Exploration d'un Resnet pour la classification d'images

Nicolas Dépelteau - 2083544 , Dung Nguyen - 2916011
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 bénéfices 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 [Simonyan and Zisserman, 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 [He *et al.*, 2015], a introduit les connexions résiduelles pour permettre un apprentissage en profondeur plus efficace et surmonter ce problème de dégradation. Dans ce texte nous allons présenter les réseaux de neurones convolutifs et les connexions résiduelles ainsi que nos expériences sur ceux-ci.

2 Les réseaux convolutifs (CNN)

Un réseau convolutif est un réseau dans lequel les neurones sont organisées 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:

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

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 Regroupement (Pooling)

Le regroupement est une opération qui est utilisée pour réduire la taille de la carte des caractéristiques. Le regroupement 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 regroupement comme le "average pooling". Le regroupement par la moyenne 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 hyper paramètre du réseau.

2.3 Le problème du gradient qui disparaît (Vanishing gradient problem)

Le problème du gradient qui disparaît est un problème qui est présent dans les réseaux profonds. Ce problème est causé par la fonction d'activation ReLU. La fonction ReLU est définie comme suit:

$$f(x) = \max(0, x) \tag{3}$$

La dérivée de la fonction ReLU est définie comme suit:

$$f'(x) = \begin{cases} 1 & \text{si } x > 0 \\ 0 & \text{sinon} \end{cases} \tag{4}$$

Lorsque la fonction ReLU est utilisée comme fonction d'activation, la dérivée de la fonction d'activation est soit 0

ou 1. Lorsque la dérivée de la fonction d'activation est 0, le gradient est 0. Lorsque le gradient est 0, le poids n'est pas mis à jour. Lorsque le poids n'est pas mis à jour, le réseau ne peut pas apprendre. C'est pourquoi le problème du gradient qui disparaît est un problème qui empêche les réseaux profonds d'apprendre puisqu'il y a plus de couches à traverser lors de la propagation par l'arrière. Effectivement, un gradient nul à une couche profonde empêche la mise à jour des poids des couches précédentes. Et cela se produit de plus en plus souvent lorsque la profondeur du réseau augmente.

2.4 Le problème de dégradation

Le problème de dégradation est un problème qui est présent dans les réseaux profonds.

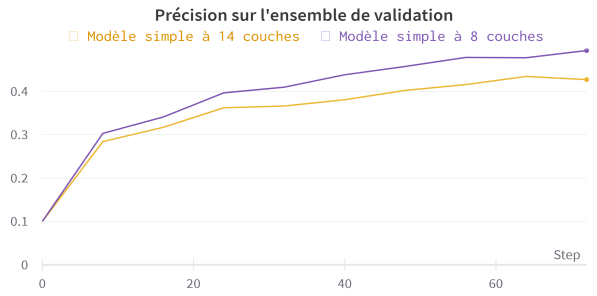

Figure 1: Problème de dégradation: Réseau plus profond avec une plus petite précision

Ce phénomène est contre-intuitif, car on s'attendrait à ce que l'ajout de couches supplémentaires permette au réseau de mieux capturer les caractéristiques des images et donc d'augmenter ses performances. Cependant, les auteurs de l'article "Deep Residual Learning for Image Recognition" [He *et al.*, 2015] proposent une explication à ce phénomène en affirmant que les couches profondes ont des difficultés à apprendre des fonctions identités, 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 [He *et al.*, 2015]. En effet, au lieu d'apprendre directement la fonction $H(x)$ où x est l'entrée d'une pile de couche dont la sortie est $H(x)$. Le réseau apprend plutôt son résidu $F(x) = H(x) - x$. La fonction $F(x)$ est plus facile à apprendre que la fonction $H(x)$, car elle est plus proche de la fonction identité. La fonction $H(x)$ est alors approximée par la fonction $F(x) + x$. Le réseau Resnet est composé de plusieurs blocs de convolution, puis d'une couche dense complètement connectée.

3.1 Connection résiduelle

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

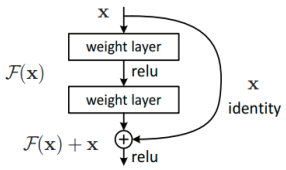

Figure 2. Residual learning: a building block.

Figure 2: Connection résiduel dans un block de ResNet. [He *et al.*, 2015]

Le bloc de ResNet est composé de deux couches de convolution et d'une couche pour ajuster les dimensions entre $F(x)$ et x. La fonction du bloc i est définie comme suit:

$$H_i(x) = \sigma(W_{i,2} * \sigma(W_{i,1} * x) + W_{i,3} * x) \tag{5}$$

$$W_{i,3} = \begin{cases} I \text{ si les dimensions de x sont identiques à la sortie} \\ W_{i,3} \text{ avec un noyau de } 1 \times 1 \text{ sinon} \end{cases} \tag{6}$$

où x est l'entrée du bloc, σ est la fonction d'activation ReLU, $W_{i,1}$ et $W_{i,2}$ sont les paramètres de la couche de convolution et $W_{i,3}$ est le paramètre de la couche de la connection résiduelle.

3.2 Normalisation par lots (Batch Normalization)

ResNet est la première architecture a appliqué normalisation par lots à l'intérieur de chacune des ses couches. La normalisation par lots aide le modèle à rester stable lors de la descente de gradient et à soutenir la convergence rapide du processus de formation vers un point optimal. La normalisation par lots est appliquée sur chaque mini-lot par une normalisation standard à l'aide d'une gaussienne $N(0, 1)$. On a $B = x_1, x_2, \ldots x_m$. où m indique la taille du mini-lot. Tous les échantillons d'entrée sont redimensionnés comme ci-dessous:

$$\mu = \frac{1}{m} \sum_{i=1}^{m} x_i \tag{7}$$

$$\sigma^2 = \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu)^2 \tag{8}$$

Le nouvel échantillon normalisé est:

$$\hat{x}_i = \frac{x_i - \mu}{\sigma} \tag{9}$$

4 Architecture

L'architecture des modèles que nous avons utilisé est basée sur l'architecture générique de ResNet présentée dans l'article "Deep Residual Learning for Image Recognition" [He *et al.*, 2015]. Cette architecture est composée de plusieurs blocs de convolution, puis d'une couche dense complètement connecté. Pour la version avec résiduel, il y

a l'opération de l'addition entre l'entrée du bloc avec ajustement si nécessaire et la sortie, comme montré dans la figure 3. Il y a également une normalisation par lots après les convolutions. Il y a un dropout avec une probabilité de 0.5.

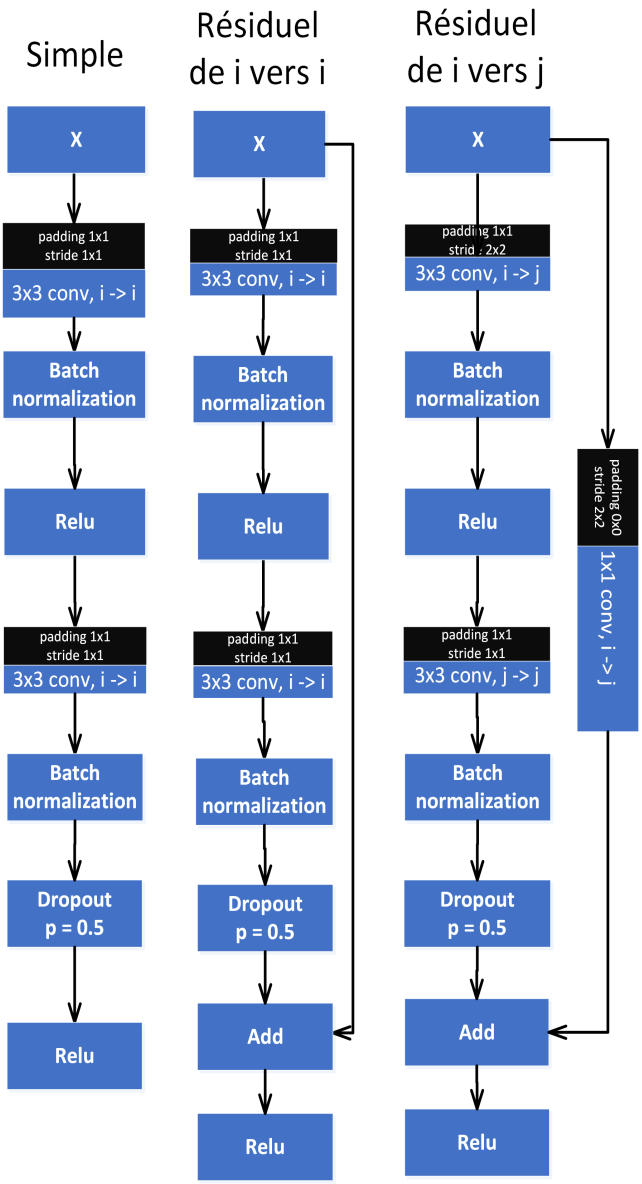

Figure 3: Blocs résiduels

Le tableau 1 montre les différentes architectures que nous avons utilisées. Les architectures sont définies par leur nombre de blocs résiduels. Par exemple, la plus petite architecture avec seulement 8 couches contient des blocs résiduels respectifs de 16 à 32, 32 à 64 et 64 à 128. Il ne faut pas oublier que ces blocs sont insérés dans l'architecture générique de ResNet présentée dans la figure 4.

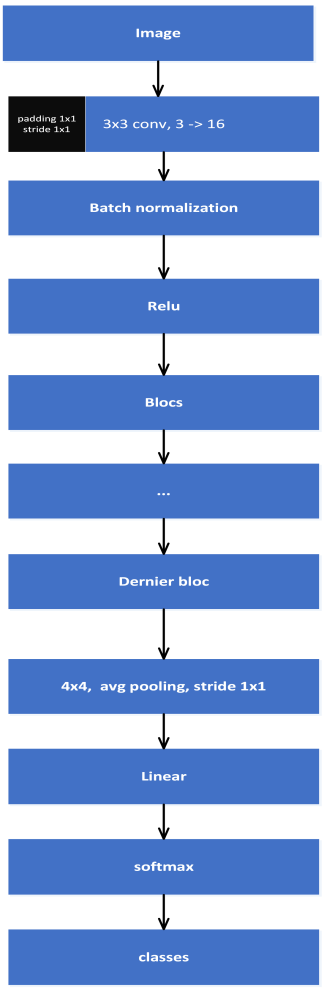

Figure 4: Architecture générique du model

Blocs résiduels	8	14	18	34	50
$16 \rightarrow 16$	-	1	1	3	6
$16 \rightarrow 32$	1	1	1	1	1
$32 \rightarrow 32$	-	1	1	3	5
$32 \rightarrow 64$	1	1	1	1	1
$64 \rightarrow 64$	-	-	1	5	5
$64 \rightarrow 128$	1	1	1	1	1
$128 \rightarrow 128$	-	1	1	2	5

Table 1: Nombre de blocs résiduels pour chacune des architectures ayant un nombre de couche différent

5 Expérimentation

Dans ce rapport, nous allons explorer la performance de cinq modèles de ResNet: ResNet-8, ResNet-14, ResNet-18, ResNet-34 et ResNet-50. Nous avons entraîné les cinq modèles de ResNet sur un jeu de données d'images, en utilisant la bibliothèque PyTorch.

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 32×32 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 représentée par trois valeurs de 0 à 255, une pour chaque canal de couleur soit le rouge, le vert et le bleu.

5.2 Les hyper paramètres

Nous avons utilisé une fonction de perte de type "cross-entropy" et un optimiseur de type "Adam" paramétré avec 0.9 et 0.99 pour entraîner les modèles. Les modèles ont été entraînés pendant 10 époques avec un taux d'apprentissage de 0.001.

5.3 Les résultats

Commençons par les résultats des modèles simples, c'est-à-dire sans résiduel. La figure 5 montre les résultats des modèles simples. On peut voir que les modèles avec 8 et 14 couches ont une précision dans le tableau 2 de 49.4% et 42.7% respectivement. On peut y remarquer le problème de la dégradation du modèle, car le modèle avec 14 couches ont une erreur plus importante que celui de 8 couches. On peut également observer le phénomène du gradient qui disparaît pour le modèle avec 18 et plus couches, car le modèle avec 18 couches a une précision de 10%, et les modèles avec 34 et 50 couches n'ont pas pu être entraînés. En effet, ces modèles de plus grande taille ne réussissent pas à converger, car le gradient disparaît.

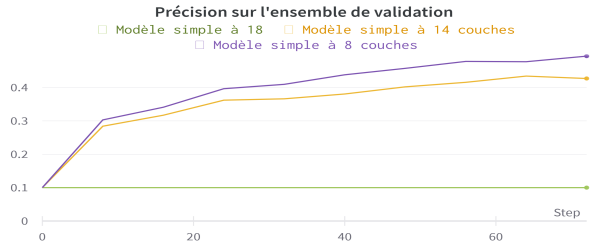

Figure 5: Résultats des modèles simples sur le jeu de données CIFAR-10

Taille du modèle	Précision après 10 époques en %
8	49.4
14	42.7
18	10.0
34	-
50	-

Table 2: Résultats des expérimentations avec les modèles simples

Par la suite, pour les résultats des modèles résiduels, la figure 6 montre les résultats des modèles résiduels. On peut voir que les modèles avec 8 et 14 couches ont une précision dans le tableau 3 de 53.5% et 55% respectivement. Les modèles résiduels ont une meilleure performance que les modèles simples. Le problème du gradient qui disparaît est alors atténué par les connection résiduelles. On peut également observer que les modèles avec 18 couches est plus performant que ceux de 14 et 8 couches. Cependant, les modèles avec 34 et 50 couches ont une précision de 50.1% et 51.5% respectivement, ce qui est moins performant que les modèles avec 8 et 14 couches. Par contre, le modèle à 50 couches est plus performant que celui à 34 couches. Donc, on peut conclure que les modèles résiduels de plus grande taille ne sont pas nécessairement plus performants que les modèles résiduels de plus petite taille. Il en reste que le nombre de couches est grandement supérieur.

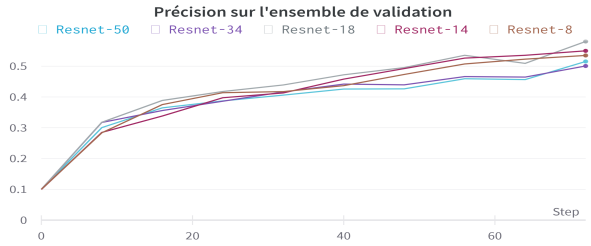

Figure 6: Résultats des modèles résiduels sur le jeu de données CIFAR-10

Taille du modèle	Précision après 10 époques en %
8	53.5
14	55.0
18	58.0
34	50.1
50	51.5

Table 3: Résultats des expérimentations avec les modèles résiduels

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. De plus, il y a pas beaucoup de données, ce qui permet de tester rapidement les modèles. La durée de l'entraînement de 10 époques semble raisonnable pour le jeu de données CIFAR-10 et pour les modèles de ResNet testés. Il est arrivé que les modèles ne convergeaient pas, car le gradient disparaissait. Cependant, suite à une réinitialisation des poids, les modèles ont convergé. Il est donc possible que nos résultats soient biaisés dans ce sens puisque pour chacune des tailles nous avons pris le premier modèle qui a convergé. Il aurait été intéressant de faire plusieurs essais pour chaque taille de modèle afin de pouvoir comparer les moyennes de ces échantillons au lieu d'un exemplaire seulement, notamment pour la performance du modèle à 50 couches qui est supérieurs à celui à 34 couches, mais inférieur à celui à 18 couches. Il est possible que ce résultat soit dû à un biais de sélection. Il aurait également été intéressant de tester les modèles avec plus d'époques, car il est possible que les modèles de plus grande taille convergent après plus d'époques. Cependant, il aurait fallu augmenter le

nombre d'époques pour tous les modèles, car il est possible que les modèles de plus petite taille convergent plus rapidement que les modèles de plus grande taille. Il aurait également été intéressant de tester les modèles avec des hyper paramètres différents, car il est possible que les modèles de plus grande taille aient besoin d'hyper paramètres différents pour converger vers des meilleurs résultats. Cependant, avec la restriction de temps et de ressources informatiques pour l'entraînement des modèles, il n'a pas été possible de tester ces différentes approches. Les modèles ont été entraînés sur un ordinateur portable personnel avec une carte graphique NVIDIA GeForce RTX 2060. Aussi, l'ajout de la normalisation de lots a peut-être nuit à la performance des modèles simples, puisque le modèle résiduel de l'article original l'utilisait sur les modèles résiduels à chaque bloc. Alors, nous avons laissé la normalisation de lots pour les modèles simples. Somme toute, les résultats obtenus sont intéressants, car ils montrent que les modèles résiduels sont plus performants que les modèles simples pour la classification d'images.

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 ont atteint des performances de classification élevées par rapport aux modèles simples 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 peuvent être utilisés pour les tâches tel que ResNeXt, Wide ResNet, YOLO, GANs, etc.

References

[He *et al.*, 2015] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[Simonyan and Zisserman, 2015] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.