Réseaux de neurones profonds

Catherine ACHARD
Institut des Systèmes Intelligents et de Robotique

catherine.achard@sorbonne-universite.fr

Références

http://cs231n.github.io/
http://www.yaronhadad.com/deep-learning-most-amazing-applications
https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html
http://yann.lecun.com/exdb/lenet/index.html
https://blog.octo.com/classification-dimages-les-reseaux-de-neurones-convolutifs-en-toute-simplicite/
http://www.deeplearningbook.org/
https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/
https://deeplearning4j.org/

Applications

http://www.yaronhadad.com/deep-learning-most-amazing-applications/

Classification

- But : L'algorithme doit dire à quelle catégorie appartient une entrée
- Exemple : L'image d'une personne ➔ son identité

Régression

- But : L'algorithme doit associer une valeur numérique à une entrée
- Exemple : Paramètres météorologique ➔ température à 24 heures

Retranscription

- But : A partir de l'observation d'une entrée, la transcrire sous une forme textuelle
- Exemple : A partir d'images de google street, donner les numéros de rue

Traduction

- But : convertir une séquence de symbole en une autre séquence de symbole dans un autre langage
- Exemple : traduction du français à l'anglais

Détection d'anomalie

- But : Trouver si une entrée est atypique ou non
- Exemple : détecter des voitures en sens contraire sur des images routières

Synthèse

- But : apprendre à générer de nouveaux exemples proches de ceux d'une base de données
- Application : Synthétiser des visages sous un autre point de vue

Débruitage

- But : Apprendre à générer des données non bruitées à partir de données bruitées

Applications

http://www.yaronhadad.com/deep-learning-most-amazing-applications/

Re-colorier des images ou vidéos noir et blanc

Iizuka, S., Simo-Serra, E., & Ishikawa, H. (2016). Let there be color!: joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM Transactions on Graphics*

Applications

http://www.yaronhadad.com/deep-learning-most-amazing-applications/

Super-résolution : augmentation d'un facteur 3

Image originale | Interpolation bi-cubique | Apprentissage (deep)

Dong, C., Loy, C. C., He, K., & Tang, X. (2014, September). Learning a deep convolutional network for image super-resolution. In *European Conference on Computer Vision* (pp. 184-199). Springer, Cham.

Applications

http://www.yaronhadad.com/deep-learning-most-amazing-applications/

Estimation de la pose multi-personnes temps réel

Cao, Z., Simon, T., Wei, S. E., & Sheikh, Y. (2017, July). Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR* (Vol. 1, No. 2, p. 7).

Applications

http://www.yaronhadad.com/deep-learning-most-amazing-applications/

Décrire des photos

Karpathy, A., & Fei-Fei, L. (2015). Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*

Applications

http://www.yaronhadad.com/deep-learning-most-amazing-applications/

Analyse temps réel du comportement

Applications

http://www.yaronhadad.com/deep-learning-most-amazing-applications/

Traduction automatique

Applications

http://www.yaronhadad.com/deep-learning-most-amazing-applications/

Gagner contre des jeux vidéos

Applications

http://www.yaronhadad.com/deep-learning-most-amazing-applications/

Génération automatique d'écriture manuscrite

Figure 15: **Real and generated handwriting**. The top line in each block is real, the rest are unbiased samples from the synthesis network. The two texts are from the validation set and were not seen during training.

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Applications

http://www.yaronhadad.com/deep-learning-most-amazing-applications/

La conduite de véhicules automatique

Applications

http://www.yaronhadad.com/deep-learning-most-amazing-applications/

La publicité

- accroître la pertinence des publicités
- créer de la publicité prédictive axée sur les données
- faire des enchères en temps réel pour les publicités
- publicité d'affichage ciblée

Applications

http://www.yaronhadad.com/deep-learning-most-amazing-applications/

La robotique

Applications

http://www.yaronhadad.com/deep-learning-most-amazing-applications/

La finance

Stratégie de trading

Applications

http://www.yaronhadad.com/deep-learning-most-amazing-applications/

La génération de voix

La composition misicale

Le perceptron

$\boldsymbol{x} = [x_1, \dots, x_n]^T$: signal d'entrée

Un neurone est défini par :

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

Avec

- w_i les poids
- b le biais
- $a = \sum_{i=1}^{n} w_i x_i + b$ activation
- f la fonction d'activation, fonction sigmoïde par exemple

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

Le perceptron

Différentes fonctions d'activations possibles

Sigmoïde

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

Le perceptron

Apprentissage

En choisissant judicieusement les poids w_i et b, le neurone peut séparer les données linéairement séparables (2 classes)

$$y = f\left(\sum_{i=1}^{2} w_i x_i + b\right)$$

Le perceptron

Pour un problème non linéaire, on utilise des couches cachées

Il s'agit du **perceptron multicouches (multilayer perceptron MLP)**
Deux types de couches :

- Une ou plusieurs **couches cachées** composées de plusieurs neurones
- Une **couche de sortie** composée de K neurones $\boldsymbol{y} = [y_1, \dots, y_K]^T$

Les réseaux de neurones

En **régression**

- sortie continue
- la dernière couche a une fonction d'activation linéaire

En **classification à K classes**

One Hot Encoding : la dernière couche a K neurones.

Pour un exemple *t* de la classe k,

- $y_{t,k} = 1$
- $y_{t,k' \neq k} = 0$
- $y_{t,k}$: valeur du *k*ième neurone de sortie pour l'exemple *t*

La dernière couche a une fonction d'activation **softmax** qui transforme la sortie du réseau en probabilités d'appartenance aux classes :

$$y_{t,k} = \frac{\exp(y_{t,k})}{\sum_{k'} \exp(y_{t,k'})}$$

Les réseaux de neurones

Apprentissage : trouver les poids du réseau pour que les sorties soient le plus proches possible des sorties désirées.

Minimisation d'une fonction appelée **fonction perte** ou **loss function :**

$$\mathcal{L}(\boldsymbol{W}) = \frac{1}{N}\sum_{t=1}^{N} l_w(\boldsymbol{y}_t, \widehat{\boldsymbol{y}_t})$$

Où $\boldsymbol{W}$: vecteur contenant tous les poids du réseau,

$\boldsymbol{y}_t$: sortie désirée pour l'exemple $\boldsymbol{x}_t$

$\widehat{\boldsymbol{y}_t}$: sortie prédite par le réseau

N : nombre d'exemples d'apprentissage

En **régression**, **erreur quadratique** (MSE : Mean Square Error):

$$l_w(\boldsymbol{y}_t, \widehat{\boldsymbol{y}_t}) = \frac{1}{2}\|\boldsymbol{y}_t - \widehat{\boldsymbol{y}_t}\|^2$$

En **classification à K classes, entropie croisée** (après le softmax) :

$$l_w(\boldsymbol{y}_t, \widehat{\boldsymbol{y}_t}) = -\sum_{k=1}^{K} y_{t,k} \log \widehat{y_{t,k}}$$

Les réseaux de neurones

Exercice

Considérons un problème de classification à 3 classes (K=3).

On considère 3 exemples appartenant à la classe 1 dont la sortie du réseau, avant le softmax, vaut : $\begin{pmatrix} 0.9 \\ 0.2 \\ 0.2 \end{pmatrix} \begin{pmatrix} 0.2 \\ 0.3 \\ 0.2 \end{pmatrix} \begin{pmatrix} 0.2 \\ 0.9 \\ 0.2 \end{pmatrix}$

Déterminer la valeur de l'entropie croisée pour ces 3 exemples.

Les réseaux de neurones

Exercice

Considérons un problème de classification à 3 classes (K=3).

On considère 3 exemples appartenant à la classe 1 dont la sortie du réseau, avant le softmax, vaut : $\begin{pmatrix}0.9\\0.2\\0.2\end{pmatrix}$ $\begin{pmatrix}0.2\\0.3\\0.2\end{pmatrix}$ $\begin{pmatrix}0.2\\0.9\\0.2\end{pmatrix}$

Déterminer la valeur de l'entropie croisée pour ces 3 exemples.

Il faut d'abord faire le softmax. Les 3 vecteurs deviennent:

$$\begin{pmatrix}0.5017\\0.2491\\0.2491\end{pmatrix} \begin{pmatrix}0.3220\\0.3559\\0.3220\end{pmatrix} \begin{pmatrix}0.2491\\0.5017\\0.2491\end{pmatrix}$$

Puis $l_w(\boldsymbol{y}_t, \widehat{\boldsymbol{y}_t}) = -\sum_{k=1}^{K} y_{t,k} \log \widehat{y_{t,k}}$

$$l_{w1} = -1\text{x}log(0{,}5017) - 0\text{x}log(0.2491) - 0\text{x}log(0.2491) = 0{,}30$$
$$l_{w2} = -1\text{x}log(0{,}3220) - 0\text{x}log(0.3559) - 0\text{x}log(0.3220) = 0{,}49$$
$$lw_3 = -1\text{x}log(0{,}2491) - 0\text{x}log(0.5017) - 0\text{x}log(0.2491) = 0{,}60$$

➔ Le dernier exemple amène à une erreur plus forte.

➔ La fonction perte que l'on cherche à minimiser est la somme des entropies croisées : on veut que tous les exemples soient bien classés

Les réseaux de neurones

Minimisation d'une fonction par descente de gradient

➔ il faut que la fonction perte soit dérivable, dépendant des poids

Raisonnons sur une fonction générale $\mathcal{L}(W)$

Partant de poids initiaux w_0, le minimum de $\mathcal{L}(W)$ s'obtient par itération avec :

$$w_t = w_{t-1} - \lambda \nabla \mathcal{L}(w_{t-1})$$

Où

- $\nabla \mathcal{L}(w)$ est le gradient de $\mathcal{L}$ par rapport à un poids w
- λ est le pas d'apprentissage.

Les réseaux de neurones

La convergence dépend du pas d'apprentissage

Et du point initial

Les réseaux de neurones

Différentes fonctions d'activations possibles

Sigmoïde

- $f(x) = \frac{1}{1+e^{-\lambda x}}$
- $\frac{\partial f(x)}{\partial x} = \frac{\lambda e^{-\lambda x}}{(1+e^{-\lambda x})^2} = \mathrm{f(x)}(1 - f(x))$

Deux inconvénients

- Sature et tue les gradients pour de fortes activations du neurones
- La sortie n'est pas centrée

Les réseaux de neurones

Tangente hyperbolique

$$f(x) = tan\,h(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

$$\frac{\partial f(x)}{\partial x} = 1 - f^{2}(x)$$

Inconvénient

- Sature et tue les gradients pour de fortes activations du neurones

Avantage

- La sortie est centrée

Les réseaux de neurones

RELU (REctified Linear Unit)

$$f(x) = \max(0, x)$$
$$\frac{\partial f(x)}{\partial x} = \begin{cases} 0 \; si \; x \leq 0 \\ 1 \; si \; x > 0 \end{cases}$$

Avantages

- Très rapide à calculer (pas d'exponentielle)
- Permet au réseau de converger beaucoup plus vite (six fois)
- Forts gradients (0 ou 1)

Inconvénient

- Certains neurones « meurent » en ne produisant que des zéros (les poids sont tels que la sortie est nulle pour toutes les entrées). Comme le gradient devient nul, ils ne se réactivent jamais

Les réseaux de neurones

LeakyRELU

$$f(x) = \max(\alpha x, x) \text{ avec } \alpha = 0{,}01$$

Avantages

- Très rapide à calculer (pas d'exponentielle)
- Permet au réseau de converger beaucoup plus vite (six fois)

Les réseaux de neurones

Pour mettre en place un MLP, l'utilisateur doit donc déterminer

1. les variables d'entrée et la variable de sortie
2. Le nombre de couches cachées et le nombre de neurones de chaque couche (dilemme biais/variance)
3. Le nombre maximum d'itérations
4. Le taux d'apprentissage λ (ou sa stratégie d'évolution)

L'initialisation des poids est aléatoire

Les réseaux de neurones

Les variables d'entrée et la variable de sortie

Les variables d'entrées doivent être **centrées et normalisées** de manière à bien activer les fonctions d'activation

➔ Pour chaque dimension, on retire la moyenne et divise par l'écart-type

ATTENTION : moyenne et écart-type doivent être estimés sur les données d'apprentissage et appliqués ensuite à toute la base (apprentissage et test)

L'initialisation des poids

Pour ne pas saturer les fonctions d'activation, les poids doivent être initialisés à de petites valeurs.

Une solution est de les tirer selon une loi normale centrée, réduite

Les réseaux de neurones

Le nombre de couches cachées et le nombre de neurones de chaque couche (dilemme biais/variance)

Apprentissage avec 1 couche cachée composée de

3 neurones 6 neurones 20 neurones

https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

➔ Avec plus de neurones, on peut approximer des fonctions plus complexes mais **attention à l'overfitting**

➔ Il vaut mieux utiliser des modèles plus complexes avec une régularisation pour éviter l'overfitting (voir plus loin)

Les réseaux de neurones

Pas d'apprentissage

Le réglage du pas d'apprentissage est important :

- s'il est trop fort, il peut amener à une divergence
- S'il est trop faible, l'apprentissage aura du mal à converger

Les réseaux de neurones

Pas d'apprentissage

Plusieurs solutions

- Pas d'apprentissage constant (**step decay**)
- Diminution par paliers : il diminue d'un pourcentage toutes les T itérations

- Diminution exponentielle (**Exponential decay**): $\lambda_{it} = \lambda_0 e^{-\beta it}$

Les réseaux de neurones

Nombre d'itérations : attention à l'over-fitting !

Solution : base de validation pour arrêter l'apprentissage

Les réseaux de neurones

Descente de gradient sur la base d'apprentissage

On a vu que la mise à jour des poids est réalisée avec :

$$w_{t+1} = w_t - \lambda \nabla \mathcal{L}(w_t)$$

$$w_{t+1} = w_t - \lambda \frac{1}{N} \sum_{t=1}^{N} \nabla l_w(\boldsymbol{y}_t, \widehat{\boldsymbol{y}_t})$$

➔ Le gradient est calculé pour chaque exemple $\boldsymbol{x}_t$ de la base d'apprentissage mais le modèle est mis à jour que quand tous les exemples sont passés

Avantage

- Erreur et convergence stables

Inconvénient

- La convergence peut être faite dans un minima local
- L'ensemble de la base d'apprentissage doit pouvoir être chargée en mémoire

Les réseaux de neurones

Descente de gradient stochastique, Stochastic gradient descent (SGD)

Le modèle est mis à jour après chaque exemple d'apprentissage (tiré aléatoirement selon une distribution uniforme)

Avantage :

- Pas besoin d'avoir tous les exemples en mémoire
- Moins de risque de rester dans un minima local

Inconvénient :

- Plus couteux en temps de calcul car il faut faire la mise à jour pour chaque exemple
- La descente de gradient est très bruitée

Les réseaux de neurones

Descente de gradient par mini-batch

C'est une solution intermédiaire ou la base est divisée en mini-batchs de taille fixe.
La mise à jour se fait à la fin de chaque mini-batch:

$$w_{t+1} = w_t - \lambda \nabla \mathcal{L}(w_t)$$

$$w_{t+1} = w_t - \lambda \frac{1}{N} \sum_{batch} \nabla l_w(\boldsymbol{y}_t, \widehat{\boldsymbol{y}_t})$$

En général, entre 32 et 256 exemples par mini-batch

Avantage

- Minimise l'erreur de généralisation (les gradients 'bruités' agissent comme une régularisation)
- Convergence plus rapide

Les réseaux de neurones

Terminologie

- **Mini batch ou batch** : sous ensemble de la base d'apprentissage
- **Epoch** : nombre de fois que tous les exemples sont vus en apprentissage
- **Itération** : nombre de batchs vus en apprentissage

Exercice

Considérons une base d'apprentissage composée de 200 exemples,
Nous prenons des batchs de taille 5 et 1000 epochs

- Combien y a-t-il de batchs ?
- Combien t a-t-il d'itérations ?

Les réseaux de neurones

Terminologie

- **Mini batch ou batch** : sous ensemble de la base d'apprentissage
- **Epoch** : nombre de fois que tous les exemples sont vus en apprentissage
- **Itération** : nombre de batchs vus en apprentissage

Exercice

Considérons une base d'apprentissage composée de 200 exemples,
Nous prenons des batchs de taille 5 et 1000 epochs

- Combien y a-t-il de batchs ?
- Combien t a-t-il d'itérations ?

- 40 batchs
- 40.000 itérations

La batch normalisation

Idée : Stabiliser l'apprentissage des batchs

Pendant l'apprentissage, les batchs changent à chaque itération et peuvent ne pas avoir la même distribution. Les poids du CNN s'adaptent en permanence.

➔ On normalise les entrées $x^t = [x_1^t, \dots, x_k^t, \dots, x_n^t]$

$$x_k^t = \frac{x_k^t - \mu_k}{\sigma_k}$$

Avec $\mu_k = \underset{t=1,\dots,N}{\text{mean}}\, x_k^t$ et $\sigma_k = \underset{t=1,\dots,N}{\text{std}}\, x_k^t$

Batch 1

Batch 2

La batch normalisation durant le test

Après l'apprentissage, on utilise toute la base pour estimer des statistiques fiables μ etσ.

On utilise ces statistiques durant l'étape de prédiction

La batch normalisation

Cette normalisation peut être faite sur les entrées de toutes les couches

Les réseaux de neurones

Le dropout
But : éviter le sur-apprentissage

Les réseaux profonds ont un grand nombre de paramètres ➔ sujet au sur apprentissage et à la perte de gradient
Idée du dropout : pour chaque epoch, enlever aléatoirement des neurones lors de l'apprentissage

En test, tous les neurones sont utilisés

(a) Standard Neural Net (b) After applying dropout.

Dropout: A Simple Way to Prevent Neural Networks from Overfitting (2014, N. Srivastava et. al)

Les réseaux de neurones

Le dropout

Avantages

➔ Les sous-réseaux se focalisent moins sur des détails

➔ Les sous-réseaux ont moins de paramètres ➔ évite l'over-fitting

➔ Améliore la vitesse d'apprentissage (moins de poids à mettre à jour)

Les réseaux de neurones

Le dropout

Epoch 1

Les réseaux de neurones

Le dropout

Epoch 2

Les réseaux de neurones

La régularisation

But: éviter le sur-apprentissage

Idée : un réseau avec des petits poids est sans doute plus simple qu'un réseau avec des poids important

- **Régularisation L2**
  on pénalise la fonction perte avec la norme L2

$$\mathcal{L} + \frac{1}{2}\alpha\|\omega\|^2$$

- **Régularisation 1**
  on pénalise la fonction objective avec la norme L1

$$\mathcal{L} + \frac{1}{2}\alpha|\omega|$$

➔ Tend à avoir des poids nuls (modèle sparse) et d'autre très grands

Les réseaux de neurones

Augmentation des données (data augmentation)

But: éviter le sur-apprentissage

Idée : créer de nouvelles données légèrement différentes

- Translation aléatoire
- Rotation aléatoire
- Crop aléatoire
- Ajout aléatoire dans les composante RVB

https://github.com/aleju/imgaug

Les réseaux de neurones

Application : reconnaissance de chiffres manuscrit

Image 16x16 ➔ vecteur de dimension 256
10 chiffres

Exercice :

- Combien y a-t-il d'entrée ?
- Avec 100 neurones sur la couche cachée, combien y a-t-il de poids et de biais à estimer ?
- Et si l'image était de taille 100x100 ?

Les réseaux de neurones

Application : reconnaissance de chiffres manuscrit

Image 16x16 ➔ vecteur de dimension 256
10 chiffres

Exercice :

- Combien y a-t-il de neurones en entrée et en sortie ?
- Avec 100 neurones sur la couche cachée, combien y a-t-il de poids et de biais à estimer ?
- Et si l'image était de taille 100x100 ?

- 256 neurones en entrée
- 10 neurones en sortie

Couche cachée :
256*100+100
Couche de sortie :
10*100+10
Soit : **26.710** paramètres
Pour une image de 100x100
1.011.110 paramètres

Les réseaux convolutionnels

Pourquoi aller plus loin ?

- Le perceptron multicouches ne passe pas à l'échelle (grande image)
- Une légère translation de la forme (1 pixel) change fortement les données d'entrée puisque chaque neurone de la couche d'entrée représente un pixel précis
- De la même façon, le réseau est très sensible aux changements d'échelle

Les réseaux convolutionnels

Rappel sur la convolution

$$IC(x,y) = \sum_i \sum_j G(i,j) I(x-i, y-j)$$

L'idée des réseaux convolutifs est d'appliquer des convolutions dans les premières couches du réseau

Les coefficients du filtre (et un biais supplémentaire) sont les poids du réseau
Un neurone par pixel MAIS

- Chaque neurone n'est connecté qu'à un petit nombre de neurones de la couche précédente
- Les poids sont partagés par tous les neurones
- Les résultats de convolution sont envoyés à une fonction d'activation

Les réseaux convolutionnels

Exercice :
Donner le résultat de convolution de l'image suivante par le filtre suivant (on ne gérera pas les bords)

Image

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Filtre

1	0	1
0	1	0
1	0	1

Image convoluée

https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Les réseaux convolutionnels

Exercice :
Donner le résultat de convolution de l'image suivante par le filtre suivant (on ne gérera pas les bords)

Image

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Filtre

1	0	1
0	1	0
1	0	1

Image convoluée

4	3	4
2	4	3
2	3	4

https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Les réseaux convolutionnels

Explication de :
https://medium.com/technologymadeeasy/
the-best-explanation-of-convolutional-neural-networks-on-the-internet-fbb8b1ad5df8

Les CNN travaillent sur des volumes :

Les réseaux convolutionnels

Explication de :
https://medium.com/technologymadeeasy/
the-best-explanation-of-convolutional-neural-networks-on-the-internet-fbb8b1ad5df8

Les CNN travaillent sur des volumes :

Les réseaux convolutionnels

Explication de :
https://medium.com/technologymadeeasy/
the-best-explanation-of-convolutional-neural-networks-on-the-internet-fbb8b1ad5df8

Les CNN travaillent sur des volumes :

Les réseaux convolutionnels

Explication de :
https://medium.com/technologymadeeasy/
the-best-explanation-of-convolutional-neural-networks-on-the-internet-fbb8b1ad5df8

Les CNN travaillent sur des volumes :

Les réseaux convolutionnels

https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Les réseaux convolutionnels

Dans les réseaux convolutifs, plusieurs convolutions sont estimées pour chaque couche, par exemple

Pour un filtre donné, tous les neurones extraient exactement la même caractéristique (un contour vertical par exemple) à des positions différentes

➔ Les CNN sont bien adaptés pour être invariants en translation

http://yann.lecun.com/exdb/lenet/index.html

Les réseaux convolutionnels

Stride et padding

Ces deux valeurs sont utiles pour définir une couche de convolution.

- Le **stride** indique le pas avec lequel on déplace le filtre. Un stride de 1 amène à une convolution classique. Un stride de 2 amène à une image 2 fois plus petite.

- Le **padding** sert à ajouter des colonnes et des lignes de zéros sur les bords de l'image pour que la convolution puisse être calculée sur toute l'image

Exercice
Considérons une image de taille 8x8 à laquelle on applique une couche convolutionnelle composée de 6 filtres 3x3, avec un stride de 2 et un padding de 0. Comment sera la sortie ?
Même question avec un padding de 1.

Les réseaux convolutionnels

Exercice

Considérons une image de taille 8x8 à laquelle on applique une couche convolutionnelle composée de 6 filtres 3x3, avec un stride de 2 et un padding de 0. Comment sera la sortie ?

Même question avec un padding de 1.

Avec un stride de 2 et un padding de 0, on arrive à 6 cartes (6 filtres) de taille 3x3

Avec un padding de 1, on a 6 cartes de taille 4x4

Les réseaux convolutionnels

Chaque neurone des couches de convolution est soumis à une **fonction d'activation non linéaire**

http://mlss.tuebingen.mpg.de/2015/slides/fergus/Fergus_1.pdf

Les réseaux convolutionnels

Convolution

Activation non linéaire

Les réseaux convolutionnels

Les CNN comportent aussi des couches de sous-échantillonnage appelées pooling,

Le pooling

- réduit la dimension spatiale ➔ le nombre de paramètres ➔ contrôle le sur-apprentissage
- ajoute de l'invariance spatiale en translation
- ajoute de l'invariance spatiale en échelle
- Ils dépendent également du stride et du padding

Deux principaux types de pooling : max pooling ou average pooling

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

6	8
3	4

Max pooling

3	5
2	2

Average pooling

Les réseaux convolutionnels

http://mlss.tuebingen.mpg.de/2015/slides/fergus/Fergus_1.pdf

Les réseaux convolutionnels

Réseau de neurones convolutifs

Flatten
Codage

classifieur

La première partie, composée de couches de convolution et de pooling, extrait les caractéristiques

La second partie, avec des neurones entièrement connectés (comme un MLP), réalise la classification

https://blog.octo.com/classification-dimages-les-reseaux-de-neurones-convolutifs-en-toute-simplicite/

Les réseaux convolutionnels

Exercice

- On considère le réseau ci-dessus pour lequel, toutes les opérations de convolution et de pooling sont faites avec un padding ‘same’.
- Les poolings se font avec une fenêtre de taille 3x3 et un stride de 2
- Les convolutions se font avec des fenêtres 3x3 et un stride de 1
- La couche « fully connected » FC a 3 neurones
- Toutes les fonctions d’activation sont des RELU, sauf sur la dernière couche où on a du soft-max

Les réseaux convolutionnels

Exercice

- On met en entrée l'image ci-dessous: Quelles seront les 3 images en sortie de la première couche sachant que les 3 filtres utilisés sont (stride=1, padding=1, pas de bais):

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

0	0	0	1	1	0	0	0
0	0	1	1	1	1	0	0
0	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	0
0	0	1	1	1	1	0	0
0	0	0	1	1	0	0	0

Les réseaux convolutionnels

Exercice

- On met en entrée l'image ci-dessous: Quelles seront les 3 images en sortie de la première couche sachant que les 3 filtres utilisés sont : (stride=1, padding=1, pas de bais):

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

0	1	3	5	5	3	1	0
1	3	6	8	8	6	3	1
3	6	8	9	9	8	6	3
5	8	9	9	9	9	8	5
5	8	9	9	9	9	8	5
3	6	8	9	9	8	6	3
1	3	6	8	8	6	3	1
0	1	3	5	5	3	1	0

0	0	0	1	1	0	0	0
0	0	1	1	1	1	0	0
0	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	0
0	0	1	1	1	1	0	0
0	0	0	1	1	0	0	0

0	0	0	0	1	2	1	0
0	0	0	0	1	2	2	1
0	0	0	0	0	1	2	2
0	0	0	0	0	0	1	3
0	0	0	0	0	0	1	3
0	0	0	0	0	1	2	2
0	0	0	0	1	2	2	1
0	0	0	0	1	2	1	0

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	1	0	0	0	0	1	1
2	2	1	0	0	1	2	2
1	2	2	1	1	2	2	1
0	1	2	3	3	2	1	0

Les réseaux convolutionnels

Exercice

- Quelles seront les sorties de la première couche de pooling 3x3 (stride=2, padding=1)?

0	1	3	5	5	3	1	0
1	3	6	8	8	6	3	1
3	6	8	9	9	8	6	3
5	8	9	9	9	9	8	5
5	8	9	9	9	9	8	5
3	6	8	9	9	8	6	3
1	3	6	8	8	6	3	1
0	1	3	5	5	3	1	0

3	8	8	6
8	9	9	9
8	9	9	9
6	9	9	8

0	0	0	0	1	2	1	0
0	0	0	0	1	2	2	1
0	0	0	0	0	1	2	2
0	0	0	0	0	0	1	3
0	0	0	0	0	0	1	3
0	0	0	0	0	1	2	2
0	0	0	0	1	2	2	1
0	0	0	0	1	2	1	0

0	0	2	2
0	0	2	3
0	0	1	3
0	0	2	2

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	1	0	0	0	0	1	1
2	2	1	0	0	1	2	2
1	2	2	1	1	2	2	1
0	1	2	3	3	2	1	0

0	0	0	0
0	0	0	0
2	2	1	2
2	3	3	2

Les réseaux convolutionnels

Suit une étape de convolution avec les filtres (mêmes valeurs sur les 3 cartes):

stride=1
padding=1
pas de biais

3	8	8	6
8	9	9	9
8	9	9	9
6	9	9	8

0	0	2	2
0	0	2	3
0	0	1	3
0	0	2	2

0	0	0	0
0	0	0	0
2	2	1	2
2	3	3	2

0	0	0	21
0	0	0	32
0	0	0	36
0	0	0	25

0	0	0	0
0	0	0	0
0	0	0	0
20	31	35	25

Les réseaux convolutionnels

Puis de nouveau un max-pooling (stride=2, padding=1)

0	0	0	21
0	0	0	32
0	0	0	36
0	0	0	25

0	32
0	36

0	0	0	0
0	0	0	0
0	0	0	0
20	31	35	25

0	0
31	35

Les réseaux convolutionnels

Puis une mise à plat :

0	32
0	36

0	0
31	35

0
0
32
36
0
31
0
35

La couche FC est composée de 3 neurones dont les poids et biais valent:

1	1	-1	-1	0	0	0	0	0
0	0	1	1	1	1	0	0	5
0	0	0	0	-1	-1	1	1	3

Que vaudra la sortie avant le soft-max ? -68, 104 et 7

Après le soft-max ? 0, 1, 0

Les réseaux convolutionnels

Un des premiers CNN (un petit CNN) a été proposé par Yan Lecun en 1998 : **LENET5**

Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, **Gradient-Based Learning Applied to Document Recognition**, *Proceedings of the IEEE, 86(11):2278-2324, November **1998***

Les couches de convolution

Exercice
Couche C1 : 6 filtres de taille 5x5
Pourquoi les features map sont de taille 28x28
Combien de paramètres y a-t-il à estimer sur la première couche ?

Les réseaux convolutionnels

Les couches de convolution

Exercice
Couche C1 : 6 filtres de taille 5x5
Pourquoi les features map sont de taille 28x28
Combien de paramètres y a-t-il a estimer sur la première couche ?

Effet de bord ➔ 2 pixels de chaque coté ➔stride=1, padding=0
Pour chaque filtre, il faut estimer les coefficients du filtre 5x5 et le biais
➔6*(5x5+1) = 156

Les réseaux convolutionnels

Le pooling (ou agglomération - subsampling)

Le pooling
Quel stride ?

Les réseaux convolutionnels

Le pooling (ou agglomération - subsampling)

Le pooling
Quel stride ?

Stride = 2

Les réseaux convolutionnels

Les fonctions d'activation

+RELU

Une fonction d'activation est mise en place après les couches de convolution et fully connected. **La fonction d'activation la plus utilisée est la fonction RELU**, Pourquoi ?

Pour des entrées importantes, la dérivée de la sigmoïde est presque nulle

➔ Les gradients sont très petits et le réseau a du mal à mettre à jour les poids

➔ Lors de la mise à jour des poids, les gradients sont multipliés d'une couche à l'autre (chain rule), ils deviennent de plus en plus petits et ont tendance à disparaitre (vanishing gradient problem)

Les réseaux convolutionnels

Exercice

On considère le réseau ci-dessus utilisé pour régresser une variable. Donner le nombre de paramètres à estimer

Les réseaux convolutionnels

Nombre total de paramètres : (3x3x8+8) + (3x3x16x8+16) +(3x3x32x16+32) +(3x3x32x32+32) + (32x7x7+1)
= 16705

L'architecture VGG16

SIMONYAN, Karen et ZISSERMAN, Andrew. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, **2014**.
http://www.robots.ox.ac.uk/~vgg/research/very_deep/

L'architecture VGG16

Toutes les convolutions sont faites avec des filtres 3x3, pourquoi ?

L'architecture VGG16

Toutes les convolutions sont faites avec des filtres 3x3, pourquoi ?

Deux convolutions 3x3 reviennent à une convolution 5x5
Trois convolutions 3x3 reviennent à une convolutions ??
Un filtre large peut être remplacé par une pile de filtres successifs plus petits
➔ Moins de paramètres à estimer
➔ Plus de non linéarités

L'architecture VGG16

Exemple : VGG16

Combien y a-t-il de paramètres à estimer ?

L'architecture VGG16

Exemple : VGG16
INPUT: [224x224x3] poids:
CONV3-64: [224x224x64] poids :
CONV3-64: [224x224x64] poids :
POOL2: [112x112x64] poids :
CONV3-128: [112x112x128] poids :
CONV3-128: [112x112x128] poids :
POOL2: [56x56x128] poids :
CONV3-256: [56x56x256] poids :
CONV3-256: [56x56x256] poids :
CONV3-256: [56x56x256] poids :
POOL2: [28x28x256] poids :
CONV3-512: [28x28x512] poids :
CONV3-512: [28x28x512] poids :
CONV3-512: [28x28x512] poids :
POOL2: [14x14x512] poids :
CONV3-512: [14x14x512] poids :
CONV3-512: [14x14x512] poids :
CONV3-512: [14x14x512] poids :
POOL2: [7x7x512] poids :
FC: [1x1x4096] poids :
FC: [1x1x4096] poids :
FC: [1x1x1000] poids :
TOTAL :

L'architecture VGG16

Exemple : VGG16

INPUT: [224x224x3] poids: 0
CONV3-64: [224x224x64] poids : (3*3*3)*64 +64= 1792
CONV3-64: [224x224x64] poids : (3*3*64)*64+64 = 36928
POOL2: [112x112x64] poids : 0
CONV3-128: [112x112x128] poids : (3*3*64)*128+128 = 73856
CONV3-128: [112x112x128] poids : (3*3*128)*128+128 = 147584
POOL2: [56x56x128] poids : 0
CONV3-256: [56x56x256] poids : (3*3*128)*256+256 = 295168
CONV3-256: [56x56x256] poids : (3*3*256)*256+256 = 590080
CONV3-256: [56x56x256] poids : (3*3*256)*256+256 = 590080
POOL2: [28x28x256] poids : 0
CONV3-512: [28x28x512] poids : (3*3*256)*512+512 = 1180160
CONV3-512: [28x28x512] poids : (3*3*512)*512+512 = 2359808
CONV3-512: [28x28x512] poids : (3*3*512)*512+512 = 2359808
POOL2: [14x14x512] poids : 0
CONV3-512: [14x14x512] poids : (3*3*512)*512+512 = 2359808
CONV3-512: [14x14x512] poids : (3*3*512)*512+512 = 2359808
CONV3-512: [14x14x512] poids : (3*3*512)*512+512 = 2359808
POOL2: [7x7x512] poids : 0
FC: [1x1x4096] poids : 7*7*512*4096+4096 = 102764544
FC: [1x1x4096] poids : 4096*4096+4096 = 16781312
FC: [1x1x1000] poids : 4096*1000+1000 = 4097000
TOTAL : ~138.000.000 paramètres

L'architecture VGG16

VGG16

Appris sur **ImageNet** :
138.000.000 de paramètres à estimer
1.200.000 images en apprentissage
1000 classes d'images à reconnaitre

- Taille du batch=256
- Tous les neurones sont équipés de la fonction d'activation **RELU**
- Descente de gradient avec **momentum** (m=0.9)
- **Dropout** lors de l'apprentissage des deux premières couches entièrement connectés
- Pas d'apprentissage initial = 0,01. Il décroit d'un facteur 10 trois fois pour arriver à 0,00001 à la fin
- Appris sur imagenet (3 semaines sur 4 GPU)

L'architecture VGG16

Visualiser les filtres de convolution

On ne visualise pas les filtres eux mêmes mais les images qui maximisent la réponse de chaque filtre :

- On choisit une fonction perte qui maximise la valeur d'un filtre
- On part d'une image blanche
- On fait une descente de gradient pour modifier l'image d'entrée

L'architecture VGG16

Exemple : VGG16

8 filtres (parmi 64) de la première couche de convolution

Le premier banc de filtres détecte des couleurs

https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

L'architecture VGG16

Exemple : VGG16

8 filtres (parmi 128) de la seconde couche de convolution

8 filtres (parmi 256) de la 3ième couche de convolution

L'architecture VGG16

Exemple : VGG16

8 filtres (parmi 512) de la 4ième couche de convolution

8 filtres (parmi 512) de la 5ième couche de convolution

L'architecture VGG16

Couche 1

Couche 2

Couche 3

Couche 4

Couche 5

Les couches 1 et 2 détectent principalement des couleurs, contours, formes simples

Qui on s'enfonce dans le réseau, on voit des formes de plus en plus complexes

Par exemple, le filtre 41 du bloc 5 semble encoder des têtes d'oiseaux

https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

L’architecture VGG16

Les réseaux de neurones

Fine tuning ou transfer learning

But: Profiter des réseaux déjà existants (ils mettent 2 à 3 semaines à être appris sur de multiple GPU)

Le Fine Tuning consiste à utiliser un réseau déjà entrainé comme initialisation pour un nouveau problème.

➔ On profite d'un réseau déjà appris sur des millions d'images comme extracteur de caractéristiques

➔ Il suffit de changer les dernières couches entièrement connectées pour les réapprendre sur le nouveau problème

➔ Lors de l'entrainement du nouveau réseau, on peut
- Geler les couches initiales et apprendre seulement les nouvelles couches
- Utiliser un faible taux d'apprentissage pour les couches déjà apprises

Les réseaux de neurones

Exemple avec l'architecture VGG :

On récupère toute la première partie du réseau et on apprend juste les couches fully connected

L'architecture GoogleNet - Inception

Inception : autre architecture de réseau

- Les parties intéressantes de l'image peuvent beaucoup varier en taille
- Choisir la taille des filtres ou leur nombre est difficile
- Des réseaux très profonds sont sujets à l'overfitting

https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202

Pourquoi pas des filtres de taille différente au même niveau ?
Le réseau devient plus large mais pas plus profond (problème du vanishing gradient)

L'architecture GoogleNet - Inception

D'où la première idée naïve

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*

➔ Beaucoup de paramètres à estimer

L'architecture GoogleNet - Inception

Pour limiter le nombre de map d'entrée (et donc le nombre de coefficients à estimer) on ajoute une **convolution 1x1**

D'où le module **Inception**

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*

L'architecture GoogleNet - Inception

Inception : autre architecture de réseau
D'où le réseau GoogleNet

Convolution | Concatenation | Pooling | Soft-max

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*

L'architecture GoogleNet - Inception

Problème du « vanishing gradient »

➔ On ajoute **2 classifieurs auxiliaires**. La fonction de perte totale est une somme pondérée de la perte auxiliaire et de la perte réelle.

➔ Pour aller plus loin https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202

L'architecture GoogleNet - Inception

Erreur de classification Sur ImageNet

Nombre de couches

Image issue de : a tutorial on Deep Residual Networks - Kaiming He - ICML 2016.

L'architecture Resnet

Autre architecture de réseau Resnet

- Quand la profondeur des réseaux augmente, la précision sature puis diminue rapidement
- Ce n'est **pas dû au sur-apprentissage** quand on travaille sur de très grandes bases de données : l'erreur augmente aussi sur la base d'apprentissage
- Probablement dû au **problème de la disparation du gradient** (vanishing gradient). Solution par RELU, DROPOUT, et pour aller plus loin Resnet
- En partant d'un réseau peu profond aux bons résultats, on peut construire un réseau plus profond en gardant les couches du premier réseau et en ajoutant des couches identités ➔ on ne devrait pas augmenter l'erreur
- Mais, on ne sait pas faire de couches identités avec des fonctions non linéaires

L'architecture Resnet

Image issue de : He, K., Zhang, X., Ren, S., & Sun, J. Deep residual learning for image recognition. CVPR2016

- Les couches apprennent une fonction résiduelle

$$H(x) = F(x) + x$$

où $H(x)$ est la fonction désirée

- On suppose que cet apprentissage est plus simple : à l'extrême, on pourra forcer $F(x)$ à zéro pour prendre la connexion directe

- En ajoutant la connexion directe **(skip connexion)**, on n'ajoute pas de paramètres à apprendre

Comparaison de 3 architectures sur ImageNet:

- VGG 19 couches
- 34 couches pleines
- 34 couches résiduelles

L'architecture Resnet

Trait fin : erreur en apprentissage
Trait gras : erreur en test

Couches pleines
18 (VGG) ou 34 couches

Couches résiduelles
18 ou 34 couches

Image issue de : He, K., Zhang, X., Ren, S., & Sun, J. Deep residual learning for image recognition. CVPR2016

L'architecture Resnet

➔ **Resnet** : la révolution de la profondeur

Erreur de classification Sur ImageNet

Image issue de : a tutorial on Deep Residual Networks - Kaiming He - ICML 2016.

Les grands jeux de données

ImageNet

http://www.image-net.org/

La base est organisée en « synonym set » or « **synset** » : ensemble de mots ou phrases décrivant un concept compréhensible ou catégorie. Ils peuvent être décomposés en sous-catégories

Statistiques d'avril 2010

- Nombre de synsets: **21,841**
- Nombre total d'images : **14,197,122**
- Nombre de d'images avec boite englobante annotée: **1,034,908**

- J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, ImageNet: A Large-Scale Hierarchical Image Database. *CVPR 2009.*
- Olga Russakovsky*, Jia Deng*, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 2015

Les grands jeux de données

Catégorie et sous-catégories

Les grands jeux de données

Challenges

- **Localisation d'objet** : L'algorithme produit 5 classes d'objets et 5 boites englobantes. La qualité de la localisation est évaluée sur l'objet qui « matche » le mieux à la vérité

- **Détection d'objets** : L'algorithme produit un ensemble de classes d'objets et leur boite englobante. L'ensemble doit contenir toutes les instances des objets présents dans l'image. Pénalisation si oubli ou détection multiple

- **Détection d'objets à partir de vidéo** : même chose sur des vidéos

- **Classification de scène** : L'algorithme doit retourner les 5 catégories de scènes qui correspondent le plus à une image donnée

- J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, ImageNet: A Large-Scale Hierarchical Image Database. *CVPR 2009.*
- Olga Russakovsky*, Jia Deng*, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 2015

Les grands jeux de données

MS coco

http://cocodataset.org

Statistiques

- 91 type d'objets
- Chaque objet est étiqueté avec sa segmentation pour une localisation précise
- 2,500,00 millions d'instances d'objet
- 328,000 images

(a) Image classification (b) Object localization

(c) Semantic segmentation (d) This work

- Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014, September). Microsoft coco: Common objects in context. In *European conference on computer vision* (pp. 740-755). Springer, Cham.

Les grands jeux de données

Coco challenges

- **Détection d'objet** (boite englogante ou region)

- **Détection de points d'intérêt**

- Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014, September). Microsoft coco: Common objects in context. In *European conference on computer vision* (pp. 740-755). Springer, Cham.

Les librairies

- Matlab https://fr.mathworks.com/solutions/deep-learning.html?s_tid=hp_solutions_deep

- C++
  - Caffe https://caffe2.ai

- Python
  - Pytorch https://pytorch.org
  - Tensorflow https://www.tensorflow.org
  - Keras https://keras.io/