Projet modélisation et traitement de l'image

Capucine GARCON - Thomas WALDURA - Hadrien SALEM - Emilie SALEM

Sommaire

Involution: Inverting the Inherence of Convolution for Visual Recognition

- 1° Contexte et type d'approche
- 2° L'opérateur d'involution
  - 2°1 Rappels sur la convolution
  - 2°2 Description de l'involution
  - 2°3 Comparaison des deux opérateurs
- 3° Résultats de l'article
- 4° Nos expérimentations
- Conclusion

1° Contexte et type d'approche

- **Objectif** : Proposer une nouvelle couche pour la construction de réseaux de neurones ⇒ l'**involution** [1]
- Alternative à la convolution, ayant des propriétés "complémentaires"
- **Deux avantages principaux** :
  - Diminution importante du coût de calcul
  - Capacité à capturer les interactions spatiales entre les pixels

[1] Li et al. (2021)

2°1 Rappel sur la convolution

- Image : $X \in \mathbb{R}^{H \times W \times C_i}$
- C_o groupes de filtres de convolution :

$$\mathcal{F}_k \in R^{C_i \times K \times K} \text{ with } k = 1, 2..., C_o$$

- Opération de convolution :

$$Y_{i,j,k} = \sum_{c=1}^{C_i} \sum_{(u,v) \in \Delta_K} \mathcal{F}_{k,c,u+\lfloor K/2 \rfloor, v+\lfloor K/2 \rfloor} X_{i+u,j+v,k}$$

2°2 Description de l'involution

- Image : $X \in \mathbb{R}^{H \times W \times C}$
- Noyau d'involution : $\mathcal{H} \in \mathbb{R}^{H \times W \times K \times K \times G}$
- Pour chaque pixel, calcul du noyau d'involution H à l'aide de la fonction de génération ϕ :

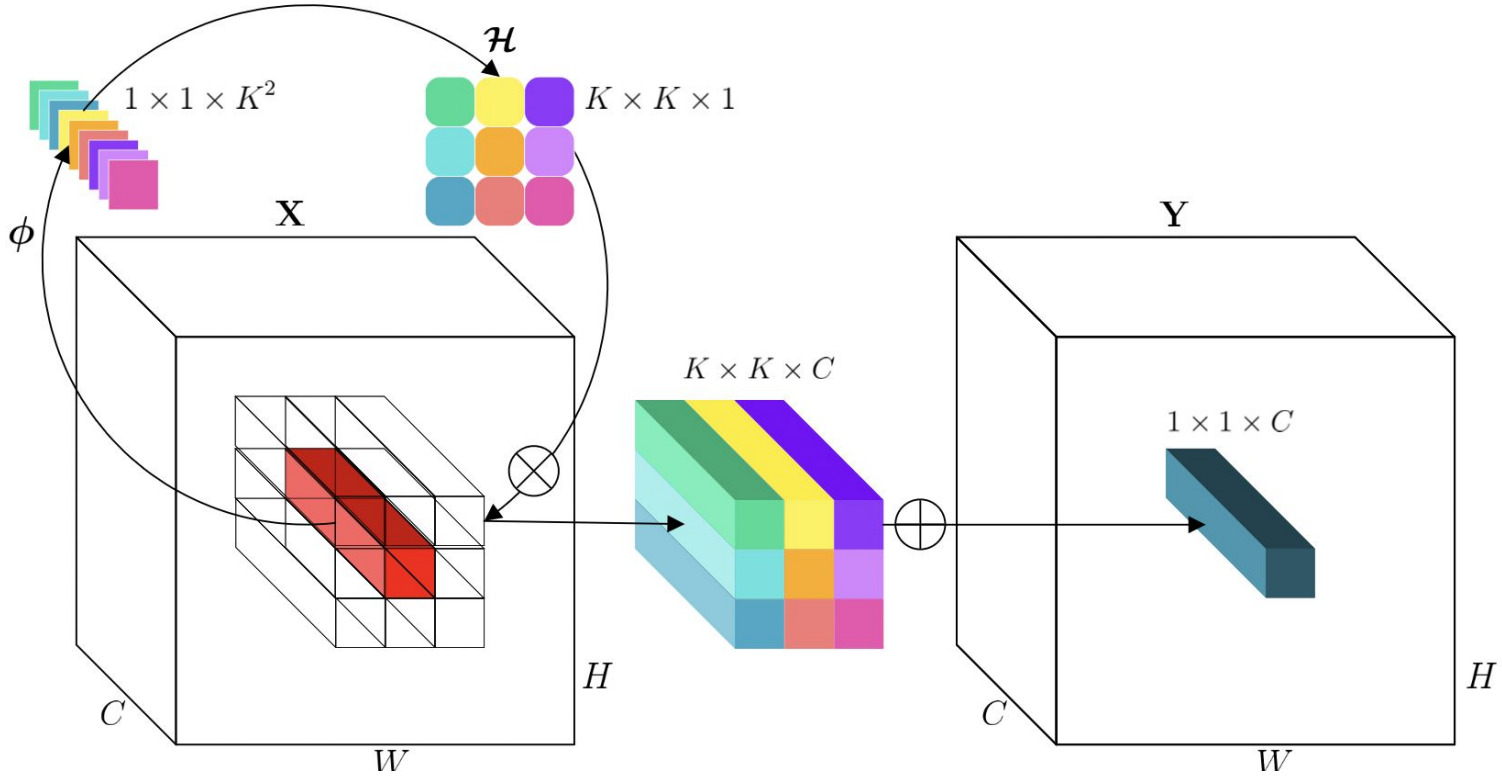
$$\mathcal{H}_{i,j} = \phi(X_{i,j}) = W_1 \sigma(W_0 X_{i,j})$$

- Opération d'involution :

$$Y_{i,j,k} = \sum_{(u,v) \in \Delta_K} \mathcal{H}_{i,j,u+\lfloor K/2 \rfloor, \lceil kG/C \rceil} X_{i+u,j+v,k}$$

Formule similaire à la *self-attention* ⇒ généralisation de ce mécanisme

2°2 Description de l'involution

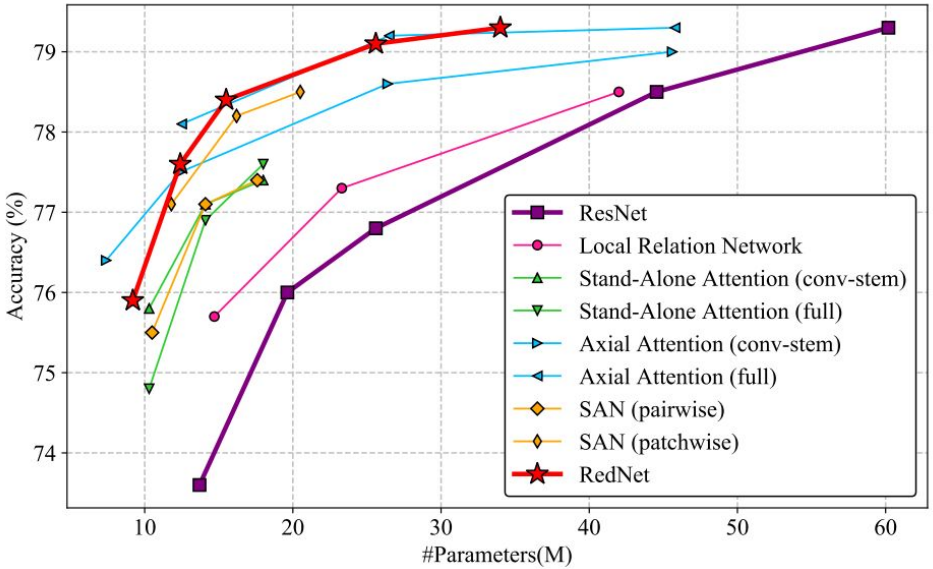


[1] Li et al. (2021)

2°3 Comparaison des deux opérateurs

Convolution	Involution
Spatial-agnostic Même filtre pour tous les pixels	*Spatial-specific* Un filtre spécifique à chaque pixel
Channel-specific Un filtre spécifique à chaque *channel*	*Channel-agnostic* Filtre partagé par plusieurs *channels*
Apprentissage individuel de chaque filtre	Apprentissage de la fonction de génération des filtres ⇒ Beaucoup moins de paramètres à apprendre !

3° Résultats de l'article



Courbe de l'accuracy en fonction du nombre de paramètres pour la classification sur le dataset ImageNet

- Comparaison avec des modèles basés sur la convolution et la *self-attention*
- Comparaison en classification, en détection d'objet et en segmentation
- **Résultats obtenus** :
  - *Accuracy* meilleure ou équivalente
  - Moins de paramètres à entraîner
  - Coût en calculs moins important

[1] Li et al. (2021)

4° Nos expérimentations

- Utilisation d'une implémentation TensorFlow de l'opérateur d'involution [2]
- Construction de modèles de même architecture pour comparer involution et convolution [4]
- Comparaison sur plusieurs datasets :
  - De l'*accuracy*
  - Du nombre de paramètres
  - Du temps d'entraînement

[2] Gosthipaty (2021), code available at `https://github.com/ariG23498/involution-tf`
[4] Lecun et al (2021)

4° Nos expérimentations

		Convolution		Involution	
Accuracy	MNIST	**98,97 %**		88,82 %	
	Fashion-MNIST	**91,91 %**		82,34%	
	CIFAR10	**64,78 %**		31,10 %	
Nombre de paramètres	MNIST	34 826		**544** (64 × *moins*)	
	Fashion-MNIST	257 162		**3 916** (65 × *moins*)	
	CIFAR10	319 178		**13 074** (24 × *moins*)	
Temps d'entraînement	MNIST	**65,26 s** (GPU)	502,75 s (CPU)	73,35 s (GPU)	**323,23 s** (CPU)
	Fashion-MNIST	**86,23 s** (GPU)	982,03 s (CPU)	479,08 s (GPU)	**621,97 s** (CPU)
	CIFAR10	**142,44 s** (GPU)	1077,86 s (CPU)	203,27 s (GPU)	**922,53 s** (CPU)

Conclusion

- *Accuracy* légèrement moins importante, mais diminution drastique du nombre de paramètres à entraîner
- En GPU, ne se reflète pas sur le temps de calcul (pas une implémentation optimisée sur noyau CUDA)
- En CPU, entraînement de l'involution plus rapide !

⇒ Montre l'intérêt de l'involution et les perspectives qu'elle pourrait offrir une fois implémentée sur GPU

Références

[1] Li, D., Hu, J., Wang, C., Li, X., She, Q., Zhu, L., Zhang, T., and Chen, Q. (2021). *Involution: Inverting the inherence of convolution for visual recognition*. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 12321–12330.

[2] Gosthipaty, A. R. (2021). Tensorflow implementation of involution. https://github.com/ariG23498/involution-tf.

[3] Reich, C., Memmel, M., and shikishima TasakiLab (2021). Pytorch implementation of involution. https://github.com/ChristophReich1996/Involution.

[4] LeCun, Y., Kavukcuoglu, K., and Farabet, C. (2010). *Convolutional networks and applications in vision*. In Proceedings of 2010 IEEE International Symposium on Circuits and Systems, pages 253–256.

Merci pour votre attention !