PSB Paris School of Business

Réseau neuronal convolutif (Convolutional neural network)

Hakim Daif

25 janvier 2021

Table des matières

1 Introduction

les notions de features en vision et les méthodes traditionnellement utilisées pour faire de la classification d'images consistent à extraire les features de chaque image du jeu de données, puis à entraîner un classifieur sur ces features.

Ces techniques d'apprentissage supervisé peuvent fournir de très bons résultats, et leur performance dépend fortement de la qualité des features préalablement trouvées. Il existe plusieurs méthodes d'extraction et de description de features.

En pratique, l'erreur de classification n'est jamais nulle. Les résultats peuvent alors être améliorés en créant de nouvelles méthodes d'extraction de features, plus adaptées aux images étudiées, ou en utilisant un "meilleur" classifieur.

Mais en 2012, une révolution se produit : lors de la compétition annuelle de vision par ordinateur ILSVRC, un nouvel algorithme de Deep Learning explose les records ! Il s'agit d'un réseau de neurones convolutif appelé AlexNet.

Les réseaux de neurones convolutifs ont une méthodologie similaire à celle des méthodes traditionnelles d'apprentissage supervisé : ils reçoivent des images en entrée, détectent les features de chacune d'entre elles, puis entraînent un classifieur dessus.

Cependant, les features sont apprises automatiquement ! Les CNN réalisent eux-mêmes tout le boulot fastidieux d'extraction et description de features : lors de la phase d'entraînement, l'erreur de classification est minimisée afin d'optimiser les paramètres du classifieur ET les features. De plus, l'architecture spécifique du réseau permet d'extraire des features de différentes complexités, des plus simples au plus sophistiquées. L'extraction et la hiérarchisation automatiques des features, qui s'adaptent au problème donné, constituent une des forces des réseaux de neurones convolutifs : plus besoin d'implémenter un algorithme d'extraction "à la main", comme SIFT ou Harris-Stephens.

Contrairement aux techniques d'apprentissage supervisé, les réseaux de neurones convolutifs apprennent les features de chaque image. C'est là que réside leur force : les réseaux font tout le boulot d'extraction de features automatiquement, contrairement aux techniques d'apprentissage.

Aujourd'hui, les réseaux de neurones convolutifs, aussi appelés CNN ou ConvNet pour Convolutional Neural Network, sont toujours les modèles les plus performants pour la classification d'images.

2 Convolution

Dans cette partie nous allons présenter le concept de la convolution d'une façon plus théorique :

2.1 Concept mathématique

Soient $(f(n))_{n\in\mathbb{Z}}$ et $(g(n))_{n\in\mathbb{Z}}$ deux suites de nombres réels. Le produit de convolution $f \star g$ est la suite $(h(n))_{n\in\mathbb{Z}}$ dont le terme général est défini par :

$$f \star g(n) = \sum_{k=-\infty}^{+\infty} f(n-k) \cdot g(k)$$

En particulier, dans les situations rencontrées ici, il n'y a pas Vraiment une infinité de termes à calculer. Voici une formule plus simple, lorsque l'on suppose que les termes de g sont nuls en dehors des indices appartenant à $[-K, +K]$:

$$f \star g(n) = \sum_{k=-K}^{+K} f(n-k) \cdot g(k)$$

Le résultat de la convolution est la suite : $[\ldots, 0, 0, 1, 2, 6, 11, 20, 17, 3, 0, 0, \ldots]$

où le terme de rang 0 est $f \star g(0) = 2$

Ce résultat est indépendant, à un décalage près, des choix opérés dans les définitions de f et g.

La convolution tronquée consiste à ne retenir qu'une sous-liste de même taille que la liste d'entrée en tronquant les bords (la façon de tronquer dépend du choix des définitions de f et g).

Ainsi : [1,0,3,5,1]$\star$ $[1, 2, 3] = [2, 6, 11, 20, 17]$.

Reprenons l'exemple de la convolution de [1,0,3,5,1] par [1,2,3]. Choisissons $(f(n))_{n\in\mathbb{Z}}$ la suite dont les termes non tous nuls sont $[f(0) = 1, f(1) = 0, f(2) = 3, f(3) = 5, f(4) = 1]$ et choisissons $(g(n))_{n\in\mathbb{Z}}$ la suite dont les termes non tous nuls sont $[g(-1) = 1, g(0) = 2, g(1) = 3]$. Alors on obtient la convolution $h = f \star g$ par la formule ci-dessus :

$$f\widetilde{\star}g(n) = \sum_{k=-1}^{+1} f(n-k) \cdot g(k) = f(n-1)g(1) + f(n)g(0) + f(n+1)g(-1)$$

On retrouve la méthode pratique vu précédemment : on renverse le motif g avant de faire une série de multiplications.

Parexemple

$f \star g(1) = f(0)g(1) + f(1)g(0) + f(2)g(-1) = 1 \times 3 + 0 \times 2 + 3 \times 1 = 6$

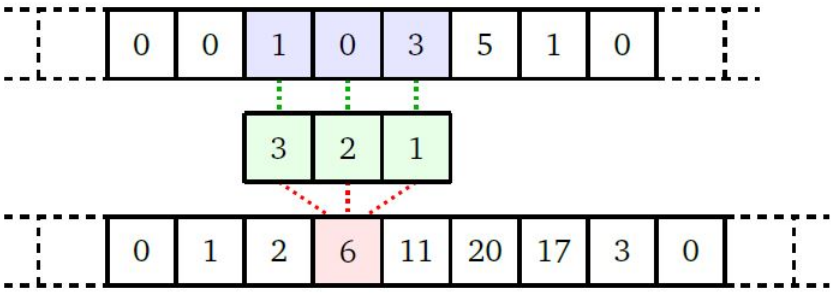


FIGURE 1 – Convolution de [1, 0, 3, 5, 1] par [1, 2, 3]

Le résultat de la convolution est la suite

$$[\ldots, 0, 0, 1, 2, 6, 11, 20, 17, 3, 0, 0, \ldots]$$

où le terme de rang 0 est $f\widetilde{\star}g(0) = 2$. Ce résultat est indépendant, à un décalage près, des choix opérés dans les définitions de f et g.
La convolution tronquée consiste à ne retenir qu'une sous-liste de même taille que la liste d'entrée en tronquant les bords (la façon de tronquer dépend du choix des définitions de f et g). Ainsi : [1,0,3,5,1]$\star$ $[1, 2, 3] = [2, 6, 11, 20, 17]$

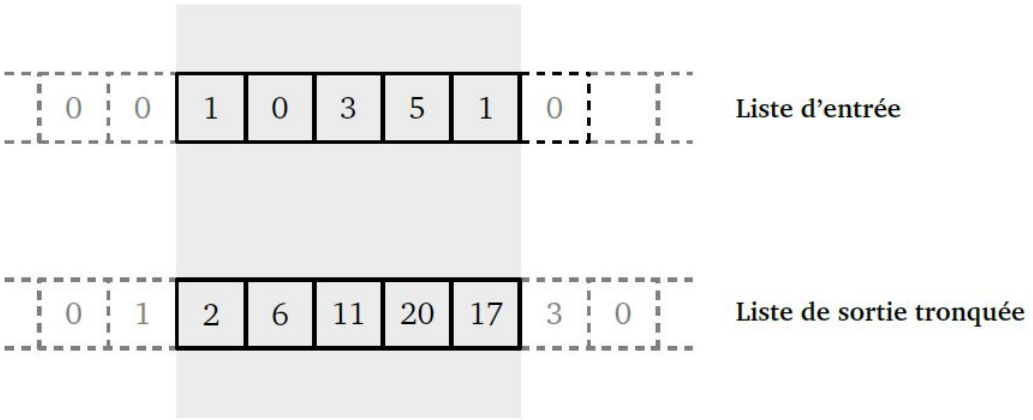


FIGURE 2 – Convolution de [1, 0, 3, 5, 1] par [1, 2, 3]

Terminons par une propriété remarquable du produit de convolution. En réindexant la

somme de la définition, On obtient :

$$f \star g(n) = \sum_{k=-\infty}^{+\infty} f(k) \cdot g(n-k)$$

Ce qui signifie que le produit de convolution est symétrique :

$$f \star g(n) = g \star f(n)$$

Remarque :
- La convolution est très utilisée en théorie du signal, c'est pourquoi on trouve aussi le vocabulaire signal d'entrée, signal de sortie. C'est aussi pour des raisons de signal qu'on renverse une des listes avant de faire les multiplications.
- Le motif s'appelle aussi noyau (kernel en anglais) mais aussi masque ou filtre.

2.2 Convolution d'une matrice

La convolution est une opération mathématique simple sur un tableau de nombres, une matrice ou encore une image afin d'y apporter une transformation ou d'en tirer des caractéristiques principales.

2.2.1 Vocabulaire (Matrice)

Nous allons voir ou revoir quelques notions (le minimum vital) sur les matrices.

Une matrice est un tableau de n lignes et p colonnes, contenant des nombres réels.

$$n\ lignes \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1p} \\ a_{21} & a_{22} & \dots & a_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{np} \end{pmatrix} pcolonnes$$

Lorsque il y a autant de lignes que de colonnes, on parle de matrice carrée. Voici une matrice 3×3 :

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

Addition : On peut additionner deux matrices de même taille. L'addition se fait terme à terme. Exemple :

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} + \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} = \begin{pmatrix} 1 & 3 & 3 \\ 5 & 1 & 7 \\ 7 & 9 & 9 \end{pmatrix}$$

Multiplicationpar un scalaire : On peut multiplier une matrice par un nombre réel. Exemple :

$$3 \cdot \begin{pmatrix} 1 & 2 & 3 & 4 \\ 8 & 7 & 5 & 6 \end{pmatrix} = \begin{pmatrix} 3 & 6 & 9 & 12 \\ 24 & 21 & 15 & 18 \end{pmatrix}$$

Multiplication : On peut multiplier deux matrices. Nous ne donnons pas ici la définition, mais la multiplication n'est pas effectuée terme à terme. Voici, sans entrer dans les détails, un exemple pour des matrices carrées :

$$\begin{pmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{pmatrix} \times \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} = \begin{pmatrix} -6 & -9 & -12 \\ 10 & 11 & 12 \\ 18 & 21 & 24 \end{pmatrix}$$

Nous définirons un peu plus loin la convolution qui est un nouveau type de multiplication de matrices.

2.3 Convolution (deux dimensions)

Définition
La convolution en deux dimensions est une opération qui :
- à partir d'une matrice d'entrée notée A,
- et d'une matrice d'un motif noté M associe une matrice de sortie $A \star M$. Tout d'abord, il faut retourner la matrice M :

$$\begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix} \quad \begin{pmatrix} m_{33} & m_{32} & m_{31} \\ m_{23} & m_{22} & m_{21} \\ m_{13} & m_{12} & m_{11} \end{pmatrix}$$

Motif de taille 3 $\times$ 2 Motif retourné

Le calcul de $A \star M$ s'effectue coefficient par coefficient :
- on centre le motif retourné sur la position du coefficient à calculer,
- on multiplie chaque coefficient de A par le coefficient du motif retourné en face (quitte à ajouter des zéros virtuels sur les bords de A),
- la somme de ces produits donne un coefficient de $A \star M$.

Voici un autre schéma pour le calcul d'un autre coefficient. Pour ce calcul, on rajoute des zéros virtuels autour de la matrice A.

Exemple :
Calculons la convolution $A \star M$ définie par :

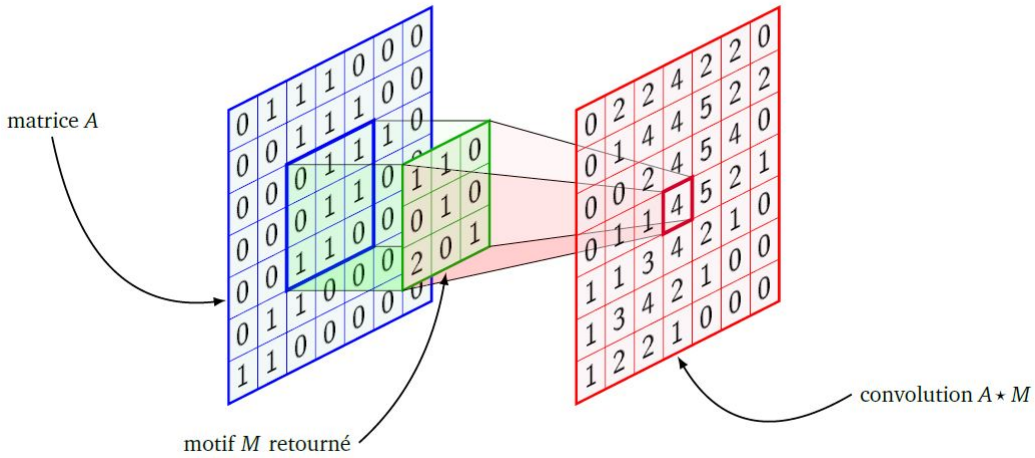


FIGURE 3 – Calcul A*M

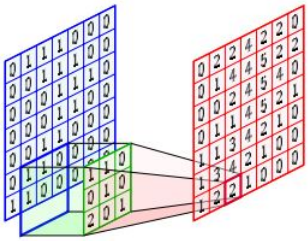


FIGURE 4 – Calcul A*M

$$\begin{pmatrix} 2 & 1 & 3 & 0 \\ 1 & 1 & 0 & 5 \\ 3 & 3 & 1 & 0 \\ 2 & 0 & 0 & 2 \end{pmatrix} \star \begin{pmatrix} 1 & 0 & 2 \\ 2 & 1 & 0 \\ 1 & 0 & 3 \end{pmatrix}$$

On commence par retourner M. Pour calculer le premier coefficient de $A \star M$, on centre le motif sur le premier coefficient de A, puis on rajoute des zéros virtuels à gauche et en haut. Ensuite on calcule le produit de chaque coefficient de la matrice M retournée avec les coefficients de A correspondants, et on les additionne :

$$0 \times 3 + 0 \times 0 + 0 \times 1 + 0 \times 0 + 2 \times 1 + 1 \times 2 + 0 \times 2 + 1 \times 0 + 1 \times 1$$

Cette somme vaut 5, c'est le premier coefficient de $A \star M$

Remarque :
- Dans la plupart de nos situations, le motif sera une matrice de taille 3×3. Par contre la matrice A pourra être de très grande taille (par exemple une matrice 600×400, codant une image).@
- Cependant, si la matrice du motif a une dimension paire on rajoute des zéros virtuels à

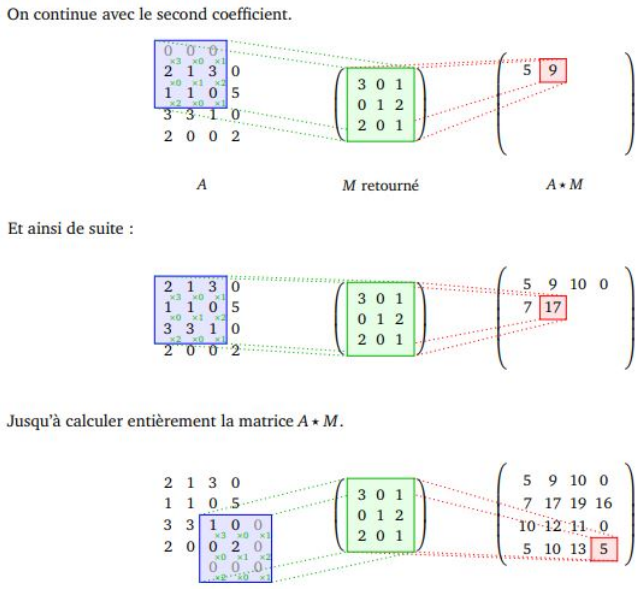


FIGURE 5 – Calcul A*M

gauche ou en haut (avant de le retourner).

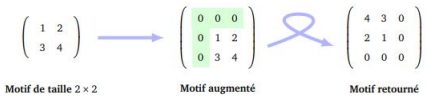


FIGURE 6 – Calcul A*M

Contrairement au pooling, les motifs disposés pour des calculs d'éléments voisins se superposent. Sur le dessin ci-dessous le motif est d'abord centré sur le coefficient 8 , puis sur le coefficient 9.

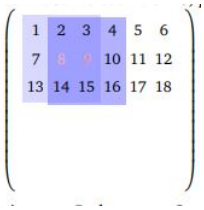


FIGURE 7 – Calcul A*M

Par définition, il faut retourner la matrice M avant de calculer les produits. Cela complique l'usage mais est cohérent avec la définition donnée pour la dimension 1 et cela permettra d'avoir de bonnes propriétés mathématiques

3 Pooling

3.1 Définition Mathématique

Le pooling (regroupement de termes) consiste à transformer une matrice en une matrice plus petite tout en essayant d'en garder les caractéristiques principales. Un pooling de taille k transforme une matrice de taille $n \times p$ en une matrice de taille k fois plus petite, c'est-à-dire de taille $n//k \times p//k$. Une sous-matrice de taille $k \times k$ de la matrice de départ produit un seul coefficient de la matrice d'arrivée.

L'opération de pooling (appelée aussi sous-échantillonnage) est un élément clef pour les réseaux de neurones convolutifs, car, elle permet d'améliorer le processus d'apprentissage du réseau et d'éviter le sur-apprentissage en réduisant progressivement les tailles des cartes de caractéristiques obtenues durant les couches de convolution tout en gardant les informations les plus pertinentes.

$$\left(\begin{array}{cc|cc|cc} a_1 & a_2 & b_1 & b_2 & c_1 & c_2 \\ a_3 & a_4 & b_3 & b_4 & c_3 & c_4 \\ \hline d_1 & d_2 & \cdot & \cdot & \cdot & \cdot \\ d_3 & d_4 & \cdot & \cdot & \cdot & \cdot \end{array}\right) \quad \text{pooling de taille 2} \quad \left(\begin{array}{c|c|c} \alpha & \beta & \gamma \\ \hline \delta & \cdot & \cdot \end{array}\right)$$

matrice de taille 2×3 matrice de taille 4×6

Autrement dit, La couche de pooling (en anglais pooling layer) (POOL) est une opération de sous-échantillonnage typiquement appliquée après une couche convolutionnelle. En particulier, les types de pooling les plus populaires sont le max et l'average pooling, où les valeurs maximales et moyennes sont prises, respectivement.

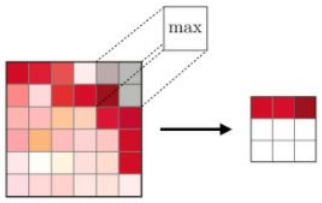
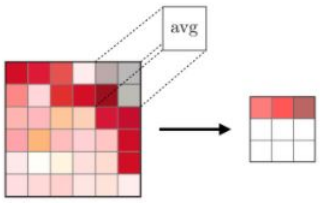
Type	Max pooling	Average pooling
But	Chaque opération de pooling sélectionne la valeur maximale de la surface	Chaque opération de pooling sélectionne la valeur moyenne de la surface
Illustration		
Commentaires	• Garde les caractéristiques détectées • Plus communément utilisé	• Sous-échantillonne la *feature map* • Utilisé dans LeNet

FIGURE 8 – Convolution de [1, 0, 3, 5, 1] par [1, 2, 3]

Le max-pooling de taille k consiste à retenir le maximum de chaque sous-matrice de taille $k \times k$.
Le pooling en moyenne de taille k (average pooling) consiste à retenir la moyenne des termes de chaque sous-matrice de taille $k \times k$

4 Fonctions d'activation

Le processus d'implémentation d'un réseau de neurones nécessitent la construction de plusieurs fonctions dont chacune effectue une tâche bien spécifique.
Souvent, beaucoup d'attention est apportée sur l'écriture des fonctions en lien avec l'apprentissage du réseau : le calcul des gradients (Back-Propagation), l'algorithme d'optimisation (Descente de gradient ou autres) et l'initialisation des poids des connexions entre les neurones. En allant plus dans le détail sur les plus petites pièces du puzzle, on se rend compte que ces derniers sont aussi importants que les fonctions citées plus haut (ou peut être même plus importants). Dans cette partie nous allons essayer d'éclaircir le fonctionemment des fonctions d'activation d'un point de vu mathématique et puis on va essyer de définir leur role important dans le traitement d'images.

Qu'est-ce qu'une fonction d'activation (à travers son utilité) ?

Une fonction d'activation sert à mimer la manière dont les neurones du cerveau traitent les signaux électriques qu'ils reçoivent. En termes mathématiques, on peut l'appeler une fonction de transfert qui transforme une entrée par une sortie à travers une fonction.

4.1 Définition

Téoriquement la fonction ou la règle d'activation c'est :

Chaque neurone $i \in [1,\ \mathrm{N}]$ possède une règle f_i appelée aussi fonction d'activation. En général, elle est identique à tous les neurones du réseau ou ceux d'une même "couche", notée alors simplement f. Elle s'applique sur le résultat de h_i la fonction d'entrée, c'est-à-dire sur l'activation pondérée e_i. A l'instant $t > 0$, nous avons :

$$a_i(t) = f_i\left(e_i(t-1), \Theta_i\right)$$

La fonction d'activation peut être :
- à valeurs continues ou discrètes (pour les VLSI)
- déterministe ou stochastique (différentes réponses selon une distribution de probabilité donnée)
- à mémoire ou sans mémoire.

Ses caractéristiques sont :
- monotonie : la fonction d'activation f est en général croissante ;
- seuillage : pour la résistance au bruit le neurone i est affecté d'un seuil Θ_i; si $f(x) < \Theta_i$ alors $f(x)$ est négligeable ;
- saturation pour éviter de propager de grandes valeurs : si $f(x) > \mathrm{M}$ alors $a_i = \mathrm{M}$
- dérivabilité, pour l'apprentissage, par exemple.

4.2 Types de fonction d'activation

Quelles sont les différents types de fonction d'activation ? il y existe plusieurs types de fonctions d'activation :

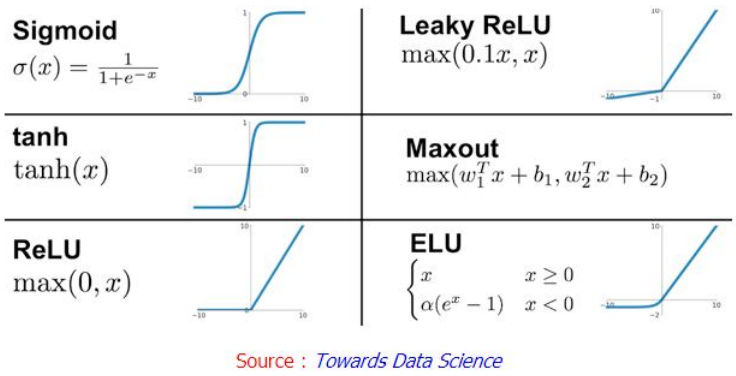


FIGURE 9 – Types des fonctions d'activation

Nous allons à présent présenter quelques-unes :

Rappelons que h est la fonction d'entrée (soit disant d'un neurone artificiel) i et Θ_i son éventuel seuil. Nous décrivons ci-dessous les fonctions d'activation les plus utlilisée.

Exemple 1 : Fonction linéaire
$f(x) = \lambda x$. D'où $a_i = \lambda h(i)$.

Exemple 2 : Fonction de seuil ou fonction de signe
Soit un seuil Θ.

$$f(x) = \begin{cases} 1 & \text{si } x \geq \Theta \\ 0 & \text{sinon} \end{cases}$$

Exemple 3 : Fonction linéaire bornée
par Θ^- et Θ^+ :

$$f(x) = \begin{cases} \text{M} & \text{si } x > \Theta^+ \\ kx & \text{si } \Theta^- \leq x \leq \Theta^+ \\ m & x < \text{si } \Theta^- \end{cases}$$

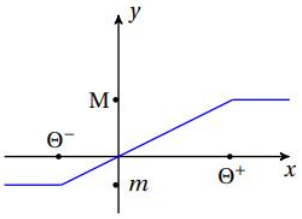


FIGURE 10 – Fonction linéaire bornée

Exemple 4 : Unité linéaire rectifiée (" Rectified Linear Unit', ReLU)

- $g(x) = \max(0, x)$ <=> 0 si $x < 0$ et x si $x \geq 0$ - Cette fonction d'activation (et ses variantes) est fréquemment utilisée en apprentissage profond. - Fonction utilisée pour les couches cachées ("hidden layers") des réseaux de neurones. - Avantages et inconvénients : o Cout de calcul faible et fonction facile à optimiser. o Convergence plus rapide que sigmoide ou tanh. o Améliore la propagation du gradient. o Si x < 0 => poids non actualisés => pas d'apprentissage. o Non différentiable à zéro. o Pas centrée en zéro et non bornée.

Exemple 5 : Fonction sigmoide exponentielle

$$f(x) = \frac{1}{1 + e^{-x}}$$

Sa dérivée est (1)

$$f'(x) = f(x)(1 - f(x))$$

ce qui montre, avec la notation $a_i = f(e_i)$, l'identité suivante (2)

$$\frac{da_i}{de_i} = a_i (1 - a_i)$$

DÉMONSTRATION. Avec $u(x) = 1 + e^{-x} = 1/f$, nous avons $u'(x) = -e^{-x} = 1 - u(x) = 1 - 1/f$ et donc $f'(x) = -\frac{u'(x)}{u(x)^2} = (1/f - 1)f^2 = (1 - f)f$.

$$\lim_{x \to \infty} f(x) = 1 \quad \text{et} \quad \lim_{x \to -\infty} f(x) = 0$$

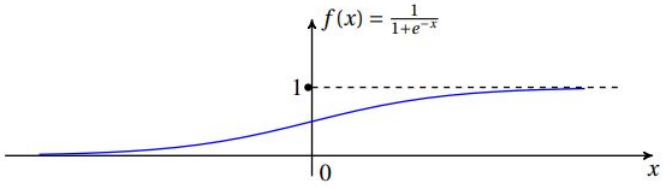


FIGURE 11 – Fonction sigmoide exponentielle

Cette fonction continue monotone croissante bornée est celle souvent utilisée dans le réseau appelé le perceptron multicouches que nous étudierons plus tard.

Exemple 6 : La fonction sigmoide tangentielle

La fonction sigmoide tangentielle converge plus vite que la sigmoide exponentielle.

$$f(x) = \tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \quad \text{et } f'(x) = 1 - f(x)^2$$

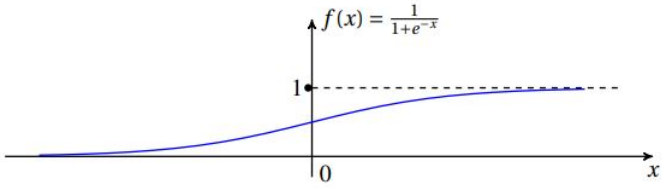


FIGURE 12 – La fonction sigmoide tangentielle

5 Les réseaux de neurones convolutifs et le traitement d'images

Les réseaux de neurones convolutifs sont à ce jour les modèles les plus performants pour classer des images. Désignés par l'acronyme CNN, de l'anglais Convolutional Neural Network, ils comportent deux parties bien distinctes. En entrée, une image est fournie sous la forme d'une matrice de pixels. Elle a 2 dimensions pour une image en niveaux de gris. La couleur est représentée par une troisième dimension, de profondeur 3 pour représenter les couleurs fondamentales [Rouge, Vert, Bleu]. En sortie, une prédiction : quels objets apparaissent dans l'image, ou encore la fonction qui représente l'allure de la route.
Les CNN possèdent plusieurs paramètres : les filtres, les poids et les biais. La force de ces algorithmes est qu'ils ont la capacité d'apprendre par eux-mêmes ce qui rend le pré-traitement des données beaucoup moins important que pour d'autres algorithme de classification. Un CNN est une fonction non-linaire.

6 Architecture d'un CNN (Traitement d'images)

Le CNN est une fonction qui prend en entrée une image et qui donne un vecteur de probabilité, chaque classe se voit attribuée une probabilité, ainsi la prédiction est la classe assimilée à la plus grande probabilité. L'architecture des CNN est divisée en 2 partie : « Feature Learning » et « Classification »

6.1 Premiére partie : Feature Learning

La première partie permet d'extraire des caractéristiques et de diminuer le nombre de données à traiter dans la classification. Le bloc « Feature Learning » est sous divisé en plusieurs blocs de différentes couches appelées « layers ». Il est fréquent de rencontrer au moins 3 blocs. Chaque bloc est composé de 3 couches :
-la « Convolutional layer »,
-la « Activation Layer »
-la « MaxPooling Layer ».

Cet enchaînement de bloc nous permet d'extraire des caractéristiques de l'image. Ce sont ces dernières qui vont être classifiées.

6.1.1 Couche de Convolution (Convolutional Layer)

Dans cette couche, l'image est filtrée par produits de convolution vu précédemment au travers de filtres (kernels), les valeurs de ces filtres sont initialisés de manières aléatoire, et l'algorithme apprend les valeurs durant la phase d'entraînement de l'algorithme.
Les paramètres à spécifier sont, le nombre de filtres, la taille des filtres et l'architecture du réseau.
Plus de filtres implique plus de caractéristiques extraites et plus le CNN devient bon à reconnaître les patterns.

La taille de la « Feature Map » ou « Image filtrée » dépend de 3 paramètres :

-**La Profondeur (Depth) :** correspond au nombre de filtres utilisés.
-**Le pas (Stride) :** correspond au nombre de pixels par lequel le filtre « glisse » sur la matrice d'entrée. Lorsque l'écart est de 1, le filtre est déplacé tous les 1 pixel, c'est à dire que tous les pixels sont traités, lorsque l'écart est de 2, le filtre est déplacé tous les 2 pixels etc...
-**La marge (Padding) :** En appliquant un filtre à une image, chaque pixel résultant dépend de ses voisins. Au centre de la matrice, cela ne pose aucun problèmes, mais quand est-il des pixels aux bords qui n'ont par définition aucun voisin.

En effet, le « valid-padding », c'est à dire de ne pas prendre de marge, autrement-dit d'utiliser l'image d'entrée brute, réduit la dimension de la « Features Map », car nous perdons les pixels situés aux bords. Ainsi, avec une image d'entrée 5x5 et un filtre 3x3, notre Features Map sera de dimensions 3x3. Afin de garder la taille de l'image initiale, il est possible d'ajouter une ou plusieurs bordures de 0 autour de l'image simulant ainsi des pixels voisins des pixels des bords.

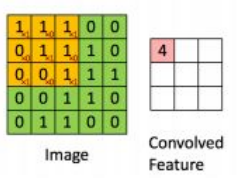


FIGURE 13 – Padding "valid"

La taille de la Features Map est données par cette relation : $D = \frac{W-K+2P}{S} + 1$ avec D : la taille de la Features Map, W la taille de l'entrée, K la taille du filtre, P le nombre de bordures de marges, S le pas.

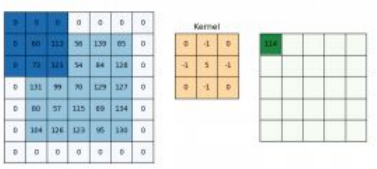


FIGURE 14 – Padding "same"

6.1.2 Fonction D'activation ou Normalisation (ReLU)

Après chaque opérations de convolutions, une opération non-linéaire est ajoutée. Le but de ces fonctions nonlinéaire, est d'introduire des propriétés non-linéaires à notre CNN. En effet, la plupart des données de l'environnement autour de la voiture ne sont pas linéaires, et nous voulons que notre CNN puissent apprendre et prédire des relations non-linéaires. Les opérations de convolutions sont linéaires puisque ce sont seulement des multiplications de matrices et additions.
Comme on l'a vu dans la section Fonction d'activation, plusieurs fonctions existent mais la plus courante est la fonction Relu (Rectified Linear Unit). par exemple, si la fonction Relu appliquée à un « Features Map » ca va remplacer tous les pixels dont la valeur est négative par un 0.

6.1.3 Pooling Layer

L'étape du Pooling (= Mise en commun) aussi-appelée sous-échantillonnage réduit la dimension de la Features Map mais garde les informations les plus importantes. et comme onl'a vu aussi dans la section Pooling, Il existe plusieurs types : Max, Moyenne, Somme etc.. La plus utlisée est la Max-Pooling.

Cette étape consiste à définir une matrice (2x2 par exemple), et de la même manière que les filtres pour la convolution, à faire glisser cette matrice sur une entrée. Le Max-Pooling consiste a prendre le plus grand élément de la matrice d'entrée au seins de la matrice de Pooling.
On glisse notre matrice 2x2 tous les 2 pixels (Stride = 2), et nous gardons la plus grande valeur pour chaque région. Ainsi, la dimension de notre Features Map est réduite, ce qui participe à résoudre le problème initial, qui est : trop de données.

Le Max-Pooling est une étape primordiale afin de préparer au mieux l'étape de prédiction et de classification, en particulier
- Il rend les dimensions plus petites et faciles à gérer - Il rend le CNN invariant aux petites transformations dans l'image d'entrée. (Une petite modification ne changera pas la sortie puisque nous prenons la valeur maximale dans une région locale de l'image.

Chaque bloc de Convolutional Layer + ReLU + MaxPooling extrait les caractéristiques utiles et uniques de l'image, et isole donc les différences entre deux images déstinctes. Afin d'optimiser ces extraction plusieurs blocs sont mis à la suite.

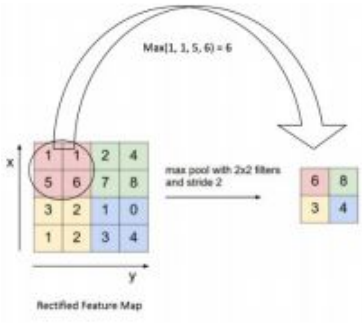


FIGURE 15 – Max-Pooling par une matrice 2x2

6.2 Deuxième partie : Classification

6.2.1 Couche totalement connectée (Fully-connected)

C'est cette layer qui a pour rôle de classifier les éléments de l'image. C'est un traditionnel " Multi Layer Perceptron w, utilisant la fonction softmax en sortie. Le terme $\ll$ Fully-Connected $\gg$ implique que chaque neurone est connecté à chaque neurones de la couche suivante. La première étape est de transformer nos matrices d'entrée, les features map issues du processus d'extraction de caractéristiques, en un seul vecteur. Chaque pixel représente une entrée, avec une valeur, issue de tout le processus précédent.

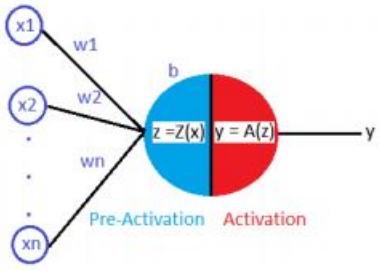


FIGURE 16 – Schéma d'un neurone

Chaque connexion a un poids, initialement aléatoire. La valeur d'un neurone est égale à la somme des entrées multipliée par le poids de la connexion. Cette valeur peut être comparée à un biais, afin d'y ajouté un seuil d'activation. Cette somme est ensuite passée er argument d'une fonction d'activation, la fonction ReLU. Ainsi si la somme est négative, la valeur du neurone sera alors de 0. $Z(X_1, X_2, \ldots, X_n) = X_1 * w_1 + X_2 * w_2 + \ldots + X_n * w_n +$ biais $Z(X) = \sum_{i=1}^{n} (X_i * w_i) + b$

$$Y = \text{ReLU}(z) = \max(0, z)$$

Les poids des différentes connexions, ainsi que les biais de chaque neurone sont aléatoires, l'apprentissage consiste à trouver les poids et les biais corrects afin d'obtenir une extrême précision en sortie.

Références

[1] ARNAUD BODIN, FRANÇOIS RECHER, Deepmath mathématiques (simples) des reseaux de neurones (pas trop compliqués), Exo7

[2] Annick VALIBOUZE, RÉSEAUX DE NEURONES ARTIFICIELS,2018-2019

[3] Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, Proceedings of the 9th Python in Science Conference, pages 51 – 56, 2010.

[4] Deeplizard, Machine Learning Deep Learning Fundamentals

[5] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-Net : Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. In 2016 Fourth International Conference on 3D Vision (3DV), pages 565–571, Stanford, CA, USA, October 2016. IEEE.