Exploration d'un Resnet pour la classification d'images

Nicolas Dépelteau , Dung Nguyen
Polytechnique Montréal
nicolas.deplteau@polymtl.ca, thi-ngoc-dung.nguyen@polymtl.ca

Abstract

Dans ce rapport écrit pour notre projet dans le cadre du cours INF8225, nous présentons les bénificents de l'utilisation d'un Resnet dans le cadre de la classification d'images. Nous avons également comparé les résultats de notre Resnet avec ceux d'un réseau de neurones convolutif simple.

1 Introduction

Plusieurs travaux ont été proposés pour améliorer les performances des réseaux de neurones profonds pour la classification d'images. Parmi ces travaux, on peut citer l'article "Very Deep Convolutional Networks for Large-Scale Image Recognition" proposé par Karen Simonyan et Andrew Zisserman en 2015, qui proposait des réseaux VGG ayant jusqu'à 19 couches de convolution pour extraire des caractéristiques visuelles à partir des images. Cependant, malgré des performances de classification impressionnantes, ces réseaux de neurones profonds ont été confrontés à un problème de dégradation des performances avec l'augmentation de la profondeur. C'est dans ce contexte que l'article "Deep Residual Learning for Image Recognition" proposé par Kaiming He, Xiangyu Zhang, Shaoqing Ren et Jian Sun en 2016, a introduit les connexions résiduelles pour permettre un apprentissage en profondeur plus efficace et surmonter ce problème de dégradation.

2 Les réseaux convolutifs

Un réseau convolutif est un réseau dans lequel les neurones sont organisés en couches. Chaque couche est composée de plusieurs neurones. Chaque neurone est connecté à tous les neurones de la couche précédente. Chaque connexion est associée à un poids. Lorsque le réseau est activé, chaque neurone calcule une somme pondérée des valeurs de sortie des neurones de la couche précédente. Cette opération est une convolution dans le cas d'un réseau convolutif. Cette somme est ensuite passée à une fonction d'activation. La fonction d'activation est utilisée pour introduire une non-linéarité dans le réseau. La fonction d'activation la plus utilisée est la fonction ReLU. La fonction ReLU est définie comme suit:

$$f(x) = max(0, x) \tag{1}$$

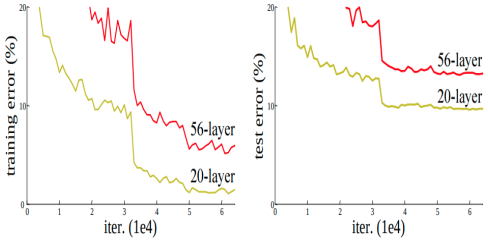

Figure 1: Problèm de dégradation: Réseau plus profond a une training error et test error plus élevée

2.1 Convolution

L'opération de convolution est une opération mathématique qui est utilisée pour extraire des caractéristiques d'une image. L'opération de convolution est définie comme suit:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{2}$$

où I est l'image d'entrée, K est le noyau de convolution et S est l'image de sortie. Le noyau de convolution est une matrice de taille $m \times n$. L'image de sortie est une image de taille $m \times n$. L'image de sortie est calculée en faisant glisser le noyau de convolution sur l'image d'entrée.

2.2 Pooling

Le pooling est une opération qui est utilisée pour réduire la taille de la carte des caractéristiques. Le pooling est effectué en faisant glisser une fenêtre sur la carte des caractéristiques. La valeur de sortie de la fenêtre est la valeur maximale de la fenêtre dans le cas du max pooling. Toutefois il existe aussi d'autres types de pooling comme le average pooling. Le average pooling est une opération qui est similaire au max pooling. La valeur de sortie de la fenêtre est la moyenne des valeurs de la fenêtre. La taille de la fenêtre est un hyperparamètre du réseau.

2.3 Le problème de dégradation

Le problème de dégradation est un problème qui est présent dans les réseaux profonds. Ce phénomène est contre-intuitif, car on s'attendrait à ce que l'ajout de couches supplémentaires permette au réseau de mieux capturer les

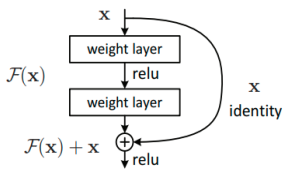

Figure 2. Residual learning: a building block.

Figure 2: Skip Connection dans un block de ResNet.

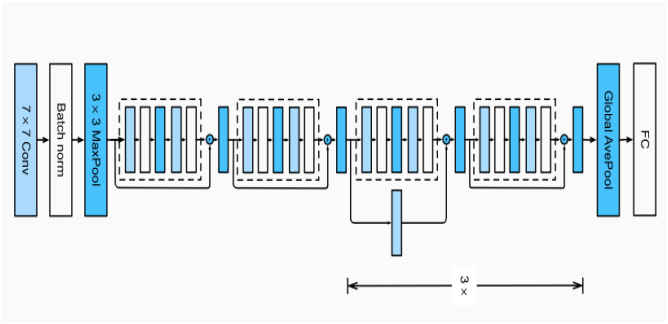

Figure 3: Architecture de ResNet-18

caractéristiques des images et donc d'augmenter ses performances. Cependant, les auteurs de l'article "Deep Residual Learning for Image Recognition" proposent une explication à ce phénomène en affirmant que les couches profondes ont des difficultés à apprendre des fonctions identité, c'est-à-dire à conserver l'information non perturbée par la transformation non-linéaire introduite par la couche. Cette perte d'information a un effet négatif sur les performances du réseau lors de l'ajout de couches supplémentaires.

3 Resnet

Pour pallier ce problème, les auteurs proposent d'utiliser des connexions résiduelles pour permettre un apprentissage en profondeur plus efficace.

3.1 Skip connection

La caractéristique la plus particulière de ResNet est que la "skip connection" est appliquée à l'intérieur de chaque bloc pour aider le modèle à garder les résidus du passé vers le futur.

Les auteurs ont ajouté une couche de mapping d'identification pour copier la couche apprise peu profonde dans une couche plus profonde. Cette couche de mapping est directement ajouté avec le bloc d'entrée précédent dans le bloc de sortie avec la même forme. Le résidu entre la couche la plus profonde par rapport à la couche la moins profonde est:

$$F(x, Wi) = H(x) - x \tag{3}$$

où x est la sortie de la couche apprise peu profonde, Fx le feed forward non-linéairede x, Wi est le paramètre de la couche de mapping d'identification et est appris par le modèle et $H(x)$ la sortie de réalité de l'ensemble du processus. Le processus d'apprentissage étudie en fait la transformation non linéaire $F(x, Wi)$ de résidu après chaque bloc entre l'entrée et la sortie. La sortie de la couche de mapping d'identification est ajoutée à la sortie de la couche apprise peu profonde pour former la sortie finale de ce bloc. Dans l'autre bloc, nous avons appliqué une transformation convolutive avant de "skip connection" afin d'étudier l'apprentissage des caractéristiques.

$$y = F(x, Wi) + Conv(x) \tag{4}$$

Pour conserver la forme de sortie inchangée et réduire le nombre total de paramètres, les couches $Conv$ ont normalement une taille de noyau de 1 x 1.

3.2 Batch Normalization

ResNet est la première architecture appliquée Batch normalization à l'intérieur de chaque. Batch normalization aide le modele à rester stable lors de la descente de gradient et à soutenir la convergence rapide du processus de formation vers un point optimal. Batch normalization est appliquée sur chaque mini-lot par normalisation standard $N(0, 1)$. On a $B = x1, x2, ...xm$ m foot index indique la taille de mini-lot. Tous les échantillons d'entrée sont redimensionnés comme ci-dessous:

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{5}$$

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu)^2 \tag{6}$$

Le nouvel échantillon normalisé est:

$$\hat{x}_i = \frac{x_i - \mu}{\sigma} \tag{7}$$

4 Architecture

En général, l'architecture commune des différents modèles ResNet profonds a la même règle. L'idée cohérente appliquée pendant l'ensemble des modèles, à savoir la couche de normalisation par lots, suit juste derrière chaque couche convolutive. Comme montré dans la figure 3, l'architeture d'un Resnet18 contient un bloc résiduel entouré d'un rectangle en tirets avec 5 couches empilées dans la figure. Les deux blocs résiduels de départ sont des blocs d'identification. Après cela, nous répétons trois fois [couche convolutive + couche mapping d'identification]. Enfin, la mise en commun de la moyenne globale s'applique pour capturer les caractéristiques générales en fonction de la dimension de profondeur et transmettre la sortie finale entièrement connectée.

5 Expérimentation

Dans ce rapport, nous allons explorer la performance de trois modèles de ResNet : ResNet-18, ResNet-34 et ResNet-50. Nous avons entraîné les trois modèles de ResNet sur un jeu de données d'images, en utilisant la bibliothèque PyTorch.

Figure 4: Résultats des modèles ResNet sur le jeu de données CIFAR-10

5.1 Le jeu de données

Le jeu de données utilisé est le jeu de données CIFAR-10, qui contient 50 000 images d'entraînement et 10 000 images de test. Les images sont de taille 32x32 pixels et appartiennent à 10 classes différentes. Les classes sont : avion, automobile, oiseau, chat, cerf, chien, grenouille, cheval, bateau et camion. Les images sont en couleur, chaque pixel est représenté par trois valeurs de 0 à 255, une pour chaque canal de couleur (rouge, vert et bleu).

5.2 Les hyperparamètres

Nous avons utilisé une fonction de perte de type "cross-entropy" et un optimiseur de type "Adam" pour entraîner les modèles. Les modèles ont été entraînés pendant 10 époques.

5.3 Les résultats

Les résultats de l'expérimentation sont présentés dans la graphique 4.

Nous constatons que les trois modèles ont une précision élevée sur le jeu de données CIFAR-10. La précision augmente avec la profondeur du réseau. ResNet-50 est le modèle qui obtient la meilleure performance, avec une précision de 94,1% sur l'ensemble de test.

6 Analyse critique de l'approche

L'utilisation du jeu de données CIFAR-10 est un bon choix pour tester la performance des modèles de ResNet, car il s'agit d'un jeu de données standard pour la reconnaissance d'images et il est assez complexe pour évaluer la performance des modèles. Cependant, le choix d'un jeu de données de taille plus importante pourrait permettre d'évaluer la performance des modèles à une plus grande échelle. a durée de l'entraînement de 10 époques semble raisonnable pour le jeu de données CIFAR-10 et pour les modèles de ResNet testés. Il serait intéressant d'explorer l'impact de la durée de l'entraînement sur la performance des modèles et de trouver un juste équilibre entre la performance et la durée d'entraînement.

7 Conclusion

Ce projet a permis d'explorer des performances des réseaux de neurones profonds pour la classification d'images, en se concentrant sur les modèles ResNet qui ont été introduits pour surmonter les problèmes de dégradation de performances rencontrés par les réseaux profonds traditionnels. Les résultats obtenus ont confirmé l'efficacité des modèles ResNet pour la reconnaissance d'images, en montrant que les modèles de ResNet profonds (ResNet-34 et ResNet-50) ont atteint des performances de classification élevées sur le jeu de données CIFAR-10. Depuis leur introduction en 2016, les modèles ResNet ont été largement utilisés dans divers domaines, notamment la reconnaissance d'images, la segmentation d'images, la détection d'objets et la classification de textes. Les modèles utilisant ResNet pour différentes tâches : ResNeXt, Wide ResNet, YOLO, GANs...etc.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[2] dive into deep learning. "Residual Networks (ResNet) and ResNeXt", 2016