Réseaux de neurones convolutifs pour classification d'images sur CIFAR-10 et CIFAR-100

Sanmar Simon
École Polytechnique Montréal, Canada
sanmar-yared.simon@polymtl.ca

Abstract

Ce projet vise à étudier les performances de cinq réseaux neuronaux convolutifs (CNN) pré-entraînés sur ImageNet pour la classification d'images sur les ensembles de données CIFAR-10 et CIFAR-100. Les CNN utilisés dans cette étude sont VGG16, EfficientNetV2_M, EfficientNet_B0, ResNet-50 et DenseNet-121. Nous avons affiné ces modèles pré-entraînés sur les ensembles de données CIFAR-10 et CIFAR-100 et évalué leurs performances à l'aide de diverses mesures telles que l'exactitude, la précision, le rappel et le score F1.
Nos résultats ont montré que tous les modèles pré-entraînés ont atteint une grande précision sur les deux ensembles de données, EfficientNetV2_M surpassant les autres sur les deux ensembles de données CIFAR-10 et CIFAR-100. Nous avons également analysé les ressources informatiques nécessaires à l'apprentissage de chaque modèle et avons constaté que EfficientNetV2_M avait besoin de moins de ressources pour l'apprentissage, tandis que VGG16 en avait besoin de plus.
Dans l'ensemble, cette étude souligne l'importance de la sélection d'un modèle pré-entraîné approprié pour une tâche de classification d'images donnée, ainsi que l'importance de prendre en compte l'efficacité informatique lors de l'ajustement de ces modèles.

1 Introduction

La classification d'images est une tâche fondamentale en vision par ordinateur, et le développement de techniques d'apprentissage profond a révolutionné le domaine. Les réseaux de neurones convolutifs (CNN) se sont imposés comme les modèles de pointe pour la classification d'images, et de nombreux modèles CNN pré-entraînés sont disponibles pour un ajustement fin (fine-tuning) sur des ensembles de données spécifiques. Ces dernières années, plusieurs études ont exploré les performances des réseaux CNN pré-entraînés dans des tâches de classification d'images, notamment sur les ensembles de données CIFAR-10 et CIFAR-100.

Par exemple, une étude de [He et al., 2016] a examiné les performances de plusieurs architectures CNN, y compris ResNet, sur les ensembles de données CIFAR-10 et CIFAR-100. Leurs résultats ont montré que ResNet a surpassé les autres modèles CNN sur les deux jeux de données, atteignant une précision de pointe. De même, une étude de [Tan et Le, 2019] a proposé une nouvelle famille de modèles CNN appelée EfficientNet, qui a atteint une précision de pointe sur plusieurs repères de classification d'images, y compris CIFAR-10.

D'autres études ont exploré les performances des CNN pré-entraînés sur des tâches de classification d'images plus spécialisées. Par exemple, une étude de [Kumar et al., 2021] a examiné les performances de modèles CNN pré-entraînés pour la classification de la rétinopathie diabétique, obtenant une grande précision en utilisant des modèles finement ajustés. Une autre étude de [Liao et al., 2021] a exploré les performances des CNN pré-entraînés pour la classification COVID-19 à partir d'images de radiographie thoracique.

Dans la présente étude, nous nous appuyons sur ces travaux antérieurs et examinons les performances de cinq CNN pré-entraînés différents, notamment VGG16, EfficientNetV2_M, EfficientNet_B0, ResNet-50 et DenseNet-121, sur les ensembles de données CIFAR-10 et CIFAR-100. Notre objectif est de déterminer quels CNN pré-entraînés sont les plus efficaces pour la classification d'images sur ces ensembles de données, et de donner un aperçu des ressources informatiques nécessaires pour entrainer ces modèles.

2 Approche Théorique

Dans cette partie nous allons ne pencher sur la théorie derrière les réseaux de neurones convolutifs, nous verrons les spécificités des modèles que nous avons choisis, ainsi que l'approche d'ajustement d'un modèle pré-entraîné.

2.1 Réseaux de neurones convolutifs

L'architecture de base d'un réseau de neurones convolutifs est composé des trois couches suivantes.

Couche convolutive
La couche convolutive est l'élément central d'un CNN. Elle applique un ensemble de filtres pouvant être appris (également appelés noyaux ou poids) aux données d'entrée. Chaque filtre glisse sur les données d'entrée, effectuant une multiplication par élément suivie d'une somme, ce qui donne une valeur de sortie unique. Les sorties de chaque filtre sont ensuite empilées pour créer une carte de caractéristiques qui représente l'activation de ce filtre sur les données d'entrée. L'objectif de la couche convolutive est d'apprendre et d'extraire des données d'entrée les caractéristiques pertinentes pour la tâche à accomplir, comme la détection des contours, la reconnaissance des textures ou la détection des objets.

Couche de mise en commun
Une couche de mise en commun est généralement utilisée après une couche convolutive pour réduire les dimensions spatiales des cartes de caractéristiques. Pour ce faire, la carte des caractéristiques est divisée en petites régions qui ne se chevauchent pas et la valeur maximale, minimale ou moyenne de chaque région est prise en compte. Cette opération réduit le nombre de paramètres dans le réseau, rend le modèle moins sensible aux petites variations de l'entrée et permet d'éviter l'ajustement excessif.

Couche de classification
La couche de classification est généralement située à la fin du CNN et prend comme entrée les cartes de caractéristiques aplaties des couches précédentes. Elle se compose d'un ensemble de neurones, dont chacun est connecté à tous les neurones de la couche précédente. L'objectif de la couche entièrement connectée est d'utiliser les caractéristiques extraites des couches précédentes pour classer les données d'entrée. Cette couche est souvent suivie d'une fonction d'activation softmax, qui convertit la sortie de chaque neurone en une distribution de probabilité sur les classes possibles. La classe ayant la probabilité la plus élevée est alors choisie comme sortie prédite.

2.2 Architectures utilisées

Les cinq différents modèles utilisés possèdent tous des spécificités dans leur architecture. On va brièvement récapituler ces dernières.

VGG-16
Le VGG-16 se compose de 16 couches, dont 13 couches convolutives et 3 couches entièrement connectées. Chaque couche convolutive comporte 3x3 filtres et un pas de 1. Elle est suivie d'une fonction d'activation linéaire rectifiée (ReLU) et d'une couche de mise en commun maximale de 2x2. Les couches entièrement connectées comportent 4096 neurones chacune et sont suivies d'une fonction d'activation softmax pour la classification.

EfficientNetV2_M
EfficientNetV2_M est un modèle de taille moyenne qui utilise une combinaison de couches convolutives et de blocs mobiles à goulot d'étranglement inversé (mobile inverted bottleneck blocks). Le modèle se compose de couches de tiges, de plusieurs blocs de goulots d'étranglement inversés de largeurs et de profondeurs variables, et d'une couche de tête pour la classification. Les blocs de goulot d'étranglement inversés utilisent des convolutions en profondeur et ponctuelles pour réduire le nombre de paramètres et accroître l'efficacité.

EfficientNet-B0
EfficientNet-B0 est le plus petit modèle de la famille EfficientNet, conçu pour les environnements à ressources limitées. Il utilise une architecture similaire à celle d'EfficientNetV2_M, mais avec moins de couches et des filtres plus petits. Le modèle se compose de couches de tiges, de multiples blocs de goulots d'étranglement inversés avec des largeurs et des profondeurs variables, et d'une couche de tête pour la classification.

ResNet-50
ResNet-50 est une architecture CNN profonde qui a introduit le concept de connexions résiduelles. Le modèle se compose de couches souches, de blocs résiduels multiples avec un nombre variable de couches convolutives, et d'une couche principale pour la classification. Les connexions résiduelles facilitent le flux d'informations dans le réseau et permettent d'atténuer le problème de disparition du gradient.

DenseNet-121
DenseNet121 est une architecture CNN qui utilise des blocs denses, qui concatènent les sorties de toutes les couches précédentes comme entrées de la couche actuelle. Le modèle se compose de couches souches, de blocs denses multiples avec un nombre variable de couches convolutives, et d'une couche principale pour la classification. Les blocs denses permettent une utilisation plus efficace des paramètres et des gradients, et favorisent la réutilisation des caractéristiques.

2.3 Fin ajustement des modèles

Les cinq modèles énumérés dans la sous-section précédente sont des modèles d'apprentissage profond et nécessitent donc d'importantes ressources de calcul afin de pouvoir les entraîner. C'est pour cette que l'on décider d'utiliser des versions pré-entraînés de ces modèles afin de réduire les ressources nécessaires à l'entraînement. Ainsi nous avons pris des modèles entraînés sur le jeu de données ImageNet afin de pouvoir les ajuster sur les jeux de données CIFAR-10 et CIFAR-100. Ajuster un CNN pré-entraîné sur un nouveau jeu de données consiste principalement à adapter la taille de la couche de sortie afin qu'elles reflètent correctement le nombre de classes de notre jeu de données. ImageNet est un jeu de données composés de 1000 classes. Or CIFAR-10, et CIFAR-100 sont respectivement composés de 10 et 100 classes. Pour régler ce problème ont modifie simplement la taille de sortie de nos modèles en fonction du jeu de données sur lequel on veut les entraîner. Comme le témoigne la figure 1 on modifie uniquement la couche de classification puis on entraîne le modèle intégralement.

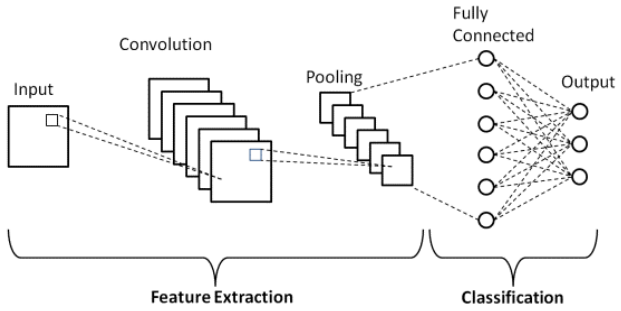

Figure 1 : Architecture typique d'un réseau de neurones convolutifs

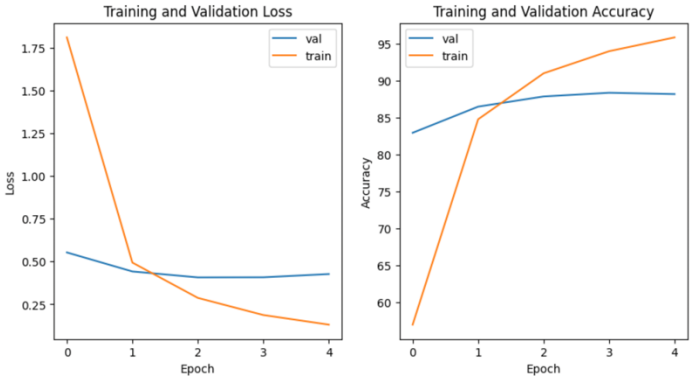

Figure 2 : Perte et exactitude pendant l'entrainement de EfficientNetV2_M sur CIFAR-100

3 Expériences

Dans cette section on présente les expériences réalisées et la configuration d'entraînement utilisée

3.1 Expériences et hyperparamètres

Dans le cadre du projet, de nombreuses expériences ont été réalisés avant de choisir la configuration d'entraînements que nous allions choisir pour nos cinq différents CNNs. Par souci d'efficacité toutes nos expériences exploratoires furent menées uniquement sur le modèle VGG-16 avant de décider s'il était nécessaire de la reproduire sur les autres modèles.

Dans un premier temps nous avons décidé d'utiliser les techniques d'augmentation de données afin d'obtenir un très haut niveau de précision. Cependant on constate que cela augmente considérablement le temps d'entraînement requis pour simplement une légère amélioration des performances, on décide donc de ne pas poursuivre cette expérience.

La seconde expérience menée fut d'essayer d'implémenter entièrement VGG-16 et l'entrainer avec des poids initialisés aléatoirement pour voir le temps d'entraînement requis. Cette expérience a été très longue et a été abandonné.

Les Hyperparamètres

Pour comparer les performances de nos différents modèles on a choisi d'utiliser les mêmes hyperparamètres pour tous. On aurait pu décider de choisir pour chaque CNN les hyperparamètres avec lequel on obtient les meilleurs résultats, mais cela aurait nécessité plus de temps car il aurait fallu trouver la configuration optimale pour chaque. Ainsi le tableau 1 présente les valeurs pour chaque hyperparamètre.

Hyperparamètre	Valeur
Taille du lot (batch size)	32
Nombre d'époches	5
Taux d'apprentissage	0.001
Momentum	0.9
Diminution des poids (weight decay)	0.0005

Tableau 1 : Hyperparamètres utilisés

Algorithme d'apprentissage et fonction de perte

Nous avons décidé d'utiliser l'algorithme de descente stochastique du gradient pour entraîner notre modèle. Quant à la fonction de perte nous avons fait appel à la fonction de perte d'entropie croisée.

4 Résultats

On présente dans cette section les performances de nos cinq différents CNN sur les jeux de données CIFAR-10 et CIFAR-100.

CNN	CIFAR-10 (%)	CIFAR-100 (%)
VGG-16	92.1	73.7
EfficientNetV2_M	**97.7**	**87.6**
EfficientNet-B0	96.1	81.7
ResNet-50	96.1	82.3
DenseNet-121	96.4	82.1

Tableau 2 : Exactitude des CNNs sur CIFAR-10 et CIFAR-100

Dans le tableau 2, les résultats montrent que les cinq modèles CNN sont capables de classifier les images avec une haute précision sur les deux ensembles de données CIFAR. Sans surprise tous les modèles ont une meilleure performance sur CIFAR-10 que CIFAR-100. Cela s'explique par le fait que CIFAR-100 est un plus grand jeu de données et possèdes 10 fois plus de classes d'images que CIFAR-10.

Le modèle EfficientNetV2_M est celui qui obtient les meilleurs résultats sur les deux ensembles de données, avec une exactitude de 97,7 % sur CIFAR-10 et 87,6 % sur CIFAR-100. EfficientNet-B0, ResNet-50 et DenseNet-121 obtiennent également des résultats similaires, avec des scores supérieurs à 96 % sur CIFAR-10 et supérieurs à 81 % sur CIFAR-100. Le modèle VGG-16 est le moins performant des cinq avec une exactitude de 92,1 % sur CIFAR-10 et 73,7 % sur CIFAR-100.

La figure 2 présente l'exactitude et la perte moyenne sur le jeu de données d'entraînement, et sur celui de validation pour EfficientNetV2_M et sur CIFAR-100. À l'aide de ce graphique on peut apercevoir qu'à partir d'un certain point (epoch 3 ici) la perte sur l'ensemble de validation de décroit plus et va à l'inverse commencer à croître. C'est donc à ce moment que le modèle est le meilleur, on s'est donc assurer de sauvegarder le meilleur modèle pour évaluer les performances sur l'ensemble de test. Bien évidemment on a effectué cette procédure pour les autres CNNs également.

Figure 3 : Matrice de confusion pour VGG-16 sur CIFAR-10

La figure 3 représente une matrice de confusion des prédictions effectuées par VGG-16 sur le jeu de données CIFAR-10. Cette représentation est utile pour repérer une anomalie dans les résultats. Dans ce cas-ci rien d'anormal a lieu, on remarque cependant que le modèle ce trompe relativement beaucoup entre les chats et les chiens. Il a tendance à plus prédire des chats alors que ce sont des chiens qu'inversement.

Les figure 2 et 3 sont des exemples, mais ces analyses ont été réalisés sur tous les autres CNNs également, et les graphiques sont tous disponible à l'adresse suivante : https://github.com/sanmarsimon/CNNs-CIFAR10-100

5 Discussion

Ce projet nous a permis d'étudier en détail et en profondeur le fonctionnement de plusieurs réseaux de neurones convolutifs différents. Les CNNs sont un pilier en termes de vision par ordinateur, et font l'objet de nouvelles recherches, et de nouvelle architecture voient le jour en permanence. De plus en plus ces nouvelles architectures utilisent un grand nombre de paramètre, et peuvent demander d'importantes ressources à entraîner. C'est pour cela le fin ajustement de modèles déjà pré-entraînés est une solution très intéressante car on l'a vu permet d'obtenir des très bons résultats avec beaucoup moins de temps d'entraînement. À l'issue de nos expériences on a pu constater que le modèle le plus récent parmi les cinq (EfficientNetV2_M) est celui qui obtient les meilleurs résultats.

Dans le futur il pourrait être intéressant de comparer les résultats obtenus durant ce projet à de nouvelles expériences mettant en place l'apprentissage par transfert. L'apprentissage par transfert consiste à simplement entraîner les couches de classification d'un modèle pré-entraîné.

References

[He et al., 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren et Jian Sun. *Deep Residual Learning for Image Recogntition.* https://arxiv.org/abs/1512.03385

[Tan et Le, 2019] Mingxing Tan et Quoc V. Le. *EfficientNet : Rethinking Model Scaling for Convolutional Neural Network*. https://arxiv.org/abs/1905.11946

[Kumar et al., 2021] Kumar, K., Mhetre, A., Ratnaparkhi, G.S., Kamat, S.S. (2021). A Superfamily-wide Activity Atlas of Serine Hydrolases in Drosophila melanogaster. Biochemistry 60(16): 1312--1324.

Kadam, S. S., Adamuthe, A. C., & Patil, A. B. (2020). CNN model for image classification on MNIST and fashion-MNIST dataset. *Journal of scientific research*, *64*(2), 374-384.