Extraction de texture par décomposition de rangs

Vincent VIGNERON[1], Hichem MAAREF[1], Jean-Philippe CONGÉ[1]

[1]Univ Evry, Université Paris-Saclay IBISC EA 4526, Evry, France.
{vincent.vigneron,hichem.maaref,jean-philippe.conge}@univ-evry.fr

Résumé – La couche de pooling est au cœur des convolutional neural networks (CNNs) contribuant à l'invariance aux variations et au bruit des données d'apprentissage. Il décrit quelle partie de l'image d'entrée un neurone de la couche de sortie peut voir. Les CNNs avec max-pooling sont plus que capables de gérer des transformations simples comme des rotations sans trop de problèmes. Le problème vient des transformations compliquées. Le classement par ordre d'importance est utilisée ici comme alternative au max-pooling. Le descripteur de texture par ordre est non paramétrique, indépendant de la disposition géométrique ou de la taille des régions d'image, et peut donc mieux tolérer les rotations. Ces fonctions de description produisent des images capables d'accentuer les basses/hautes fréquences, les contours, etc.

Abstract – The pooling layer is at the core of every CNN contributing to data invariance, variation, and perturbation. It describes which part of the input image a neuron in the output layer can see. CNNs with max pooling are more than capable of handling simple transformations like flips or rotation without too much trouble. The problem comes with complicated transforms. The rank order importance is used here as an alternative to max-pooling. The rank texture descriptor is non-parametric, independent of geometric layout or size of image regions, and can therefore better tolerate rotations. These description functions produce images capable of emphasizing low/high frequencies, contours, etc.

1 Introduction

Les structures (pyramidales) des CNNs viennent de l'idée que le réseau a besoin de voir différents niveaux de détail (résolutions) pour produire de bons résultats. Placée entre deux couches convolutives, la couche de "pooling" reçoit plusieurs cartes de caractéristiques en entrée. Le pooling (i) réduit le nombre de paramètres dans le modèle (sous-échantillonnage) et les calculs dans le réseau tout en préservant leurs caractéristiques importantes (ii) améliore l'efficacité du réseau (iii) évite le sur-apprentissage, rendant ainsi le réseau moins sensible à la position des traits : un trait un peu plus haut ou plus bas, voire ayant une orientation légèrement différente ne devrait pas provoquer de changement radical dans la classification de l'image.

La question est de savoir comment (de manière optimale) prendre en compte les caractéristiques des régions (image d'entrée) regroupées dans l'opération de pooling ? Cette question centrale dans la conception des architectures de deep learning (DL) est partiellement répondue par Lazebnik qui a démontré l'importance de la structure spatiale des voisinages de pooling [1] : en effet, les variations spatiales locales des intensités des pixels de l'image (textures) caractérisent un "phénomène de zone organisée" [2] qui ne peut pas être capturé dans les couches de pooling habituelles.

La plupart des descripteurs d'image qui encodent des structures locales e.g. local binary patterns (LBP) (et ses variantes) [3] dépendent de (i) la taille du voisinage (ii) l'ordre de lecture des voisins (iii) la fonction mathématique utilisée pour calculer la distance caractéristique entre les pixels voisins. La nouvelle valeur d'un pixel encodé $L_{P,R}$ dans une image est un entier entre 0 et 255 (pour un encodage 8 bits) calculée par :

$$L_R(P) = \sum_{p=0}^{P-1} 2^p \cdot t(g_p - g_c), \text{avec } t(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{sinon} \end{cases}, \quad (1)$$

où P compte le nombre de pixels dans le voisinage du pixel central en considérant la distance R entre le pixel central g_c et les pixels voisins $\{g_p | p = 0, \ldots, P-1\}$. Dans l'Eq. (1), $L_R(P)$ recalcule la valeur de pixel sur $8-bit$ pour un voisinage 3×3 à partir des différences entre le pixel central et ses voisins, $(g_i - g_c)$ (voir Fig. 1).

L'invariance à toute transformation monotone d'échelle de gris est obtenue avec des descripteurs de texture de type LBP en considérant dans (1) les signes des différences $t(g_i - g_c), i = 0, \ldots, P-1$. Les données binaires produites par $t(g_i - g_c)$ sont sensibles au bruit principalement dans des régions uniformes.

Cet article propose une alternative à l'opérateur habituel de pooling, capable de capturer les caractéristiques de texture, basée sur des statistiques de rang, indépendante de la disposition géométrique ou des tailles des régions de l'image, et donc capables de mieux tolérer les rotations. Elle est basée sur la définition de Savage du ranking order [4] et simple à implémenter (cf. section 2).

Notations

Les vecteurs sont en caractères gras $\mathbf{a} \in \mathbb{R}^n$, les matrices avec des lettres majuscules $A \in \mathbb{R}^{n \times m}$. Le produit scalaire entre deux vecteurs est noté $< \mathbf{a}, \mathbf{b} >$ et $\|\mathbf{a}\| = \sqrt{< \mathbf{a}, \mathbf{a} >}$ est la norme ℓ_2 d'un vecteur. Dans la suite $X_1, \ldots, X_n$ sont des variables non-ordonnées, $x_1, \ldots, x_n$ des observations non

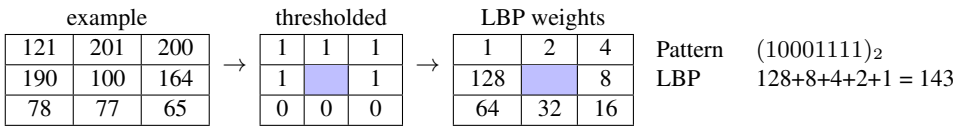


FIGURE 1 – Exemple de texturation d'une voisinage 3×3 ($P = 8$ et $R = 1$).

ordonnées. $X_{(1)} \leq \ldots \leq X_{(n)}$ et $x_{(1)} \leq \ldots \leq x_{(n)}$ sont resp. des variables et des observations ordonnées. Les statistiques d'ordre extrême sont $x_{(1)} = \min\{x_1, \ldots, x_n\}$, $x_{(n)} = \max\{x_1, \ldots, x_n\}$. La plage d'échantillonnage est $x_{(n)} - x_{(1)}$. Les X_i (non ordonnés) sont supposés être indépendants identiquement distribués (iid). Les $X_{(i)}$ sont nécessairement dépendants en raison des relations d'inégalité entre eux.

2 Fusion de rangs

Commençons par un peu de théorie sur les statistiques d'ordre. Notons $A = \{\mathsf{a}_1, \mathsf{a}_2, \ldots, \mathsf{a}_n\}$ un ensemble d'alternatives, candidats, etc. de cardinalité $|A| = n$ et V un ensemble de votants, avec $|V| = m$. Chaque électeur/juge k propose un ordre ou un classement $\boldsymbol{r}^{(k)}$ des alternatives $\mathsf{a}_1, \mathsf{a}_2, \ldots, \mathsf{a}_n$ représenté par un vecteur d'entiers $\boldsymbol{r}^{(k)} = (r_1^{(k)}, r_2^{(k)}, \ldots, r_n^{(k)})^T$, où $r_i^{(k)}$ est le rang de la $i-$ième alternative.

Les données sont collectées dans une table $(n \times m)$ $R = \{r_i^{(k)}\}$ (Fig. 2.a) qui représente le classement des n candidats attribués par les m électeurs sous la forme (a) d'un ordre total, i.e. $r_i^{(k)} \neq r_{i'}^{(k)}$, $\forall i' \neq i$ [5], (b) ou tel qu'un électeur puisse attribuer des positions *ex-aequo*.

Pour faciliter l'écriture, dans ce qui suit, $r_{ik} = r_i^{(k)}$.

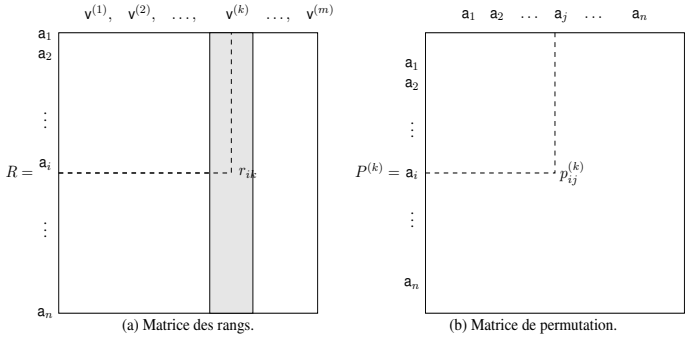


FIGURE 2 – Matrices R, $P^{(k)}$ mesurant le "désaccord" de rangs.

Le problème d'agrégation de rang consiste à trouver un ordre optimal $\boldsymbol{r}^*$ donné par un juge "virtuel" minimisant le désaccord des opinions des m juges, i.e.

$$\boldsymbol{r}^* = \arg\min_{\boldsymbol{r}} \sum_{k=1}^{m} \mathrm{d}(\boldsymbol{r}, \boldsymbol{r}^{(k)}), \quad \text{s.t.} \quad \boldsymbol{r} \in \mathcal{S}_n, \tag{2}$$

où $\mathcal{S}_n$ est le groupe symétrique des permutations $n!$ et la métrique $\mathrm{d} : \mathcal{S}_n \times \mathcal{S}_n \to \mathbb{R}^+$ est une fonction de distance choisie *a priori*.

Éq. (2) définit un programme non-linéaire dont la solution $\boldsymbol{r}^*$ est un consensus des rangs attribués par les m électeurs.

La distance entre les rangs attribués par les votants k et k' $\mathrm{d}(\boldsymbol{r}^{(k)}, \boldsymbol{r}^{(k')})$ peut être choisie, par exemple, comme la distance de l'écart absolu de rang $\sum_{i=1}^{n} |r_{ik} - r_{ik'}|$, la distance euclidienne entre les rangs $\sum_{i=1}^{n} (r_{ik} - r_{ik'})^2$, la distance de Condorcet $\sum_i \sum_j |p_{ij}^{(k)} - p_{ij}^{(k')}|$, etc. Le choix de la métrique est motivé par une gamme de propriétés rappelée dans [6].

On notera que $p_{ij}^{(k)} = \mathbb{1}_{i<j}$ désigne l'indicatrice pour laquelle $p_{ij}^{(k)} = 1$ si le rang de a_i est *inférieur* à celui de a_j et 0 sinon [7]. La permutation $\boldsymbol{r}$ peut ainsi être représentée pour un électeur k par une matrice de permutation $P^{(k)} = \{p_{ij}^{(k)}\}_{i,j=1}^{n}, x_{ij}^{(k)} \in \{0, 1\}$, avec $p_{ij}^{(k)} = 1$ si le candidat i est positionné à la place j pour le $k-$ième électeur (Fig. 2.b). On peut donc réécrire la contrainte $\boldsymbol{r}^* \in S_n$ comme

$$\sum_{j=1}^{n} x_{ij} = \sum_{i=1}^{n} x_{ij} = 1, \forall i, j \tag{3}$$

Le remplacement de $\boldsymbol{r}$ par la matrice P conduit à une fonction d'optimisation linéaire dans p_{ij} qui peut être définie comme suit :

$$\min_{\boldsymbol{r}} \sum_{ij} \sum_{k} c_{ij}^{(k)} x_{ij}, \quad \text{tel que} \quad \sum_{j=1}^{n} p_{ij} = \sum_{i=1}^{n} p_{ij} = 1, \forall i, j, \tag{4}$$

où $\sum_k c_{ij}^{(k)}$ correspond au coût d'affectation du candidat i à la place j pour l'ensemble des votants.

Comme $r_i^* = \sum_{j=1}^{n} j p_{ij}$, la fonction Euclidienne devient

$$\sum_{k=1}^{m} \sum_{i=1}^{n} (\sum_{j=1}^{n} j p_{ij} - r_{ik})^2, \tag{5}$$

puis par factorisation de p_{ij} :

$$\sum_{k=1}^{m} \sum_{i=1}^{n} (\sum_{j=1}^{n} (j - r_{ik}))^2 = \sum_{k=1}^{m} \sum_{i=1}^{n} (\sum_{j=1}^{n} (j - r_{ik})^2 p_{ij}). \tag{6}$$

Le problème de minimisation (4) est donc équivalent à :

$$\begin{aligned} \min_{\boldsymbol{r}} \sum_{i=1}^{n} \sum_{j=1}^{n} \phi_{ij} p_{ij} \quad \text{tel que} \quad \phi_{ij} = \sum_{j=1}^{n} (j - r_{ik})^2, \\ \sum_{i=1}^{n} p_{ij} = \sum_{j=1}^{n} p_{ij} = 1, \text{ et } p_{ij} \in \{0, 1\}, \end{aligned} \tag{7}$$

où ϕ_{ij} est le coût d'attribution de a_i en position j.

Si les $\boldsymbol{r}_k$ sont des ordres totaux, l'équation (5) se réécrit plus simplement :

$$\sum_{k=1}^{m} \sum_{i=1}^{n} (r_i^2 - 2 r_i r_{ik} + r_{ik}^2) = \underbrace{\sum_{k=1}^{m} \sum_{i=1}^{n} (r_i^2 + r_{ik}^2)}_{2p\times \text{ somme } n \text{ premiers entiers}} - \sum_{i=1}^{n} (r_i \sum_{k=1}^{m} r_{ik}). \tag{8}$$

Comme $\sum_{j=1}^{n} r_{ik} = pr_{i\cdot}$, où $r_{i\cdot}$ est le rang moyen de l'alternative i sur m classements et comme $r_i = \sum_j jx_{ij}$, Eq. (8) devient

$$2p\frac{n(n+1)(2n+1)}{6} - 2p\sum_{i=1}^{n}\sum_{j=1}^{n} jr_{i\cdot}p_{ij}. \tag{9}$$

Minimiser la distance de Spearman (9) revient à maximiser le terme $\sum_{i=1}^{n}\sum_{j=1}^{n} jr_{i\cdot}p_{ij}$ sous la contrainte que $P^{(k)}$ soit une matrice de permutation.

3 Décomposition en ordres totaux

L'analyse en composantes principales (ACP) ou l'analyse factorielle (AF) sont certainement les techniques les plus utilisées pour l'analyse exploratoire d'un ensemble de données continues ou binaires mais ils sont inapplicables à des tableaux de rangs. Pour extraire une base de m vecteurs de rangs "orthogonaux" capturant des informations distinctes à partir des données $\{\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_m\}$, nous avons proposé le lemme suivant.

Lemma 1 (Vigneron et Duarte [8]) *Soit un tableau de rangs $R = \{\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_m\}$. Il est toujours possible d'extraire une composante de rang total $\boldsymbol{g}_\ell$ minimisant son voisinage avec les données $\{\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_m\}$ et maximisant simultanément sa distance aux vecteurs de rangs précédemment calculés $\{\boldsymbol{g}_1, \ldots, \boldsymbol{g}_{\ell-1}\}$.*

Algorithm 1 Décomposition en ordres totaux.

Require: $P^{(1)}, \ldots, P^{(m)} \leftarrow \{\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_m\}$ {matrices de permutation $P^{(k)} = \{p_{ij}^{(k)}\}$}$\vee$ pile $A = \emptyset$ {contient les composantes réordonnées}
Ensure: $\{\boldsymbol{g}_1, \ldots, \boldsymbol{g}_m\}$ {Postcondition}
1: **for** $\ell = 1$ **to** m **do**
2: Calculer $\alpha_{ij} = \sum_{k=1}^{p} p_{ij}$, $\beta_{ij} = \sum_{k=1}^{\ell-1} z_{ij}^{(k)}$
3: $\alpha = \{\alpha_{ij}\}, \beta = \{\beta_{ij}\}$
4: **LP**$(\alpha, \beta, Z^{(\ell)})$ sous les contraintes (10) {résoud le programme linéaire}
5: $\boldsymbol{g}_\ell \leftarrow Z^{(\ell)}$
6: **end for**
7: **return** $\{\boldsymbol{g}_1, \boldsymbol{g}_2, \ldots, \boldsymbol{g}_m\}$

A l'itération ℓ, le calcul du $\ell-$ième vecteur d'ordre total $\boldsymbol{g}_\ell$ se réduit dans le cas de la distance de Spearman (voir section 2) à l'optimisation du programme non linéaire suivant :

$$\begin{aligned} &\max_{Z^{(\ell)}} \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_{ij} z_{ij}^{(\ell)} - \max_{Z^{(\ell)}} \sum_{i=1}^{n}\sum_{j=1}^{n} \beta_{ij} z_{ij}^{(\ell)} \quad \text{tel que} \\ &\alpha_{ij} = \sum_{k=1}^{p} p_{ij},\ \beta_{ij} = \sum_{k=1}^{\ell-1} z_{ij}^{(k)},\ z_{ij}^{(\ell)} + z_{ji}^{(\ell)} = 1, i < j, \\ &z_{ii} = 0\ \ z_{ij}^{(\ell)} + z_{ji}^{(\ell)} - z_{ik}^{(\ell)} \leq 1, i \neq j \neq k,\ z_{ij}^{(\ell)} \in \{0, 1\}. \end{aligned} \tag{10}$$

L'algorithme (1) s'arrête quand $\ell = m$ et produit la base de vecteurs $\{\boldsymbol{g}_1, \ldots, \boldsymbol{g}_\ell\}$ tel que $\boldsymbol{g}_\ell$ soit orthogonal aux rangs précédents $\{\boldsymbol{g}_1, \ldots, \boldsymbol{g}_{\ell-1}\}$. Jusqu'à présent il y a maintenant moyen, contrairement à principal component analysis (PCA) de calculer des index capables d'indiquer la quantité d'information captée par chaque vecteur $\boldsymbol{g}_\ell$.

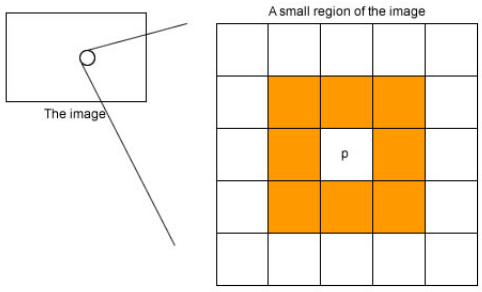
L'algorithme est donné dans le tableau 1.

Exemple d'application de la décomposition d'ordres à des images texturées

Considérons le voisinage de 8 pixels autour d'un pixel 'p' dans une image I (Fig. 3a). Les pixels colorés de la Figure (3b) connectent le pixel central avec 8,15 ou 48-voisins.

1	2	3
8		4
7	6	5

(a) Ordre dans un grille de 8 pixels.



(b) les pixels oranges forment le voisinage du pixel 'p'.

FIGURE 3

Par exemple, l'image 4×4 I dans la Fig. 4a peut être décomposée en la matrice 16×8 $R = \{\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_8\}$ (Fig. 4b) où $\boldsymbol{r}_1$ désigne la colonne du pixel 1 (sens anti-trigonométrique), $\boldsymbol{r}_2$ la colonne du pixel 2, etc. Les 8 voisins autour du pixel central peuvent être vus comme des "votants" dont on attend un ordre total. La résolution du programme linéaire Eq. (10) fournit 8 ordres totaux $\{\boldsymbol{g}_1, \ldots, \boldsymbol{g}_8\}$ (Fig. 4c).

À titre d'illustration, la LBP est appliqué à l'image originale de Lena 5a et fournit la représentation de texture donnée dans la figure 5b. A partir de la figure 5a, on obtient une série de décompositions (Figures 5c-5f) basée sur l'algo 1. Visuellement, le 1er tracé est plus informatif que le 2ème, qui lui-même est plus informatif que le 3ème, et ainsi de suite. Chacune des image 5c-5f correspond à la décomposition $\boldsymbol{g}_1$, puis $\boldsymbol{g}_2$, puis $\boldsymbol{g}_3$, etc.

On peut utiliser cet opérateur de décomposition pour remplacer le max-pooling dans un CNN et capturer des information texturées et toujours de manière non paramétrique.

	1	2	3	4
1	4	8	20	18
2	17	12	17	17
3	6	1	17	17
4	4	6	5	14

(a) Image 4×4 I.

pixel central	r_0	r_1	r_2	r_3	r_4	r_5	r_6	r_7
12	4	8	20	17	17	1	6	17
17	8	20	18	17	17	17	1	6
1	17	12	17	17	5	6	4	6
17	12	17	17	17	14	5	6	1

(b) Matrice R .

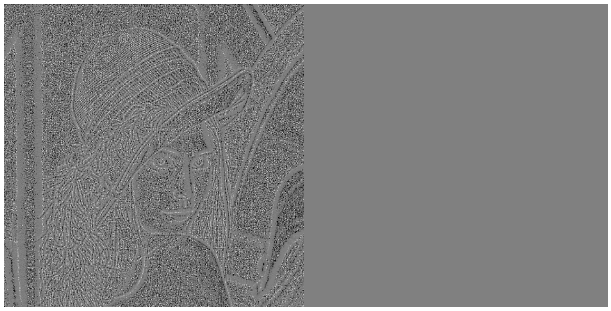
pixel central	g_0	g_1	g_2	g_3	g_4	g_5	g_6	g_7
12	1	1	4	1	4	1	3	4
17	1	4	3	2	3	4	1	3
1	4	2	2	3	1	3	2	2
17	3	3	1	4	2	2	4	1

(c) Matrice des ordres totaux Z.

FIGURE 4

(a) Image de Lena original **(b)** Classic LBP

(c) Ordre total g_1. **(d)** Ordre total g_2.



(e) Ordre total g_3. **(f)** Ordre total g_8.

FIGURE 5 – La Lena originale (a) est comparée à la représentation LBP classique (b) et aux ordres totaux obtenus par décomposition de rangs (Eq. 10). La 8-ième composante est apparemment la composante la moins informative

4 Conclusion

Pourquoi et quand devrions-nous utiliser la décomposition de rangs comme opérateur de pooling dans un CNN ? La réponse dépend des considérations suivantes : (i) petits échantillons (théorème central limite non aplicable) (ii) classes déséquilibrées (iii) images très texturées ou multimodales comme en IRM (iv) un champ perceptif faible.

Cette méthode est non supervisée non paramétrique comme son alter ego pour les données continues l'ACP ou la séparation de sources. Ses performances sont encore mal connues et seront explorées dans nos prochains travaux.

Références

[1] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features : Spatial pyramid matching for recognizing natural scene categories," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Feb 2006, vol. 2, pp. 2169 – 2178.

[2] R. Haralick, "Statistical and structural approaches to texture," in *Proceedings of the IEEE*, 1979, vol. 67, pp. 786–804.

[3] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture measures with classification based on feature distributions," *Pattern Recognition*, vol. 29, pp. 51–59, 1996.

[4] R.I. Savage, "Contributions to the theory of rank-order statistics – the trend case," *The Annals of Mathematical Statistics*, vol. 27, no. 3, pp. 590–615, Sept. 1956.

[5] R. Brüggemann and G.P. Patil, *Ranking and Prioritization for Multi-indicator Systems : Introduction to Partial Order Applications*, Environmental and Ecological Statistics. Springer-Verlag New York, 2011.

[6] V. Vigneron and L. Tomazeli Duarte, "Rank-order principal components. A separation algorithm for ordinal data exploration," in *2018 International Joint Conference on Neural Networks, IJCNN 2018, Rio de Janeiro, Brazil, July 8-13, 2018*, 2018, pp. 1–6.

[7] W.V. Gehrlein and D. Lepelley, *Voting Paradoxes and Group Coherence : The Condorcet Efficiency of Voting Rules*, Studies in Choice and Welfare. Springer-Verlag Berlin Heidelberg, 1 edition, 2011.

[8] V. Vigneron and L.T. Duarte, "Toward rank disaggregation : An approach based on linear programming and latent variable analysis," in *Latent Variable Analysis and Signal Separation*, Petr Tichavský, Massoud Babaie-Zadeh, Olivier J.J. Michel, and Nadège Thirion-Moreau, Eds., Cham, 2017, pp. 192–200, Springer International Publishing.