INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN NORMANDIE

Projet Deep Learning - Capsule Neural Network

"Dynamic Routing Between Capsules", Geoffrey E. Hinton, Nicholas Frosst, Sara Sabour - NeurIPS 2017

Lucie Clair
Léopold Maillard

March 2021

Table des matières

1 Introduction

Le modèle des "Capsule Neural Networks" a été mis au point par Geoffrey E. Hinton, Nicholas Frosst et Sara Sabour en 2017 dans leur article "Dynamic Routing Between Capsules". Il propose une alternative aux Convolutional Neural Networks (CNN) communément utilisés pour leur aptitude à extraire les caractéristiques d'une image, notamment pour les problèmes de classification. Comparé aux neurones d'un réseau standard, l'idée est ici d'encapsuler les informations pertinentes relatives aux images au sein d'une même structure. Les capsules alors obtenues encodent les paramètres d'instanciation d'une entité comme un objet ou une partie d'un objet. Elles permettent ainsi de conserver un maximum de propriétés relatives aux caractéristiques visuelles des éléments d'une image : position, taille, rotation, texture, teinte etc.

2 La notion de capsule

2.1 Comparaison d'une capsule avec un neurone traditionnel

La première différence est qu'une capsule prend un vecteur en entrée et en ressort un vecteur alors qu'un neurone traditionnel ne traite que des scalaires. Le vecteur porte de l'information sur plus de dimensions ce qui explique la capacité des capsules à utiliser et conserver plus d'informations sur les caractéristiques d'une image.

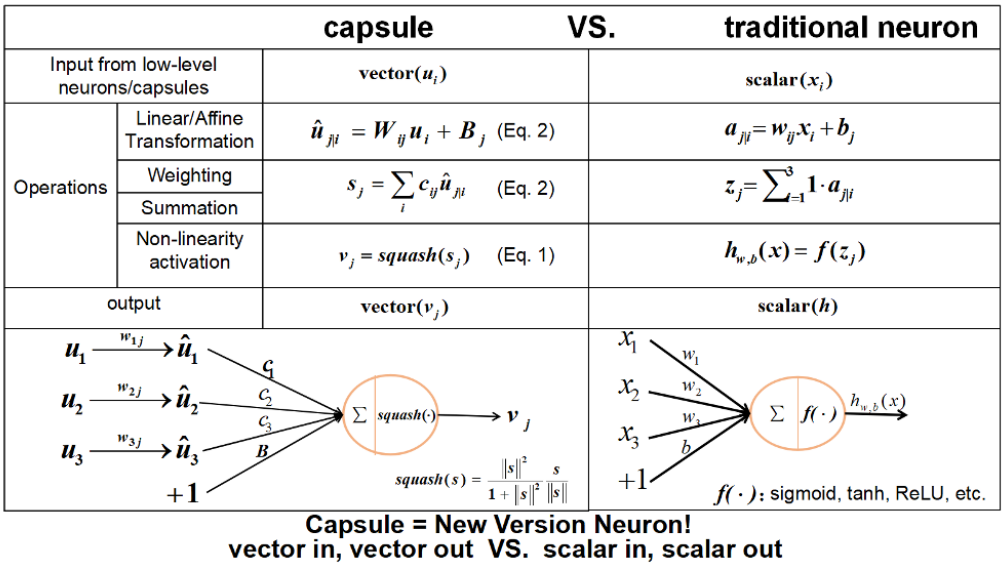

FIGURE 1 – Principales différences entres les capsules et les neurones des DNN traditionnels.[3]

2.2 L'intérêt d'un réseau à capsules

Les CNN apprennent à reconnaître les caractéristiques visuelles d'une image, puis utilisent toutes les informations apprises pour faire une prédiction. Si cette méthode est le plus souvent performante, elle présente un désavantage important : aucune information spatiale n'est utilisée et on fait alors face au *problème de Picasso* illustré sur l'image ci-dessous.

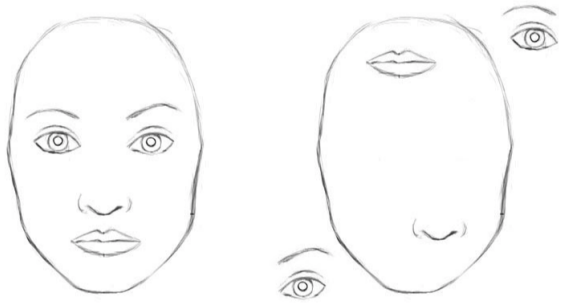

FIGURE 2 – Pour un CNN, ces 2 images sont similaires : elles comportent toutes les deux les caractéristiques d'un visage (bouche, nez, yeux etc.), même si ces dernières ne sont pas correctement disposées dans l'espace.[4]

Ainsi, la fonction de mise en commun utilisée entres les couches d'un CNN, le *max pooling*, leur a permis d'être performant, mais entraîne la perte d'informations précieuses entres les couches. En effet, seuls les neurones les plus actifs sont choisis pour la couche suivante. À propos de cette opération de mise en commun, Hinton a déclaré :

"*The pooling operation used in convolutional neural networks is a big mistake and the fact that it works so well is a disaster.*"[2]

En réponse à ce problème, il propose dans son papier un processus mettant en oeuvre des capsules, le "routing by agreement".

3 Routing by agreement

3.1 L'algorithme

La norme du vecteur en sorti d'une capsule représente la probabilité d'apparition de l'élément correspondant à la capsule. La fonction d'activation utilisée est donc toujours le "squashing" pour assurer une valeur entre 0 et 1.
Le vecteur en sortie de la capsule (pour toute sauf celles de la première couche) v_j est calculé comme la somme pondérée des vecteurs prédits $\hat{u}_i$ par les vecteurs de couplage c_{ij}.

$$v_j = \sum_i c_{ij}\hat{u}_i$$

Le vecteur prédit $\hat{u}_i$ est lui calculé par le produit du vecteur en sortie de la capsule précédente u_i et la matrice de poids w_{ij}.

$$\hat{u}_i = u_i w_{ij}$$

Le vecteur de couplage c_{ij} entre une capsule i et une capsule suivante j est calculé par la fonction "softmax" pour laquelle b_{ij} est la probabilité a priori que la capsule i soit couplée à la capsule j.

$$c_{ij} = \frac{exp(b_{ij})}{\sum_k exp(b_{ik})}$$

Les vecteurs b_{ij} sont appris au fur et à mesure de l'entraînement tout comme les poids w_{ij}. Ils sont incrémentés par la valeur de l'accord (agreement) entre les deux capsules noté a_{ij} calculé de la manière suivante :

$$a_{ij} = v_j u_i$$

Ainsi, les vecteurs de couplages entre deux capsules sont de plus en plus précis et sont directement liés à la notion d'accord entre deux capsules.

3.2 La notion d'agreement

Chaque capsule représentant la probabilité d'apparition d'un élément, l'algorithme cherche à reconnaître des objets en vérifiant la cohérence entre les output des différentes capsules. Voici un exemple très simplifié : partons du principe que nous avons deux capsule associées à un rectangle et un triangle. L'objectif est de reconnaître l'objet sur l'image ci-dessous :

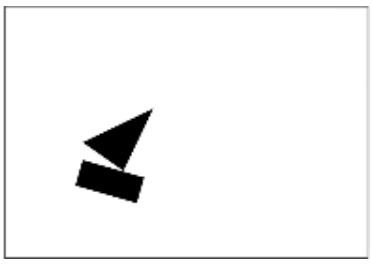

FIGURE 3 – Image simplifiée d'un bateau [1]

La capsule associée au rectangle va prédire qu'il pourrait s'agir d'un bateau ou d'une maison orienté vers le haut. En revanche, la capsule associée au triangle va prédire qu'il pourrait s'agir d'un bateau orienté vers le haut ou d'une maison à l'envers.

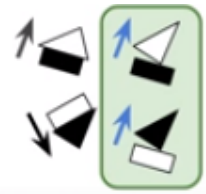

FIGURE 4 – Output des capsules [1]

La vérité se trouve là où les capsules sont d'accord : c'est un bateau orienté vers le haut.

4 L'architecture CapsNet

4.1 L'encodeur

L'architecture présentée ci-dessous est adaptée au dataset MNIST mais peut être généralisée. La structure est peu profonde puisqu'elle ne contient qu'une couche convolutionnelle 1D suivie de la couche de capsules primaires puis de la couche DigitCaps.

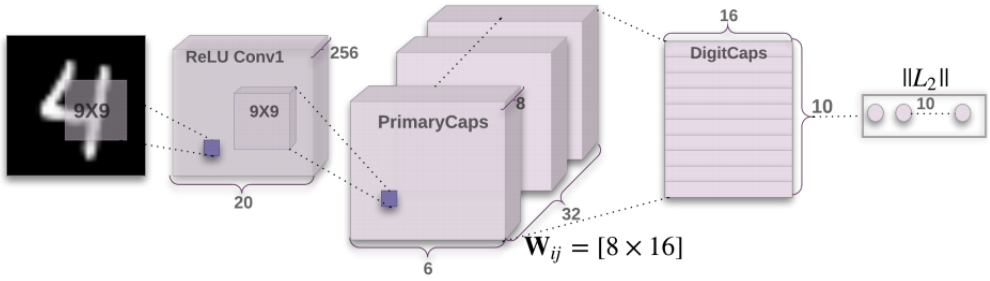

FIGURE 5 – L'architecture CapsNet [6]

La couche PrimaryCaps comporte 32 capsules primaires dont l'objectif est de créer des combinaisons des caractéristiques simples extraites par la couche de convolution. La couche DigitCaps comporte 10 capsules, une pour chaque chiffre de MNIST. Chaque capsule renvoie un vecteur de 16 dimensions qui comporte les informations d'orientation, d'échelle, d'épaisseur, d'inclinaison etc. La norme du vecteur représente la probabilité de présence de la classe (le chiffre) correspondante et permet ainsi de calculer la loss relative à la classification...

Le routing n'est effectué qu'entre les couches "PrimaryCaps" et "DigitCaps". En effet, en sortie de la première couche convolutionnelle il n'y a qu'une seule dimension et donc pas de notion d'agreement dans l'espace.

4.2 Le décodeur (reconstruction)

Le décodeur prend en entrée un vecteurs de 16 dimensions issu de la couche DigitCaps de l'encodeur correspondant à une interprétation correcte de digit. L'objectif est d'apprendre au modèle

à reconstruire l'image initiale d'entrée en calculant l'erreur quadratique de reconstruction. Il est constitué de 3 couches denses dont la dernière renvoie 784 nombres correspondant à l'intensité des 28x28 pixels de l'image reconstruite.

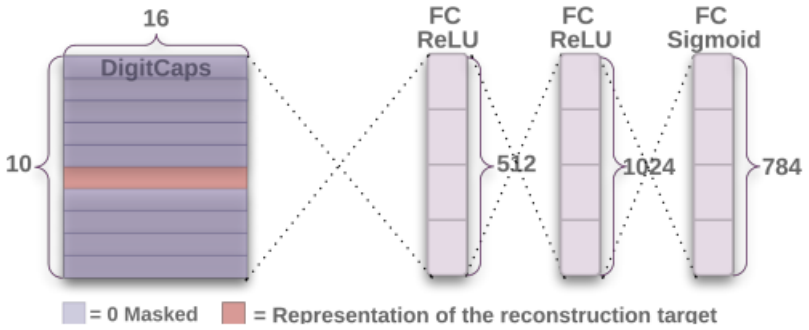

FIGURE 6 – Structure du décodeur [6]

Ainsi, plus l'erreur de reconstruction est faible, plus l'image en sortie est similaire à l'image d'entrée et donc on peut déduire que le vecteur en sortie de DigitCaps représente bien le digit. La reconstruction permet également de rendre compte de l'influence des 16 paramètres sur l'allure du digit. Ainsi, nous pouvons, en tant qu'être humain, mieux interpréter ce que représente chaque dimension.

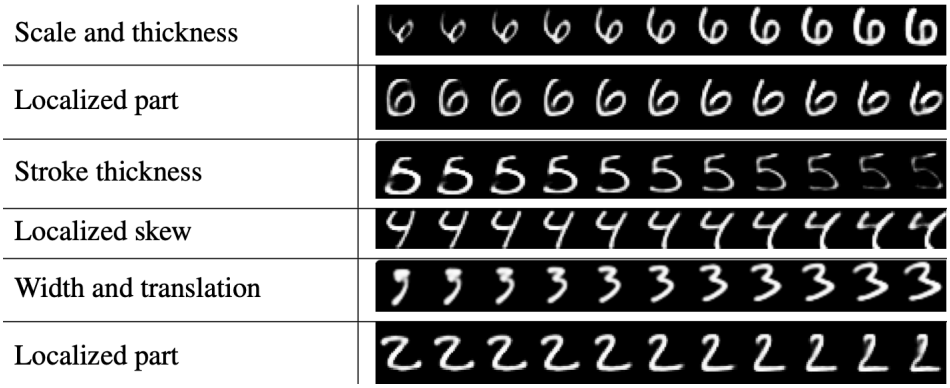

FIGURE 7 – Résultats de reconstruction après modifications d'une des 16 dimensions du vecteur DigitCaps. Chaque ligne représente la modification d'une unique dimension.[6]

5 Performances, critiques et avenir

Le modèle de réseaux de capsules ainsi défini a atteint des performances à l'état de l'art sur des datasets de référence comme MNIST ou smallNORB. Cela prouve que cette nouvelle architecture est pertinente et peut être performante. De plus, elle requiert peu de données d'entraînement ce qui est un avantage considérable dans la mesure où la récolte et la manipulation de grands jeux de données est souvent difficile. En outre, les capsules constituent une manière innovante de représenter l'information. Comme on a pu le voir, les vecteurs d'activations et leurs différents paramètres sont facile à interpréter par un humain.

Hinton a également pu démontrer que cette architecture fonctionne bien pour des objets qui se superposent avec l'exemple de MultiMNIST. Ainsi, cette technologie est particulièrement prometteuse pour les tâches de classification.

En revanche, les résultats obtenus sont peu concluants sur d'autres datasets plus complexes comme CIFAR-10 ou ImageNet. Une interprétation avancée par les auteurs est que les capsules tendent à prendre en compte toutes les informations d'une image, hors sur ces jeux de données les arrières-plan sont trop variés, rendant leur modélisation difficile. Cette quantité d'information

n'est donc pas adaptée à ce modèle ou du moins pour l'instant. Un dernier point négatif est l'entraînement particulièrement long du modèle à cause de la boucle du routing à chaque itération et de la quantité de paramètres entraînables.

On peut également mentionner le fait que les résultats du papier ne semblent pas exactement reproductibles. En effet, lors de nos recherches, nous nous sommes aperçus que les différentes implémentations de l'architecture parvenaient à se rapprocher des performances du papier, mais sans tout à fait les égaler.

Nous gardons en tête que ce concept est nouveau et est amené à évoluer pour éventuellement un jour se standardiser. En effet, les capsules continuent d'attirer l'attention des chercheurs. Une architecture publiée en début 2021, *Efficient-CapsNet* permet d'obtenir des résultats à l'état de l'art sur plusieurs datasets. De plus, elle n'utilise que 2% des paramètres du CapsNet original, rendant la phase d'entraînement particulièrement rapide.[5]

Références

[1] Aurélien Géron. *Capsule Networks (CapsNets) – Tutorial*. 2017. URL : https://www.youtube.com/watch?v=pPN8d0E3900. (accessed: 24.03.2021).

[2] Geoffrey E. Hinton. *AMA Geoffrey Hinton - Reddit Machine Learning*. 2014. URL : https://www.reddit.com/r/MachineLearning/comments/2lmo0l/ama_geoffrey_hinton/clyj4jv/. (accessed: 24.03.2021).

[3] Huadong Liao. *CapsNet-Tensorflow*. 2018. URL : https://github.com/naturomics/CapsNet-Tensorflow. (accessed: 24.03.2021).

[4] Max Pechyonkin. "Understanding Hinton's Capsule Networks." In : *Max Pechyonkin Website* (2017). DOI : https://pechyonkin.me/capsules-1/.

[5] Vittorio Mazzia, Francesco Salvetti et Marcello Chiaberge. *Efficient-CapsNet: Capsule Network with Self-Attention Routing*. 2021. eprint : arXiv:2101.12491.

[6] Sara Sabour, Nicholas Frosst, Geoffrey Hinton. "Dynamic routing between capsules". In : *Conference on Neural Information Processing Systems* (2017). DOI : https://arxiv.org/pdf/1710.09829.pdf.