12/12/2023

Convolutional Neural Network

Advanced Neural Net

Ramiqi Andi

Table des matières

1. Introduction

Les réseaux de neurones convolutifs (CNN) se distinguent comme un outil puissant et polyvalent.Ces architectures, inspirées par le fonctionnement des neurones dans le cerveau humain, ont démontré une efficacité remarquable, en particulier dans le traitement d'images et de vidéos. Leur capacité à extraire et à apprendre des caractéristiques hiérarchiques à partir de données brutes les rend idéales pour diverses applications, allant de la reconnaissance faciale à l'analyse médicale.

Dans ce contexte, l'utilisation du dataset CIFAR, un ensemble de données largement reconnu dans la communauté de l'apprentissage profond, se révèle être un excellent choix pour explorer et expérimenter avec les CNN. Le dataset CIFAR, composé d'images de petite taille réparties en plusieurs catégories, offre un terrain propice pour tester l'efficacité des CNN dans la classification d'images. Sa diversité et sa complexité sont des atouts pour évaluer la performance des modèles d'apprentissage automatique

L’ensemble de la mise en place du réseau sera effectué avec PyTorch, une bibliothèque d'apprentissage automatique réputée pour sa flexibilité et sa facilité d'utilisation. PyTorch est idéal pour construire et expérimenter avec des CNN, grâce à sa capacité à effectuer des calculs rapides et sa facilité de manipulation des réseaux de neurones.

L'objectif de ce rapport est de maîtriser l'utilisation de PyTorch et d’obtenir des meilleurs résultats sur le jeu de donnée cifar. Ce travail permettra non seulement de comprendre mieux les CNN, mais aussi d'améliorer l’apprentissage de la mise en place d’un réseau de neurones.

2. CNN

Les réseaux de neurones convolutifs sont une catégorie spécifique de réseaux de neurones utilisés principalement pour le traitement d'images. Ils sont conçus pour reconnaître automatiquement des motifs spatiaux dans les données, tels que les bords, les textures et les formes, en les analysant à travers de petites fenêtres, ou "filtres". Cette capacité les rend particulièrement adaptés aux tâches de vision par ordinateur, comme la reconnaissance faciale, la classification d'images et la détection d'objets.

Les couches

1. Couche convolutive :

Dans un réseau de neurones convolutif (CNN), l'idée principale est de faire passer une image en entrée à travers une série de filtres, appelés noyaux de convolution, pour en extraire des caractéristiques importantes et effectuer des prédictions. Chaque filtre agit un peu comme un détective spécialisé, cherchant des éléments spécifiques dans l'image - comme les bords, les angles ou les textures.

Si l'on prend l'exemple d'une photo de paysage, un filtre pourrait se concentrer sur la détection des lignes verticales, comme les troncs d'arbre, tandis qu'un autre pourrait chercher des courbes, comme celles des collines. En appliquant ces filtres, le CNN crée plusieurs "cartes de convolutions", qui sont essentiellement de nouvelles images mettant en évidence les caractéristiques détectées par chaque filtre.

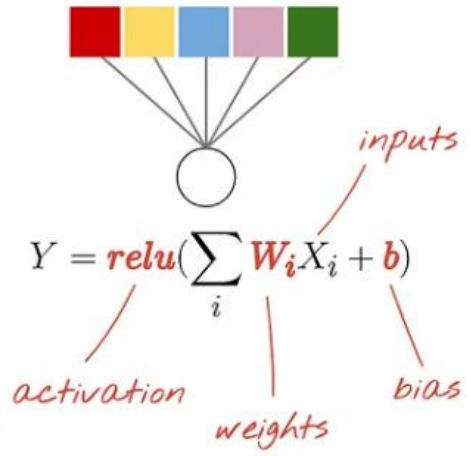
En faisant glisser une fenêtre représentant la « feature » sur l'image et en calculant le produit de la convolution entre la feature et chaque portion de l'image, il est possible d'obtenir ces filtres.

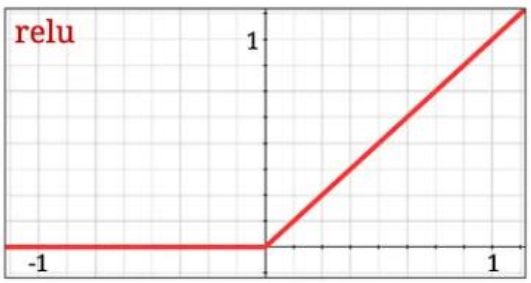
2. Couche d'activation

Après que l'image a été traitée par la couche convolutive et transformée en cartes de convolutions, elle passe ensuite à travers une couche d'activation non linéaire. L'une des fonctions d'activation les plus couramment utilisées dans les CNN est la Rectified Linear Unit, ou ReLU définie par :

$$ReLU(x) = max(0, x)$$

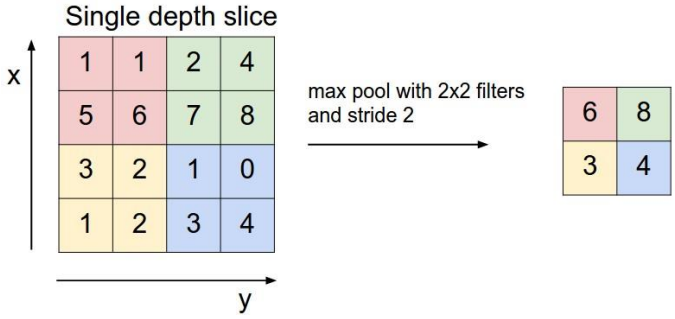
La fonction ReLU prend chaque valeur de la carte de convolutions et la transforme en 0 si la valeur est négative. En revanche, si la valeur est positive, elle reste inchangée. En éliminant ces valeurs négatives, cela introduit une non-linéarité nécessaire pour que le réseau apprenne efficacement des motifs complexes dans les données.





3. Couche de pooling :

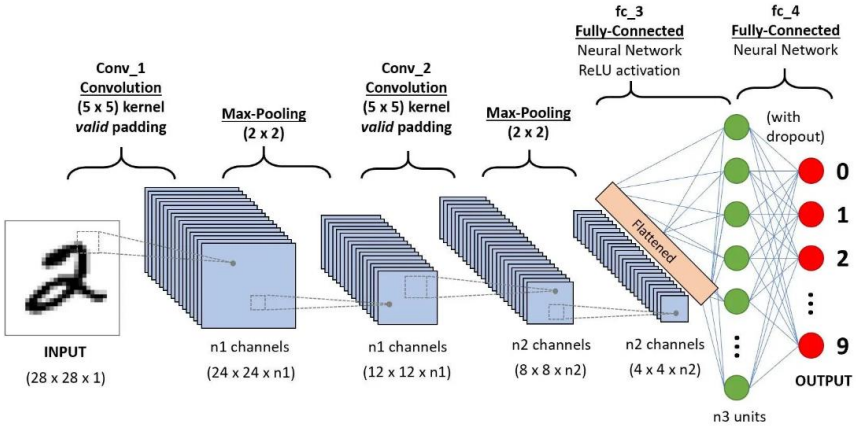
Ce type de couche est souvent placé entre deux couches de convolution et permet de réduire la dimensionnalité spatiale (hauteur et largeur) des cartes de caractéristiques, conservant les informations les plus importantes.



La mise en place de cette couche permet de réduire la quantité de paramètres et calcul dans le réseau et permettre de contrôler le sur-apprentissage.

4. Couches « Fully Connected)

Il s'agit de la dernière étape qui servira à utiliser les caractéristiques extraites des couches précédentes pour classer l'entrée dans différentes catégories. Cette couche reçoit un vecteur en entrée contenant les pixels aplatis de tous les images filtrées, corrigées et réduites par le pooling.

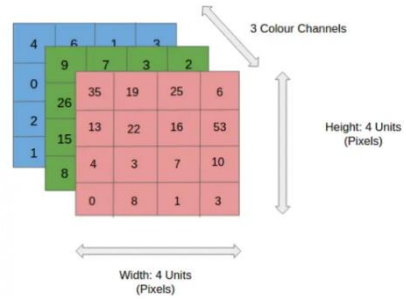


Résumé de la mise en place des différentes couches : traitement de l'image en input jusqu'à l'output permettant de prédire la classe avec les différentes couches entre le début et la fin.

Pour pouvoir mettre en place ces différentes, couches, certains critères tel que la gestion de l'image en input ainsi que la taille du filtre seront nécessaires pour la mise en place d'un CNN.

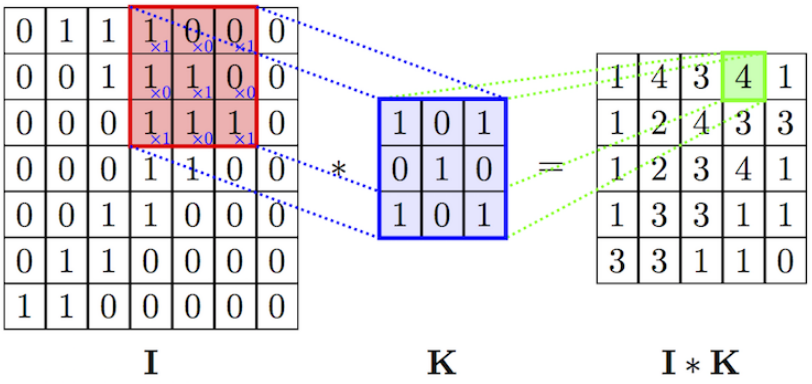
5. Image en input :

Une image est composée d'un certain nombre de pixel en largeur et longueur. De plus, l'image sera séparée par 3 panneaux (Rouge, Vert, Bleu) permettant de gérer la couleur de l'image.



6. Kernel

Il s'agit d'une petite matrice de poids utilisée pour applique une opération de convolution en se déplaçant sur l'image d'entrée. À chaque position, il effectue une opération de multiplication élément par élément entre les poids du kernel et les valeurs des pixels correspondants de l'image. Les résultats de ces multiplications sont ensuite additionnés pour obtenir une seule valeur. Ce processus est répété sur toute l'image, produisant ainsi une nouvelle matrice qui sera notre carte de caractéristiques (feature map), et qui capture les informations pertinentes extraites de l'image.



*La matrice I contient l'input (image) sur laquelle on applique une matrice K, le Kernel (filtre) afin d'obtenir une caractéristique et composer ensuite notre matrice I*K qui est la carte en déplaçant le filtre sur l'entièreté de l'image.*

Padding et Stride :

Padding

Dans les réseaux de neurones convolutifs (CNN), le padding (remplissage) et le stride (pas) sont deux paramètres cruciaux qui définissent comment une opération de convolution est appliquée aux données d'entrée

Le padding fait référence à l'ajout de pixels supplémentaires autour des bords de l'image d'entrée ou de la carte de caractéristiques. Ces pixels sont généralement réglés à zéro et sont connus sous le nom de zero-padding.

Le but principal du padding est de contrôler les dimensions spatiales des cartes de caractéristiques en sortie. Sans padding, la taille des cartes de caractéristiques diminue après chaque opération de convolution, ce qui peut conduire à une réduction significative de la taille de l'image au fur et à mesure qu'elle traverse le réseau. Le

padding permet de préserver les dimensions spatiales de l'entrée, permettant des réseaux plus profonds sans réduction rapide de taille.

Types de Padding :

- « Valid Padding » : Aucun padding n'est appliqué, et la convolution est effectuée uniquement sur les régions d'entrée valides.
- « Same Padding » : Assez de padding est appliqué pour s'assurer que la carte de caractéristiques en sortie a les mêmes dimensions spatiales que l'entrée.

Pour obtenir ainsi la même taille d'image en output qu'en input, la taille du padding doit respecter la formule suivante :

$$p = (f - 1) / 2$$

p représente le nombre de 0 qu'il faudra avoir autour de l'image en input, et il dépend de f, qui représente la taille de notre filtre.

Stride

Le stride fait référence au nombre de pixels par lesquels le noyau se déplace sur l'image d'entrée ou la carte de caractéristiques pendant l'opération de convolution.

Le stride contrôle le chevauchement entre les champs récepteurs et la taille de la carte de caractéristiques en sortie. Un stride plus grand entraîne moins de chevauchement et une dimension de sortie plus petite, effectuant ainsi un sous-échantillonnage de l'image.

- Un stride de 1 (commun dans de nombreux CNN) déplace le noyau d'un pixel à la fois, préservant ainsi une grande partie de l'information spatiale.
- Un stride plus grand, comme 2 ou plus, saute des pixels et réduit la dimensionnalité de la carte de caractéristiques en sortie.

Les différences :

- Contrôle de la Dimensionnalité : Bien que le padding et le stride influencent tous deux les dimensions de la carte de caractéristiques en sortie, ils le font de manières différentes. Le padding est utilisé pour augmenter ou préserver les dimensions, tandis que le stride est utilisé pour les réduire.
- Extraction de Caractéristiques : Le padding permet une meilleure extraction des informations sur les bords de l'image. Sans padding, les informations aux frontières pourraient être sous-représentées.
- Efficacité Computationnelle : Le stride peut être utilisé pour réduire la charge computationnelle en réduisant la taille des cartes de caractéristiques, mais cela peut également entraîner une perte de résolution spatiale et d'information.
- Champ de Vision : Augmenter le stride augmente le champ de vision de chaque opération de convolution, permettant au réseau de capturer des informations plus globales avec moins de couches. Cependant, cela se fait au détriment des informations locales détaillées.

Limitations :

Bien que puissants, les CNN ont certaines limitations :

- Nécessité de Grandes Quantités de Données : Pour bien fonctionner, les CNN requièrent de grandes quantités de données d'entraînement.
- Sensibilité aux Variations : Les CNN peuvent être sensibles aux variations dans les images, comme l'éclairage, l'angle de vue et la déformation.
- Complexité Computationnelle : Les CNN avec de nombreuses couches et filtres peuvent être computationnellement coûteux à entraîner et à utiliser.
- Manque de Compréhension Contextuelle : Les CNN sont très bons pour identifier des motifs locaux mais peuvent manquer de compréhension du contexte global de l'image.

3. Mise en place pratique

Afin de mettre en place le CNN, il a fallu dans un premier temps définir le model.

Architecture du modèle[1] :

Première Couche Convolutionnelle (conv1) :

- Cette couche utilise nn.Conv2d pour créer une convolution 2D.
- Elle prend 3 canaux en entrée (pour les images RGB) et produit 32 canaux en sortie.
- Le kernel de taille 3x3 (le troisième argument de Conv2d) est utilisé avec un padding de 1 et un stride de 1. Le padding permet de préserver la dimension spatiale de l'image.

Deuxième Couche Convolutionnelle (conv2) :

- Similaire à conv1, mais prend 32 canaux en entrée (le nombre de canaux en sortie de conv1) et produit 64 canaux en sortie.
- Utilise également un kernel de taille 3x3 avec un padding de 1 et un stride de 1.

Couche de Pooling (pool) :

- Une couche de pooling est utilisée pour réduire la dimension spatiale des données (sous-échantillonnage).
- Un kernel de pooling de 2x2 avec un stride de 2 est utilisé, réduisant de moitié les dimensions de hauteur et de largeur.

Couches Entièrement Connectées (fc1 et fc2) :

- fc1 transforme les caractéristiques extraites en vecteurs de taille 512. La dimension d'entrée (8 * 8 * 64) doit correspondre à la taille des cartes de caractéristiques après les couches convolutives et de pooling.

[1] https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html Exemple sur lequel le model a été basé

- fc2 réduit la dimension de 512 à 10, correspondant au nombre de classes dans CIFAR-10.

Fonction Forward :

- Les données d'entrée x traversent les couches dans l'ordre défini.
- Après chaque couche convolutive (conv1 et conv2), la fonction d'activation ReLU est appliquée, suivie d'une opération de pooling.
- Avant d'entrer dans la première couche entièrement connectée (fc1), les données sont aplaties.
- La fonction d'activation ReLU est également appliquée après fc1.
- La dernière couche (fc2) produit la sortie finale du réseau, qui peut être utilisée pour la classification.

Entrainement des données

Le modèle de réseau de neurones convolutifs (CNN) a été entraîné pour apprendre à reconnaître et à classer des images. Tout au long de l'entraînement, le modèle a été exposé à une variété d'images, lui permettant d'apprendre progressivement à identifier leurs caractéristiques et motifs. À des intervalles réguliers, sa performance a été évaluée sur des ensembles de données différents de ceux de l'entraînement, afin de s'assurer qu'il pouvait efficacement traiter de nouvelles images qu'il n'avait pas encore rencontrées. Ce processus d'entraînement répété a permis au modèle d'améliorer continuellement ses capacités de distinction et de classification des images.

Hyperparamètres

Les hyperparamètres clés qui ont été définis sont les suivants :

- Epochs = 15 : Le nombre total d'itérations sur l'ensemble des données d'entraînement.
- batch_size = 256 : La taille des lots de données traités dans une seule itération de l'entraînement.
- eval_every = 3 : La fréquence à laquelle le modèle est évalué sur l'ensemble de validation, en termes d'époques.
- evaluate_testset_during_training = True : Un booléen indiquant si l'ensemble de test doit être évalué pendant l'entraînement.
- Device = torch.device("cuda" if torch.cuda.is_available() else "cpu") : Le dispositif sur lequel le modèle est entraîné (GPU si disponible, sinon CPU).
- Lr = 0.001 : Le taux d'apprentissage pour l'optimiseur Adam[2].

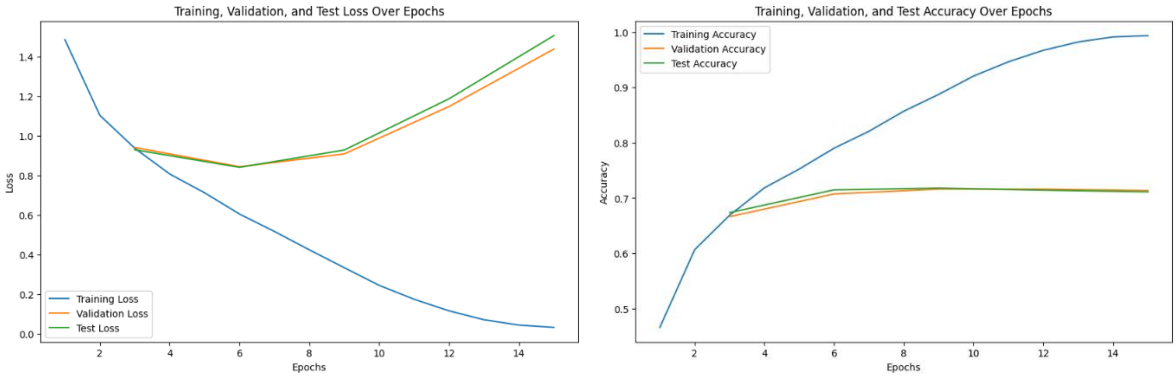
[2] L'optimiseur Adam a été laissé, mais les résultats entre lui et SGD ne sont pas vraiment différent, celui-ci a été priorisé simplement pour sa convergence rapide.

4. Analyse des résultats

```
Final Training Accuracy: 99.38%
Final Validation Accuracy: 71.35%
Final Test Accuracy: 71.11%
```

Les résultats obtenus montrent une précision d'entraînement exceptionnellement élevée de 99.38%, indiquant que le modèle a très bien appris et adapté ses paramètres aux données d'entraînement. Cependant, la précision sur les ensembles de validation et de test, étant respectivement de 71.35% et 71.11%, révèle un décalage significatif par rapport à la performance en entraînement. Cette disparité suggère un surapprentissage, où le modèle excelle à reconnaître les images qu'il a déjà vues mais peine à généraliser ses apprentissages à de nouvelles données. Ce phénomène peut être dû à une complexité excessive du modèle par rapport à la diversité des données d'entraînement, ou à un entraînement trop long permettant au modèle de mémoriser les détails spécifiques des images d'entraînement au lieu de saisir des caractéristiques généralisables.

Il est tout de même important de noter que dans le cas du jeu de donné CIFAR-10, la probabilité de déterminer une image au hasard est de 10%. Lors de la mise en place d'un MLP qui essayait de déterminer la classe avec une fonction Softmax (ancien TP), la précision n'était que de 30% (un peu moins même…), ce qui montre une amélioration très significative avec un model plus puissant qu'est le CNN lorsqu'on voir qu'on atteint une précision de 71%.



L'image de gauche représente un graphique de la perte (loss) d'entraînement, de validation et de test au fil des époques pour le modèle.

Le graphique illustre clairement la divergence entre la perte d'entraînement et celle de validation et de test à mesure que le nombre d'époques augmente. Au début, la perte d'entraînement diminue rapidement, ce qui indique que le modèle apprend et s'améliore en fonction des données d'entraînement. Cependant, après un certain point, la perte de validation et de test commence à s'écarter de la perte d'entraînement et même à augmenter, ce qui est un signe classique de surapprentissage.

Le graphique de droite montre simplement la précision du model qui atteint très vite le pourcentage max et qui semble stagner au fur et à mesure des époques.

```
Accuracy for class: plane = 72.00 %
Accuracy for class: car = 82.00 %
Accuracy for class: bird = 65.20 %
Accuracy for class: cat = 59.00 %
Accuracy for class: deer = 63.70 %
Accuracy for class: dog = 56.90 %
Accuracy for class: frog = 77.90 %
Accuracy for class: horse = 74.30 %
Accuracy for class: ship = 84.50 %
Accuracy for class: truck = 79.20 %
```

Lorsqu'on se penche sur la précision du modèle par classe du jeu de donnée CIFAR-10, il est possible de remarquer plusieurs éléments. La précision par classe varie considérablement, allant de 56.9% pour la classe 'dog' à 84.5% pour la classe 'ship', ce qui indique des performances inégales du modèle sur ces différentes catégories. Les classes 'cat' et 'dog' ont les précisions les plus basses, ce qui peut suggérer que le modèle a du mal à distinguer les caractéristiques distinctives de ces animaux, qui d'un point de vue de leur forme montre une certaine vraisemblance.

En revanche, des classes comme 'car' et 'ship' ont des précisions relativement élevées, ce qui peut indiquer que le modèle reconnaît plus facilement leurs caractéristiques uniques ou que les images dans ces catégories sont plus homogènes et donc plus faciles à apprendre. La caractéristique la plus probable reste sûrement la forme de ces objets qui est très différenciable des autres classes.

Sources

A Guide to Convolutional Neural Networks—The ELI5 way | Saturn Cloud Blog. (2018, décembre 15). https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/

CNN. (s. d.). prezi.com. Consulté 13 décembre 2023, à l'adresse https://prezi.com/p/1qicuo-nnp8j/cnn/

CNN | Introduction to Padding. (2019, juillet 23). *GeeksforGeeks*. https://www.geeksforgeeks.org/cnn-introduction-to-padding/

deeplizard (Réalisateur). (2017, décembre 9). *Convolutional Neural Networks (CNNs) explained*. https://www.youtube.com/watch?v=YRhxdVk_sIs

Obam, Y. S. (2020, septembre 2). Comprendre les Réseaux de Neurones Convolutifs (CNN). *Medium*. https://yannicksergeobam.medium.com/comprendre-les-r%C3%A9seaux-de-neurones-convolutifs-cnn-d5f14d963714

Papers with Code—CIFAR-100 Benchmark (Image Classification). (s. d.). Consulté 13 décembre 2023, à l'adresse https://paperswithcode.com/sota/image-classification-on-cifar-100

Training a Classifier—PyTorch Tutorials 2.2.0+cu121 documentation. (s. d.). Consulté 13 décembre 2023, à l'adresse https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

What is Adam? What's the main difference between Adam and SGD? (s. d.). CodeAhoy. Consulté 13 décembre 2023, à l'adresse https://codeahoy.com/questions/ds-interview/33/