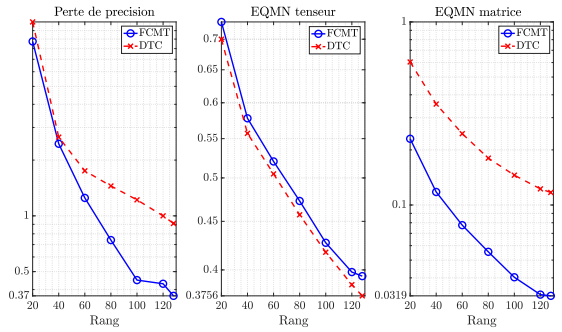

FIGURE 3 – FCMT vs DTC, avec $N = 360$, $d = 4$.

exprimées sous la forme d'une somme pondérée de fonctions de base prédéfinies. En formulant le problème comme un problème de factorisation couplée matrice-tenseur contraint, nous proposons un nouvel algorithme d'apprentissage basé sur ALS, qui peut gérer différentes bases de décomposition. Le but de cette solution est de compresser de grands RNs, en remplaçant une ou plusieurs couches du réseau initial par une couche ayant des FAs flexibles. L'apprentissage des nouvelles fonctions est effectué sur la base d'une factorisation couplée d'un tenseur jacobien, suivant une décomposition canonique polyadique contrainte, et d'une matrice d'évaluation.

Références

[1] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned CP-decomposition," *CoRR*, vol. abs/1412.6553, 2015.

[2] F. L. Hitchcock, "Multiple invariants and generalized rank of a p-way matrix or tensor," *Journal of Mathematics and Physics*, vol. 7, pp. 39–79, 1927.

[3] N. Cohen, O. Sharir, and A. Shashua, "On the expressive power of deep learning : A tensor analysis," in *29th Annual Conference on Learning Theory*, 2016, pp. 698–728.

[4] A. Apicella, F. Donnarumma, F. Isgrò, and R. Prevete, "A survey on modern trainable activation functions," *Neural Networks*, vol. 138, pp. 14–32, 2021.

[5] P. Comon, Y. Qi, and K. Usevich, "Identifiability of an X-rank decomposition of polynomial maps," *SIAM Journal on Applied Algebra and Geometry*, vol. 1, no. 1, pp. 388–414, 2017. [Online]. Available : https://doi.org/10.1137/16M1108388

[6] P. Dreesen, M. Ishteva, and J. Schoukens, "Decoupling multivariate polynomials using first-order information and tensor decompositions," *SIAM Journal on Matrix Analysis and Applications*, vol. 36, no. 12, pp. 864–879, 2015.

[7] K. Usevich, P. Dreesen, and M. Ishteva, "Decoupling multivariate polynomials : Interconnections between tensorizations," *Journal of Computational and Applied Mathematics*, vol. 363, pp. 22–34, 2020.

[8] G. Hollander, "Multivariate polynomial decoupling in nonlinear system identification," Ph.D. dissertation, Vrije Universiteit Brussel, 2018.

[9] Y. Zniyed, K. Usevich, S. Miron, and D. Brie, "Learning nonlinearities in the decoupling problem with structured CPD," in *16th IFAC Symposium on System Identification, Padova, Italy*, 2021.

[10] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Deep features for text spotting," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham : Springer International Publishing, 2014, pp. 512–528.