UNIVERSITÉ CÔTE D'AZUR

Bases de l'IA

Traitement d'image

Réseaux neuronaux convolutifs

Elena CABRIO

elena.cabrio@univ-cotedazur.fr

UNIVERSITÉ CÔTE D'AZUR

Plan pour cette séance

- Classification des images
  - Classificateurs linéaires
  - Réseaux neuronaux convolutifs

Slides from Fei-Fei Li, Justin Johnson, Serena Yeung http://vision.stanford.edu/teaching/cs231n/

Classification des images : une tâche essentielle de la vision par ordinateur

UNIVERSITÉ CÔTE D'AZUR

Supposons un ensemble donné d'étiquettes discrètes, par exemple {chat, chien, vache, pomme, tomate, camion, … }

f() = “apple”

f() = “tomato”

f() = “cow”

Dataset: ETH-80, by B. Leibe Slide credit: L. Lazebnik

Récapitulation : classification linéaire

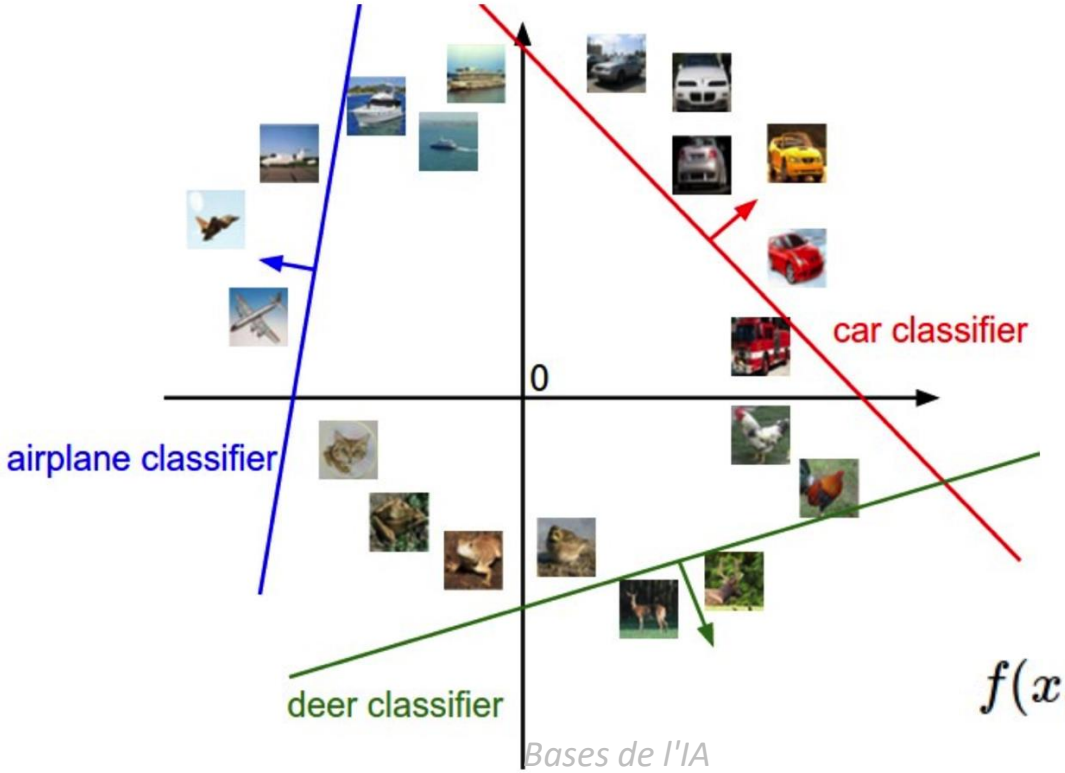
- **Fonction de score** et **fonction de perte**
  - La fonction de score associe une instance de données d'entrée (par exemple, une image) à un vecteur de scores, un pour chaque catégorie.
  - La dernière fois, notre fonction de score était basée sur un classificateur linéaire.

$$f(x,W) = Wx + b$$

f : fonction de score
x : instance d'entrée
W, b : paramètres d'une fonction linéaire

- Trouver **W** et **b** pour minimiser une perte, par exemple la perte d'entropie croisée.

Les classificateurs linéaires séparent l'espace des features en demi-espaces



$$f(x_i, W, b) = Wx_i + b$$

Réseaux de neurones

(**Before**) Linear score function: $f = Wx$

Réseaux de neurones

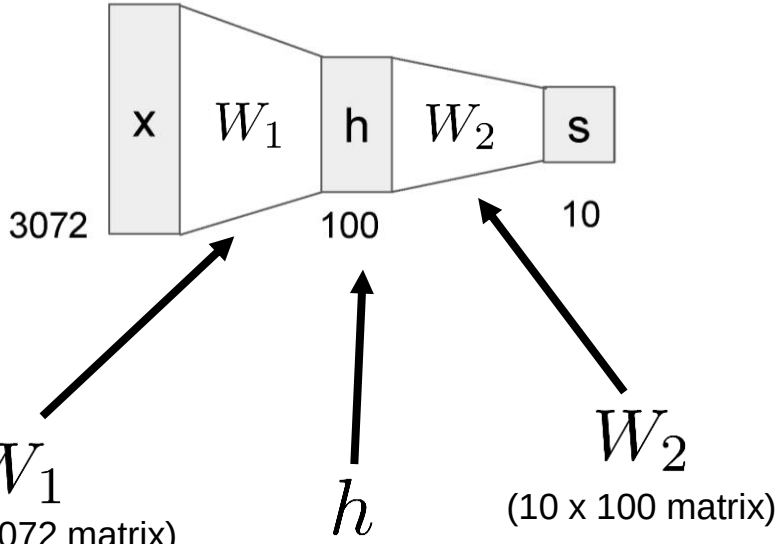
(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$

Réseaux de neurones

(**Before**) Linear score function: $f = Wx$

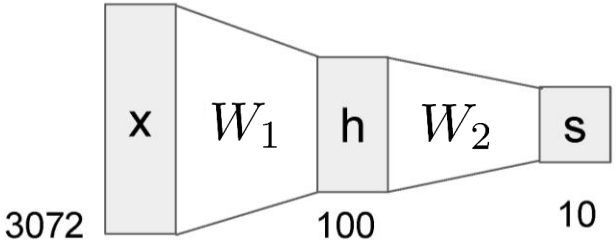
(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$



Réseaux de neurones

(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$



Nombre total de poids à apprendre :

3 072 x 100 + 100 x 10 = 308 200

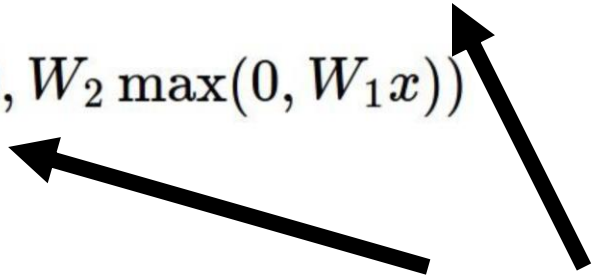
Réseaux de neurones

(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$

or 3-layer Neural Network

$$f = W_3 \max(0, W_2 \max(0, W_1 x))$$



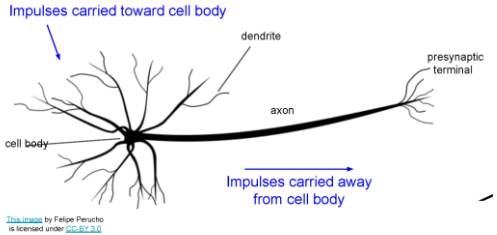
également appelés "perceptrons multicouches" (MLP)

Réseaux de neurones

- Généralisation des réseaux neuronaux :
- Fonctions linéaires enchaînées et séparées par des fonctions non linéaires (fonctions d'activation), par exemple "max".

$$f = W_3 \max(0, W_2 \max(0, W_1 x))$$

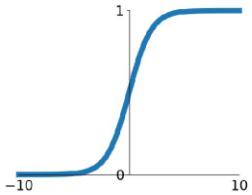
- Pourquoi séparer les fonctions linéaires par des fonctions non linéaires ?
- Très vaguement inspiré par les vrais neurones



Activation functions

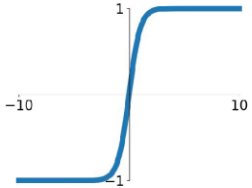
Sigmoid

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



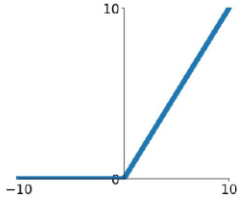
tanh

$$\tanh(x)$$



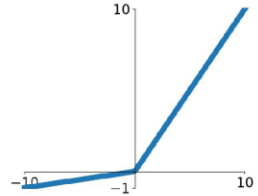
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

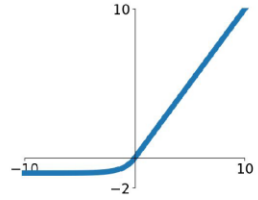


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

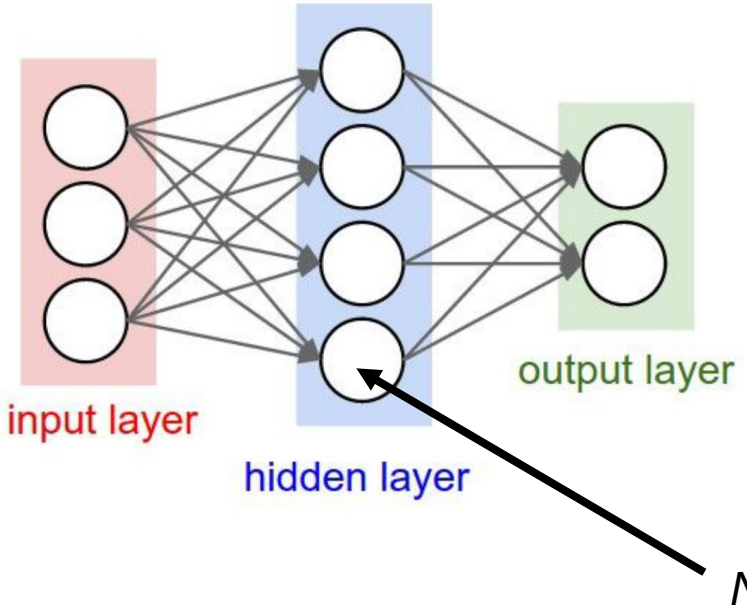
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

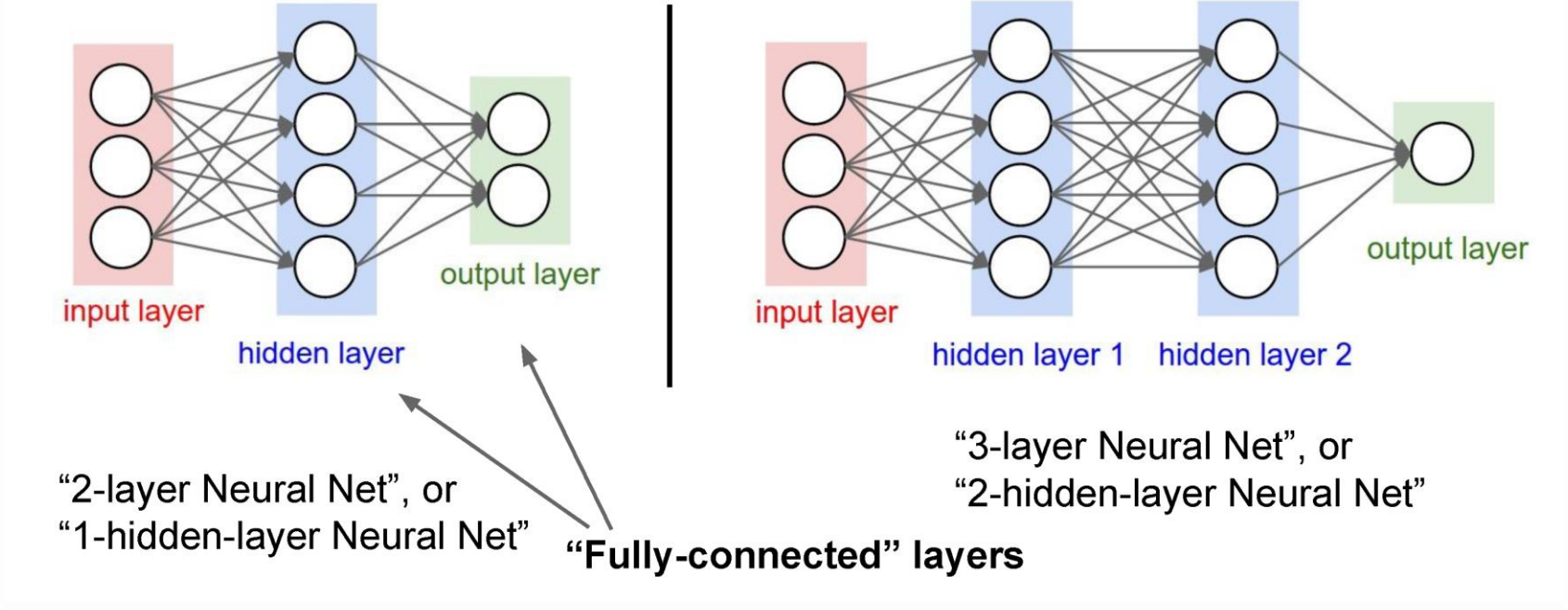


Architecture du réseau de neurones

Réseau de neurones à 2 couches

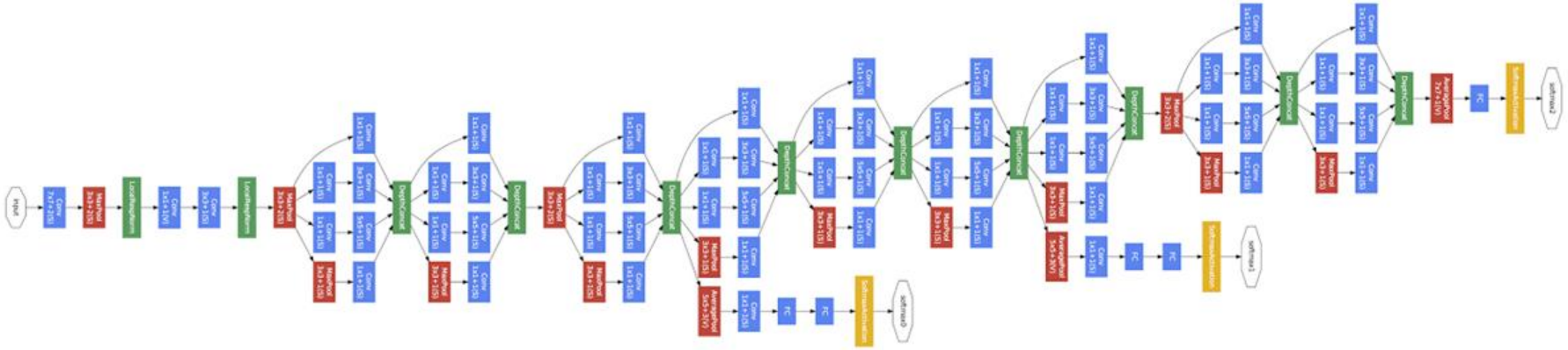


Architecture du réseau de neurones



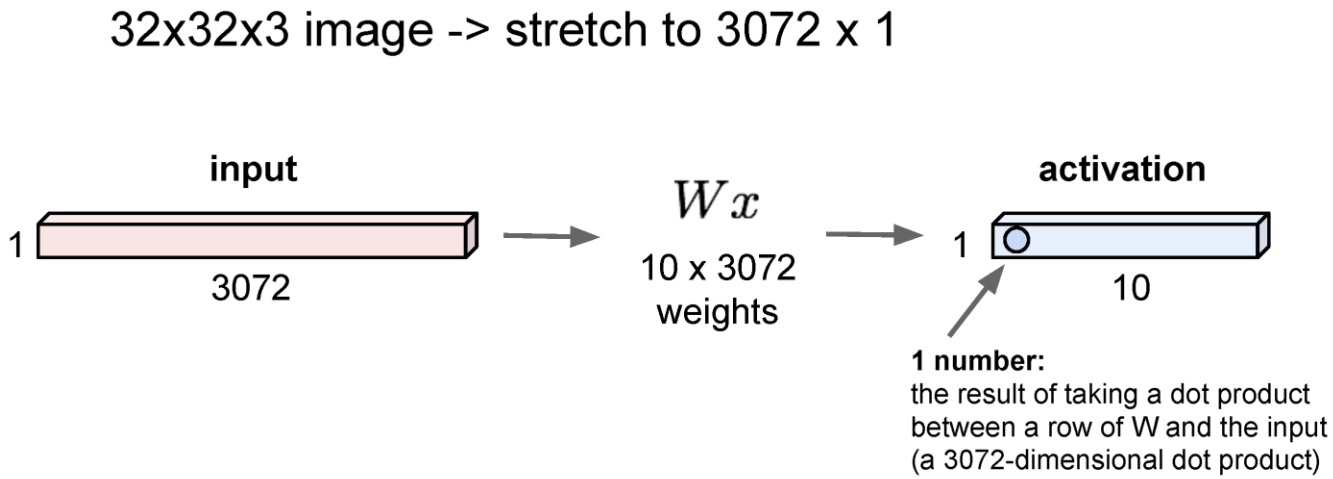
Les réseaux profonds comportent généralement de nombreuses couches et potentiellement des millions de paramètres

Réseau neuronal profond



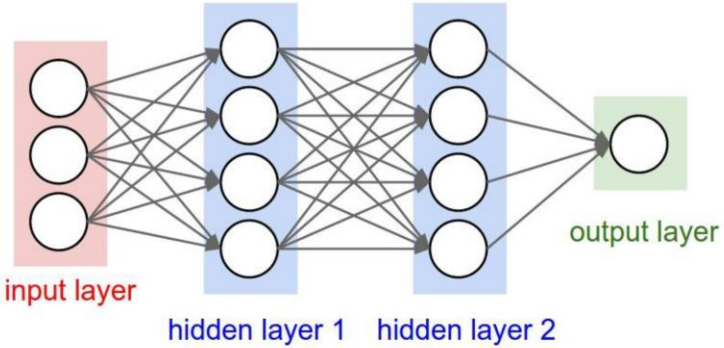
- *Inception* network (Szegedy et al, 2015)
- 22 couches

Couches entièrement connectées



Tout comme un classificateur linéaire, mais dans ce cas, il s'agit d'une seule couche d'un réseau plus grand.

Exemple de calcul feed-forward d'un réseau neuronal




```
# forward-pass of a 3-layer neural network:
f = lambda x: 1.0/(1.0 + np.exp(-x)) # activation function (use sigmoid)
x = np.random.randn(3, 1) # random input vector of three numbers (3x1)
h1 = f(np.dot(W1, x) + b1) # calculate first hidden layer activations (4x1)
h2 = f(np.dot(W2, h1) + b2) # calculate second hidden layer activations (4x1)
out = np.dot(W3, h2) + b3 # output neuron (1x1)
```

UNIVERSITÉ CÔTE D'AZUR

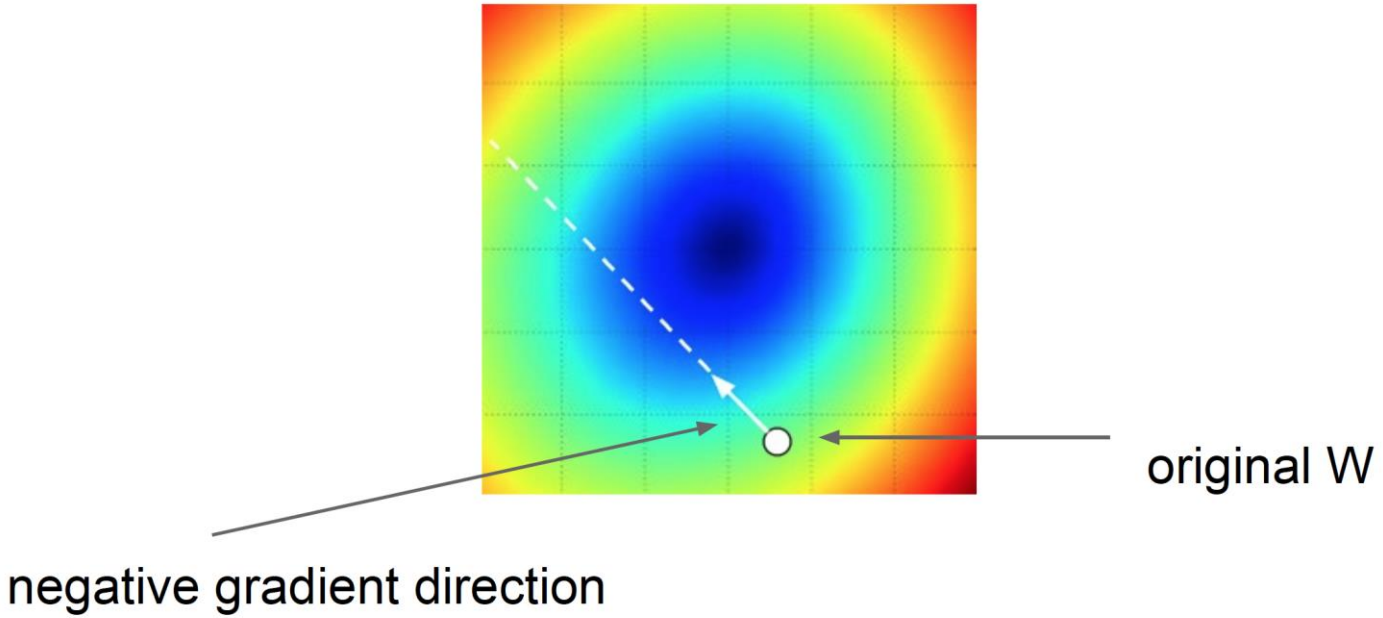
Résumé

- Nous organisons les neurones en **couches entièrement connectées**
- L'abstraction d'une couche a l'avantage de nous permettre d'utiliser un **code vectoriel efficace** (par exemple, les multiplications de matrices).

Optimisation des paramètres avec la descente de gradient

UNIVERSITÉ CÔTE D'AZUR

- **Comment trouver les meilleurs paramètres W et b ?**
- En général : descente de gradient
  - Commencer par une estimation d'un bon W et d'un bon b (ou les initialiser de manière aléatoire)
  - Calculer la fonction de perte pour cette hypothèse initiale et le gradient de la fonction de perte
  - Progresser sur une certaine distance dans la direction du gradient négatif (direction de la descente la plus abrupte)
  - Répéter les étapes 2 et 3
- Remarque : l'exécution efficace de l'étape 2 pour les réseaux profonds est appelée **rétropropagation**.



Descente de gradient : marcher dans la direction opposée au gradient

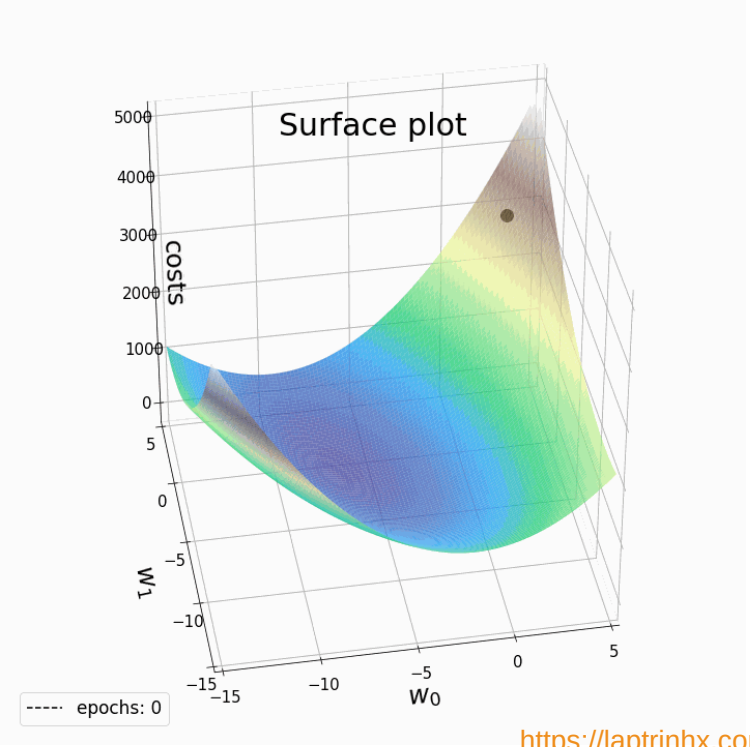
Q : Jusqu'où ?

R : Taille du pas : taux d'apprentissage

Trop grand : manque le minimum

Trop petit : convergence lente

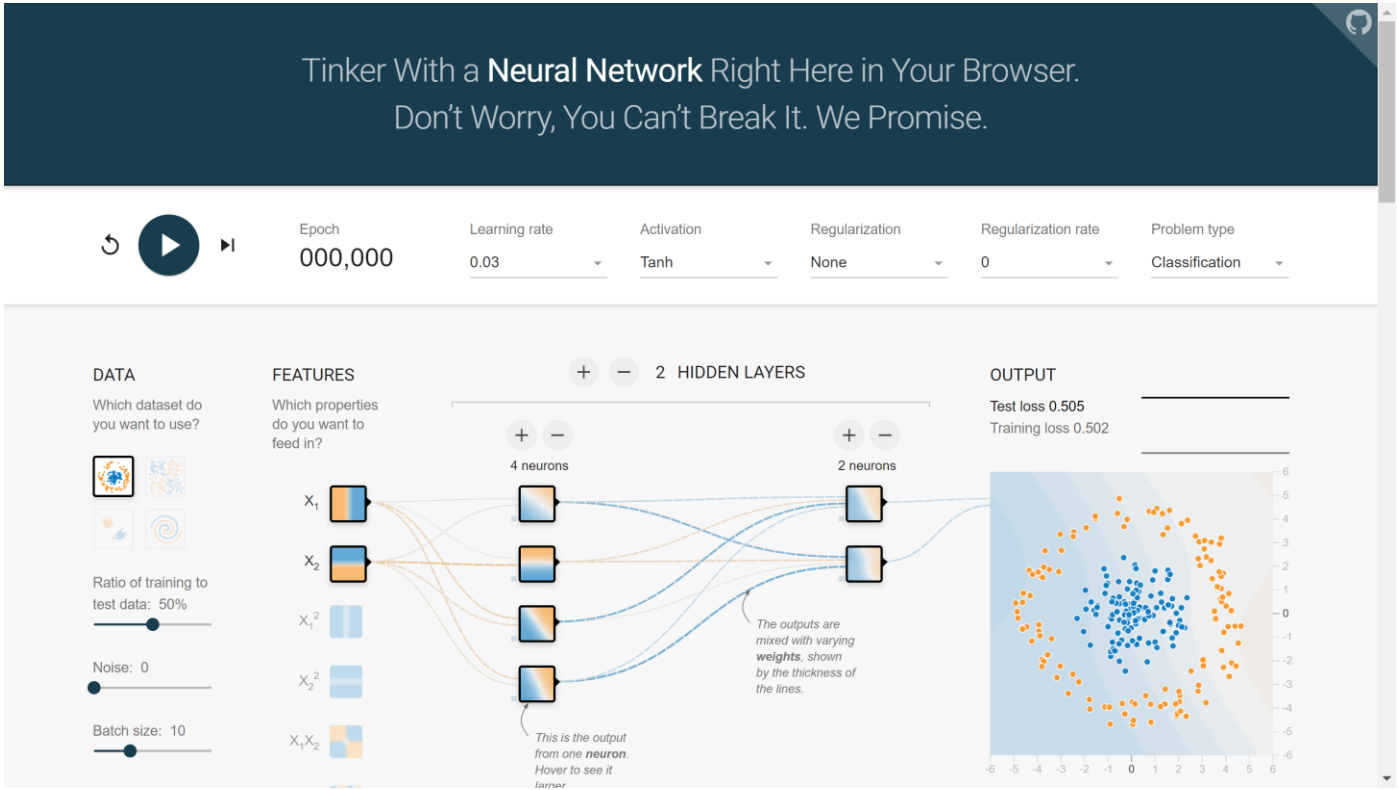
Exemple 2D de descente de gradient



En réalité, dans l'apprentissage profond, nous optimisons une fonction de perte très complexe avec des millions de variables (ou plus)

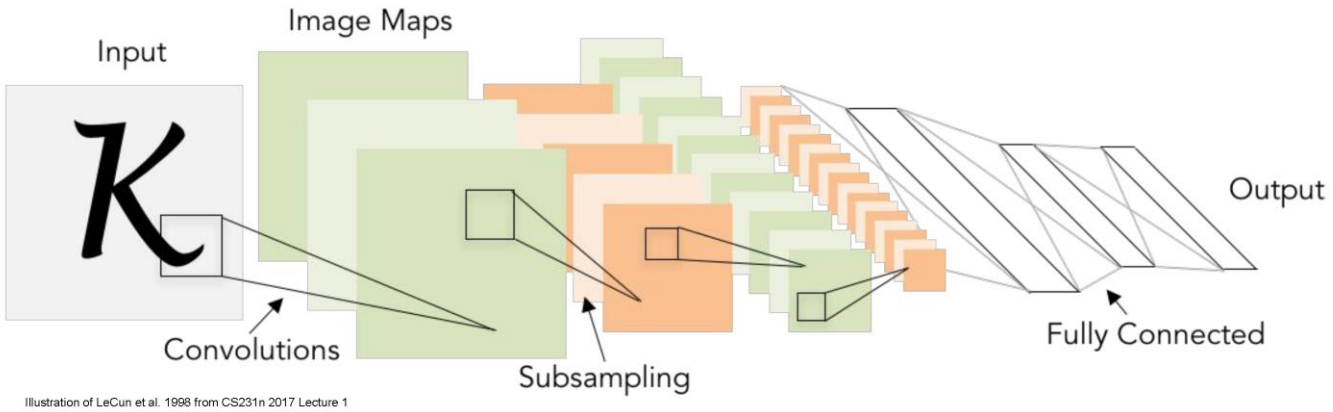
https://laptrinhx.com/gradient-descent-animation-2-multiple-linear-regression-3070246823/

Exemple en 2D : TensorFlow

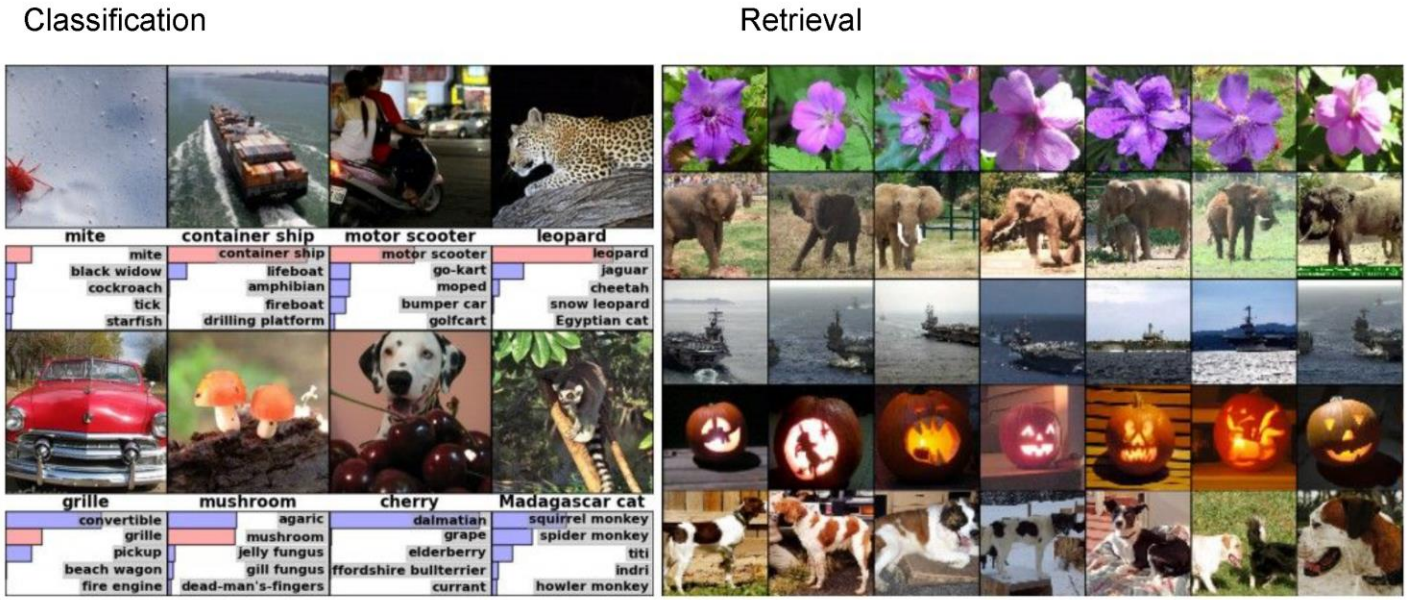


https://playground.tensorflow.org

Réseaux de neurones convolutifs



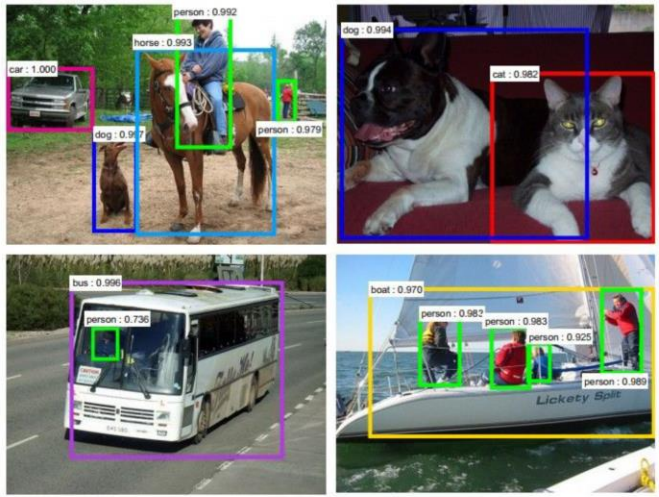
Les réseaux convolutifs sont omniprésents



Figures copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

Les réseaux convolutifs sont omniprésents

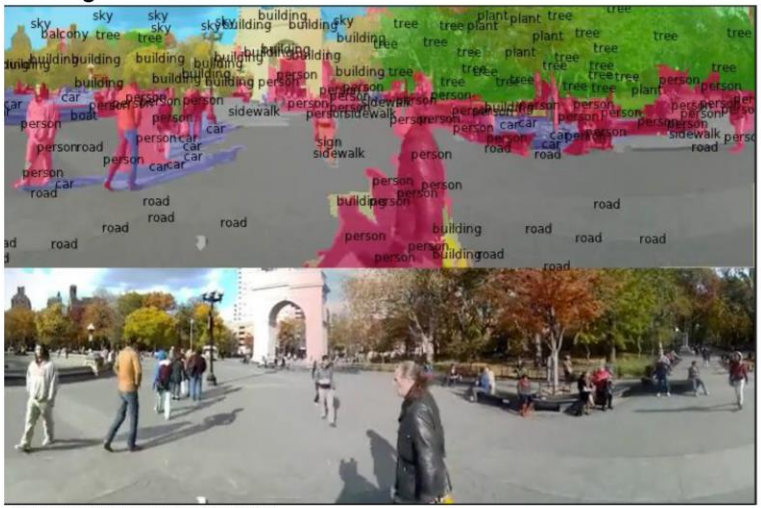
Detection



Figures copyright Shaoqing Ren, Kaiming He, Ross Girschick, Jian Sun, 2015. Reproduced with permission.

[Faster R-CNN: Ren, He, Girshick, Sun 2015]

Segmentation



Figures copyright Clement Farabet, 2012. Reproduced with permission.

[Farabet et al., 2012]

Les réseaux convolutifs sont omniprésents

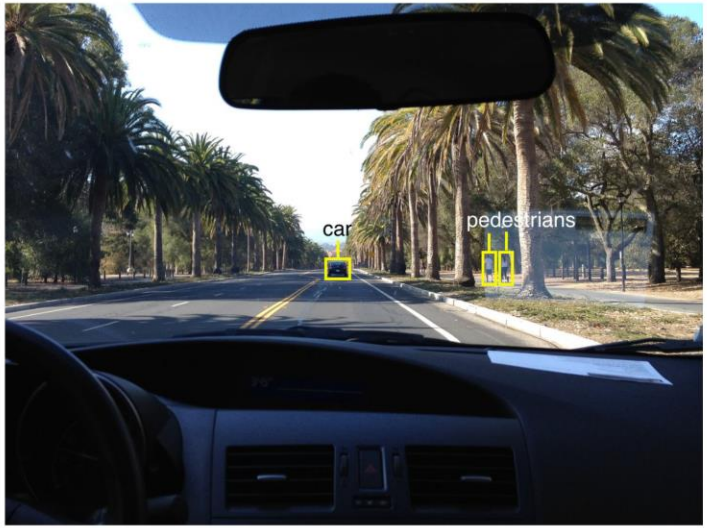


Photo by Lane McIntosh. Copyright CS231n 2017.

self-driving cars

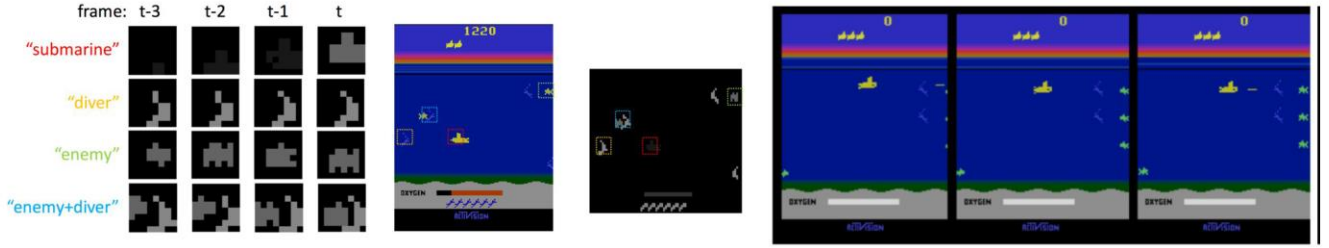
NVIDIA Tesla line
(these are the GPUs on rye01.stanford.edu)

Note that for embedded systems a typical setup would involve NVIDIA Tegras, with integrated GPU and ARM-based CPU cores.

Les réseaux convolutifs sont omniprésents

Images are examples of pose estimation, not actually from Toshev & Szegedy 2014. Copyright Lane McIntosh.

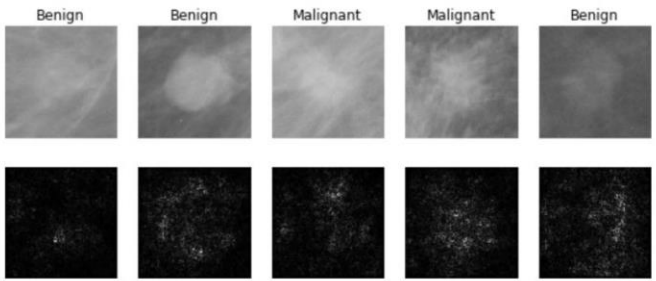
[Toshev, Szegedy 2014]



[Guo et al. 2014]

Figures copyright Xiaoxiao Guo, Satinder Singh, Honglak Lee, Richard Lewis, and Xiaoshi Wang, 2014. Reproduced with permission.

Les réseaux convolutifs sont omniprésents



[Levy et al. 2016]

Figure copyright Levy et al. 2016.
Reproduced with permission.

[Dieleman et al. 2014]

From left to right: public domain by NASA, usage permitted by ESA/Hubble, public domain by NASA, and public domain.

[Sermanet et al. 2011]
[Ciresan et al.]

Photos by Lane McIntosh.
Copyright CS231n 2017.

Image Captioning

[Vinyals et al., 2015]
[Karpathy and Fei-Fei, 2015]

No errors

A white teddy bear sitting in the grass

Minor errors

A man in a baseball uniform throwing a ball

Somewhat related

A woman is holding a cat in her hand

A man riding a wave on top of a surfboard

A cat sitting on a suitcase on the floor

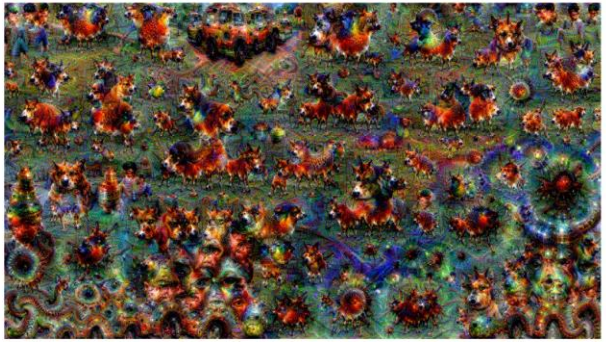
A woman standing on a beach holding a surfboard

All images are CC0 Public domain:
https://pixabay.com/en/luggage-antique-cat-1643010/
https://pixabay.com/en/teddy-plush-bears-cute-teddy-bear-1623436/
https://pixabay.com/en/surf-wave-summer-sport-litoral-1668716/
https://pixabay.com/en/woman-female-model-portrait-adult-983967/
https://pixabay.com/en/handstand-lake-meditation-496008/
https://pixabay.com/en/baseball-player-shortstop-infield-1045263/

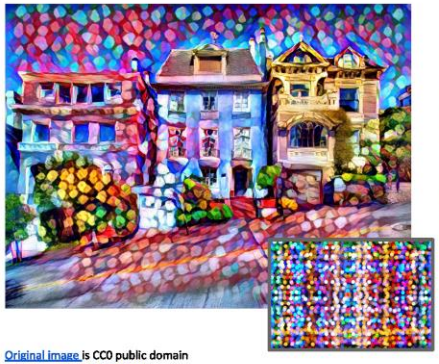
Captions generated by Justin Johnson using Neuraltalk2

De la légende au texte

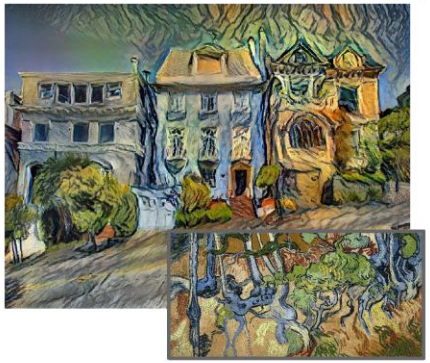
DALL·E: Creating Images from Text, OpenAI
https://openai.com/blog/dall-e/



Figures copyright Justin Johnson, 2015. Reproduced with permission. Generated using the Inceptionism approach from a blog post by Google Research.



Original image is CC0 public domain
Starry Night and Tree Roots by Van Gogh are in the public domain
Bokeh image is in the public domain
Stylized images copyright Justin Johnson, 2017;
reproduced with permission

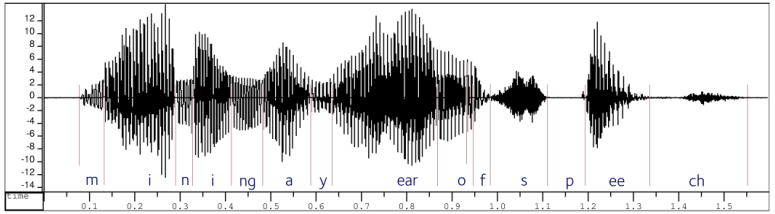


Gatys et al, "Image Style Transfer using Convolutional Neural Networks", CVPR 2016
Gatys et al, "Controlling Perceptual Factors in Neural Style Transfer", CVPR 2017

Réseaux de neurones convolutifs

Version des réseaux de neurones profonds conçue pour les signaux

- Signaux 1D (par exemple, formes d'ondes vocales)

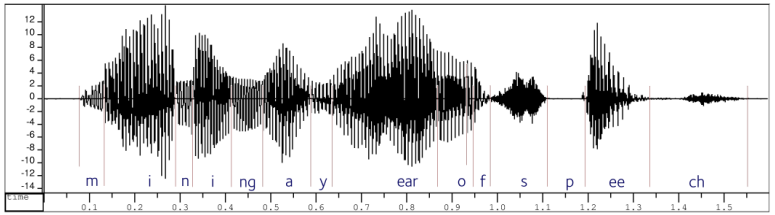


- Signaux 2D (par exemple, images)

Réseaux de neurones convolutifs

Version des réseaux de neurones profonds conçue pour les signaux

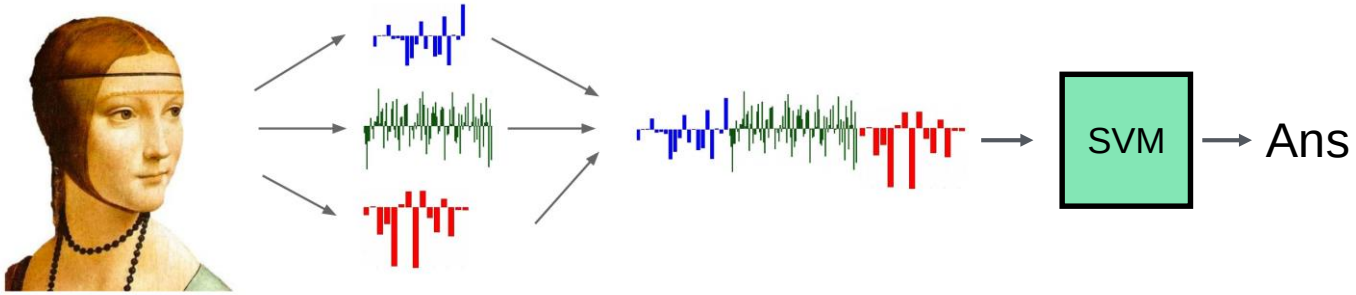
- Signaux 1D (par exemple, formes d'ondes vocales)



- Signaux 2D (par exemple, images)

Motivation:
Apprentissage des features

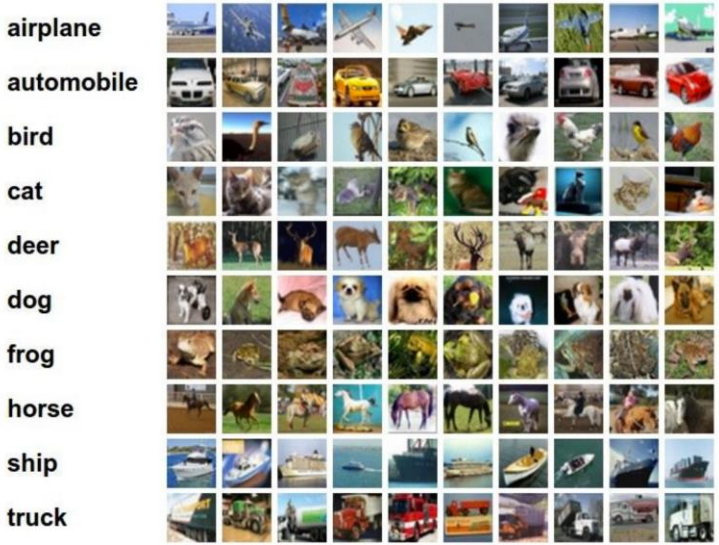
La vie avant l'apprentissage profond…



Pixels d'entrée | *Extraction de features* | *Concaténer dans un vecteur x* | *Classificateur linéaire*

Figure: Karpathy 2016

Pourquoi utiliser des features ? Pourquoi pas des pixels ?



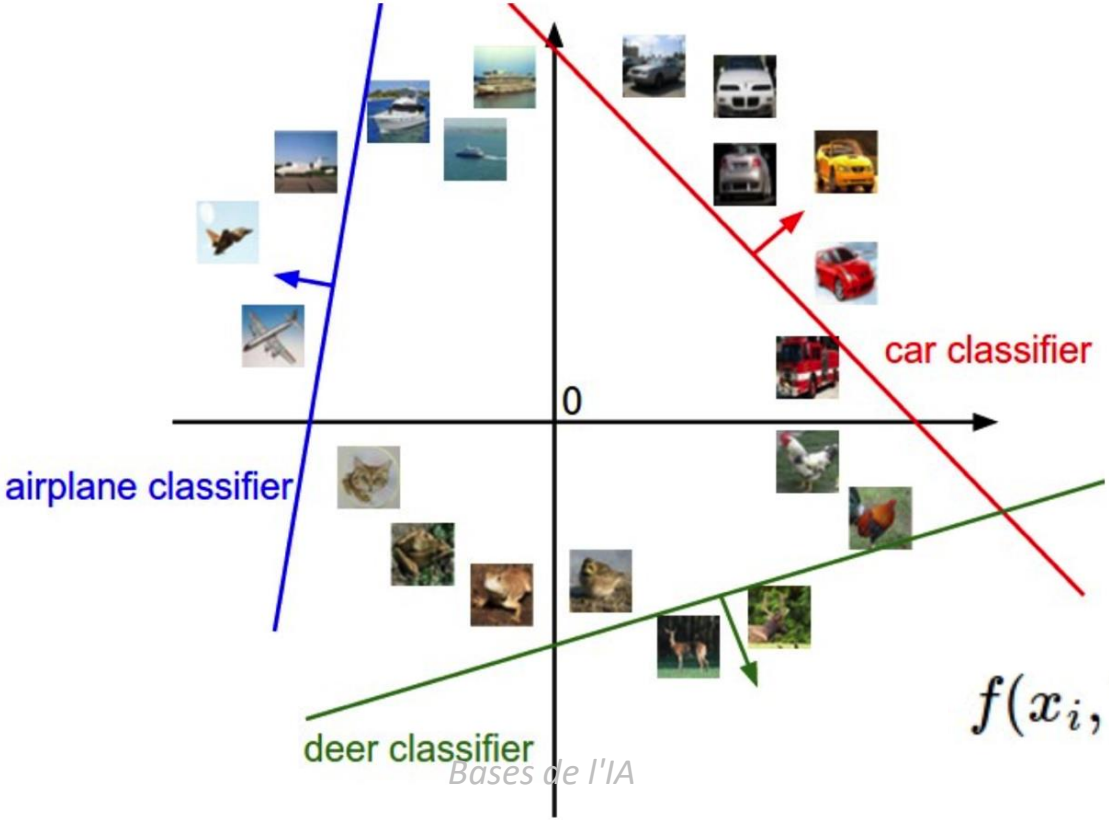
$$f(x_i, W, b) = Wx_i + b$$

Q : Quel serait un ensemble de classes très difficile à distinguer pour un classificateur linéaire ?

(en supposant que x = pixels)

Slide from Karpathy 2016

Classes séparables linéairement



$$f(x_i, W, b) = Wx_i + b$$

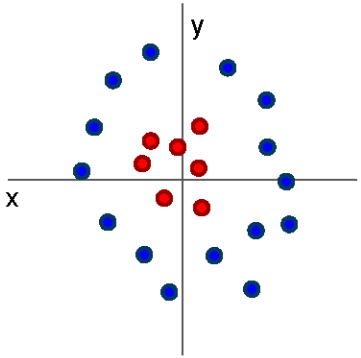
Features de l’image

$$f(x) = Wx$$

Class scores

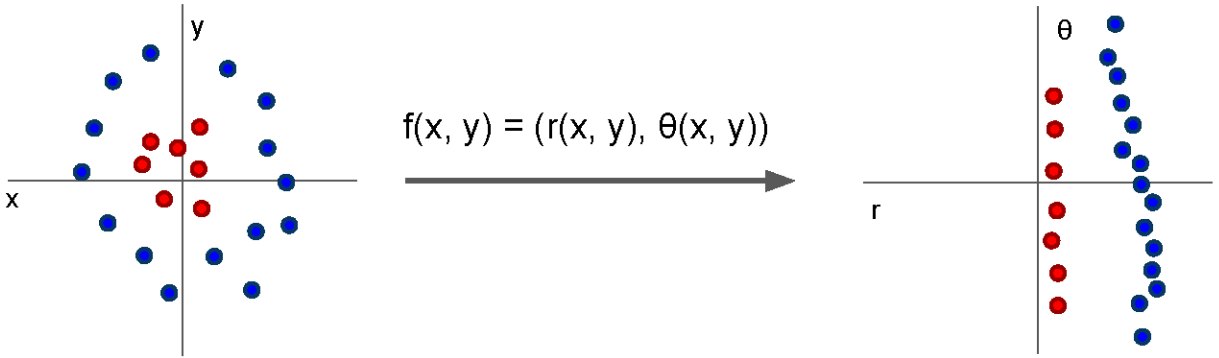
Features de l'image

Features de l'image: motivation



ne peut pas séparer les points rouges et bleus avec un classificateur linéaire

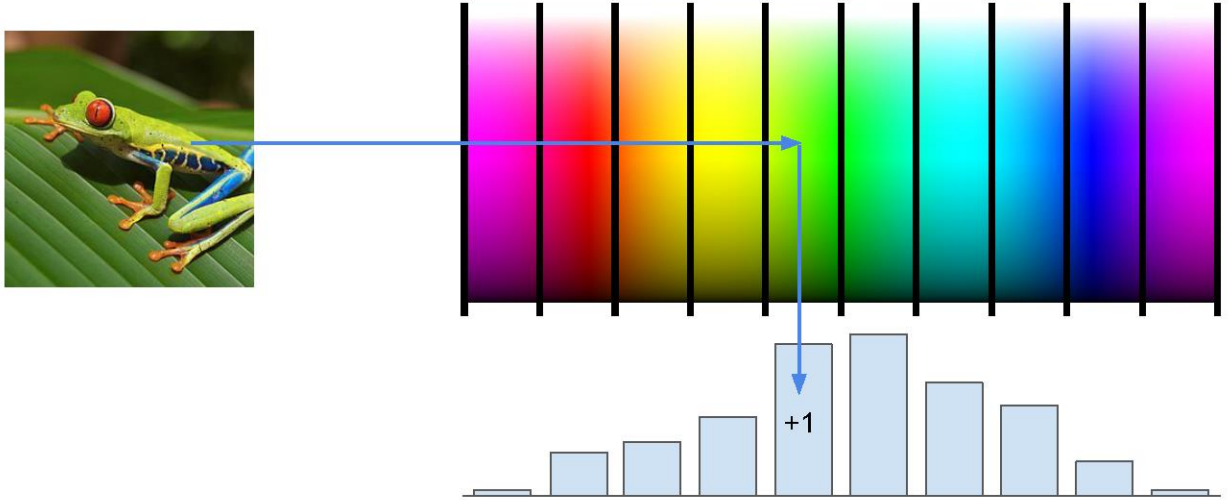
Features de l'image: motivation



ne peut pas séparer les points rouges et bleus avec un classificateur linéaire

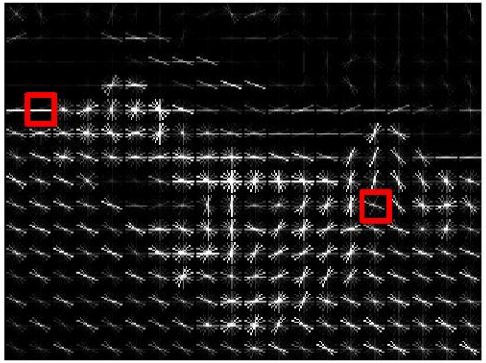
après application de la transformation des features, les points peuvent être séparés par un classificateur linéaire

Exemple : histogramme des couleurs



Exemple : histogramme des gradients orientés

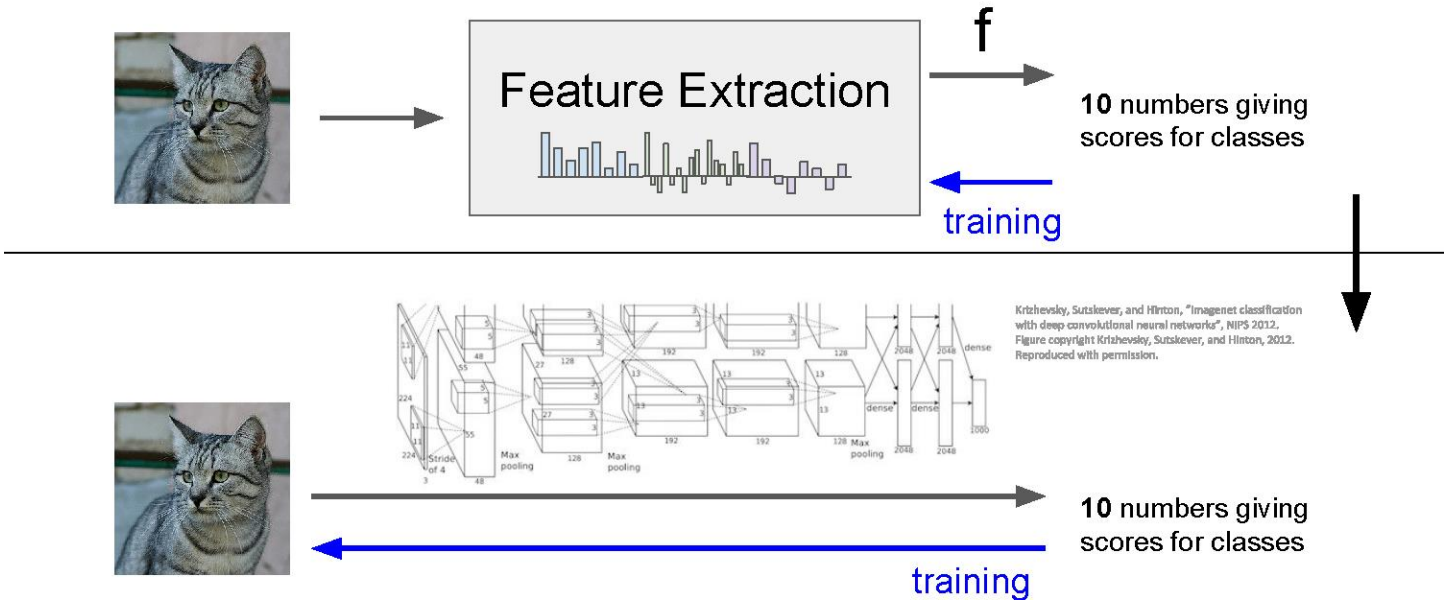
Diviser l'image en régions de 8x8 pixels
Dans chaque région, quantifier la direction des bords en 9 cases



Exemple : L'image 320x240 est divisée en 40x30 cases ; dans chaque case, il y a 9 nombres ; le vecteur de caractéristiques comprend donc 30x40x9=10800 nombres.

Lowe, "Object recognition from local scale-invariant features", ICCV 1999
Dalal and Triggs, "Histograms of oriented gradients for human detection," CVPR 2005

Features de l'image vs ConvNets



La dernière couche de la plupart des CNN est un classificateur linéaire

Pixels d'entrée

Tout exécuter à l'aide d'un grand réseau de neurones, entraîné de bout en bout

Clé : effectuer suffisamment de traitements pour que les classes soient linéairement séparables lorsque l'on arrive à la fin du réseau.

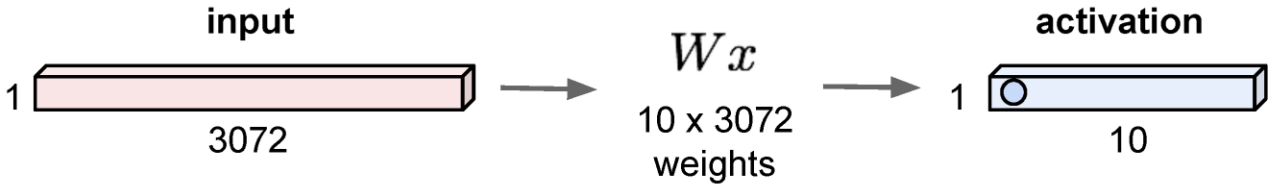
Réseaux de neurones convolutifs

Types de couches :

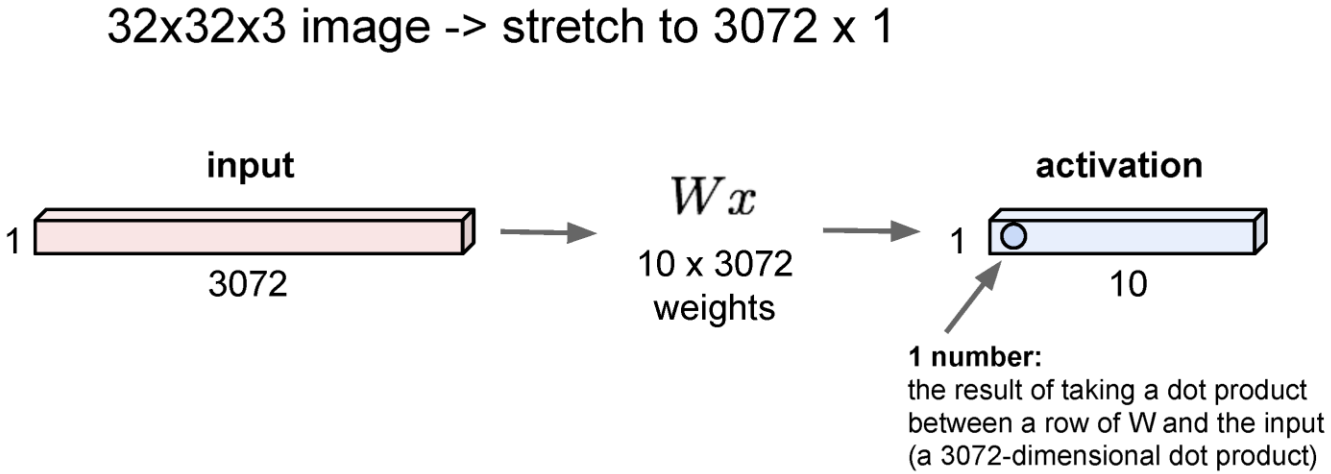
- Couche entièrement connectée
- Couche convolutive
- Pooling layer

Couche entièrement connectée

32x32x3 image -> stretch to 3072 x 1

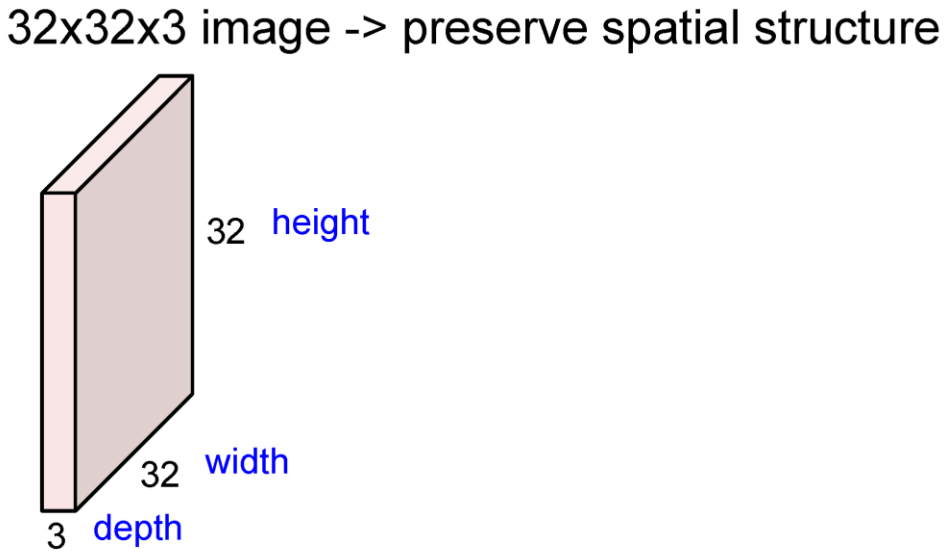


Couche entièrement connectée

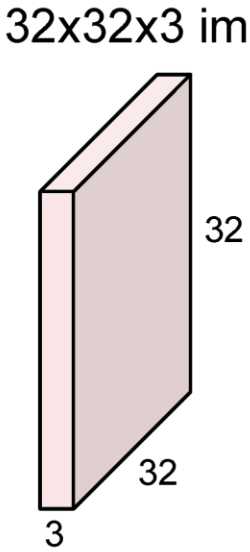


Identique à un classificateur linéaire!

Couche convolutive



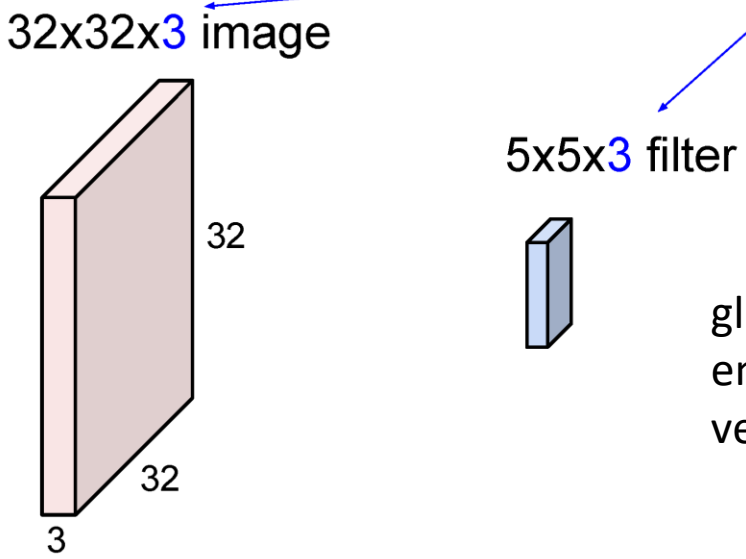
Couche convolutive



glisser le filtre sur l'image dans l'espace, en calculant les produits scalaire de deux vecteurs

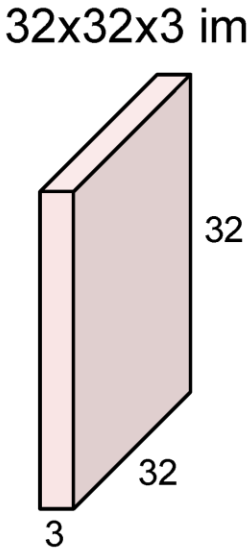
Couche convolutive

les filtres s'étendent toujours sur toute la profondeur du volume d'entrée



glisser le filtre sur l'image dans l'espace, en calculant les produits scalaire de deux vecteurs

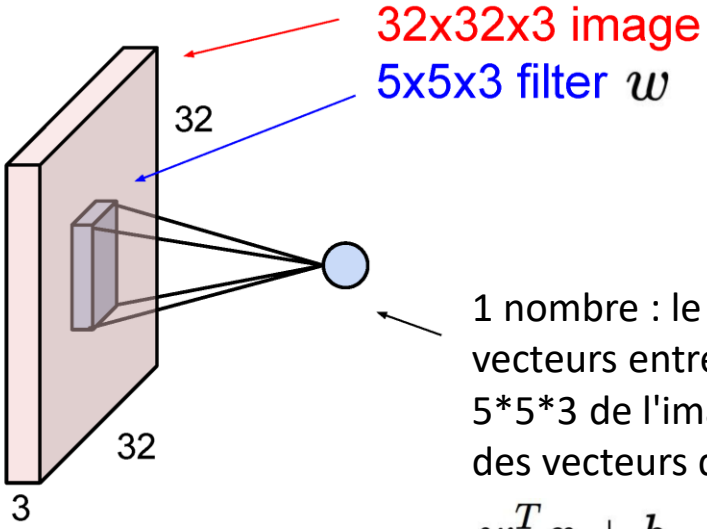
Couche convolutive



glisser le filtre sur l'image dans l'espace, en calculant les produits scalaire de deux vecteurs

Nombre de poids : 5 x 5 x 3 + 1 = 76(contre 3072 pour une couche entièrement connectée)(+1 pour le biais)

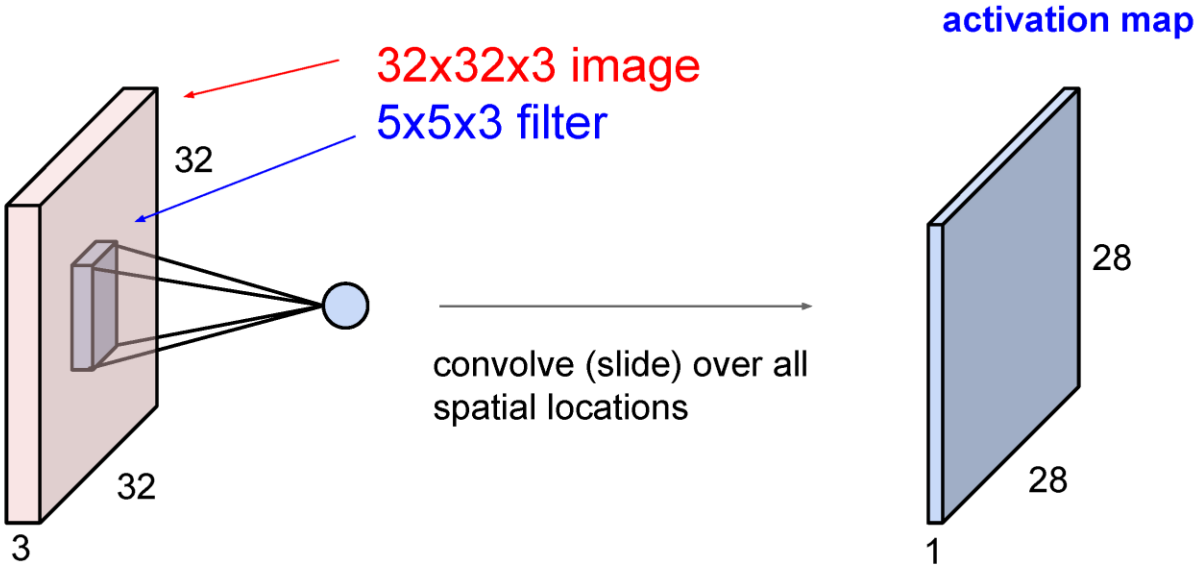
Couche convolutive



1 nombre : le résultat du produit scalaire des vecteurs entre un filtre et un petit morceau de 5*5*3 de l'image (c.-à-d. 5x5x3 = produit scalaires des vecteurs de 75 dimensions + biais).

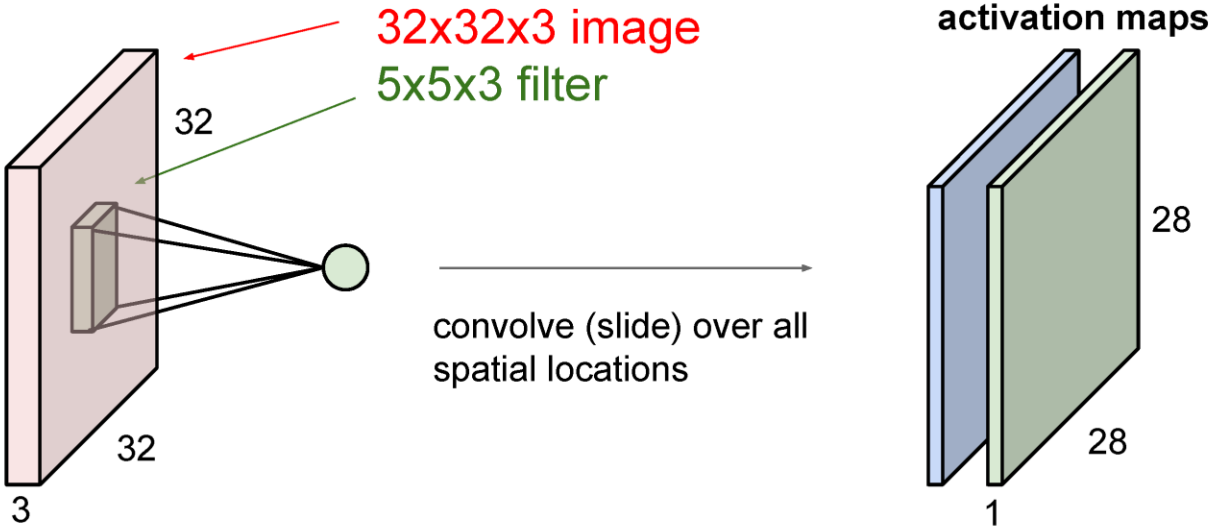
$$w^T x + b$$

Couche convolutive

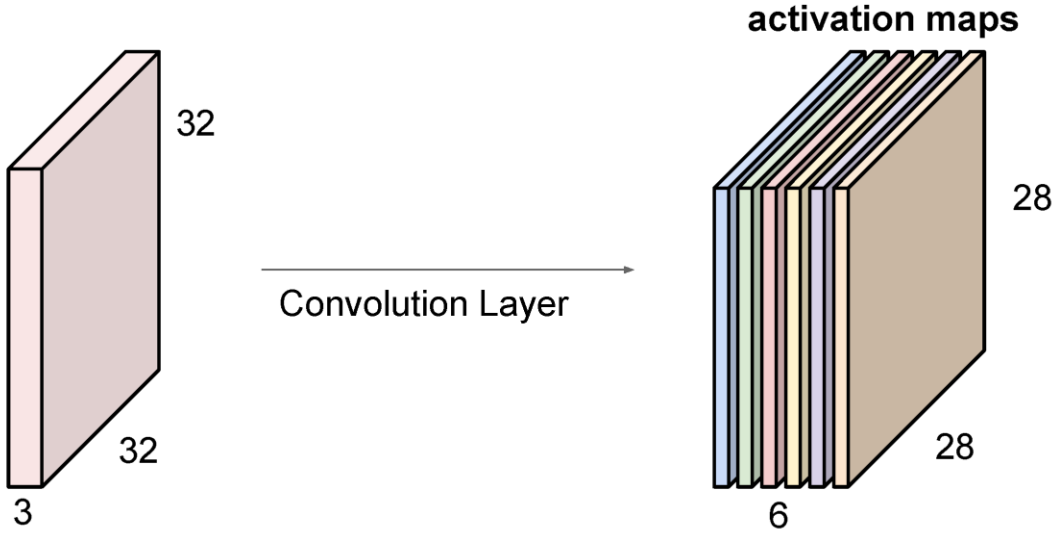


Couche convolutive

consider a second, green filter



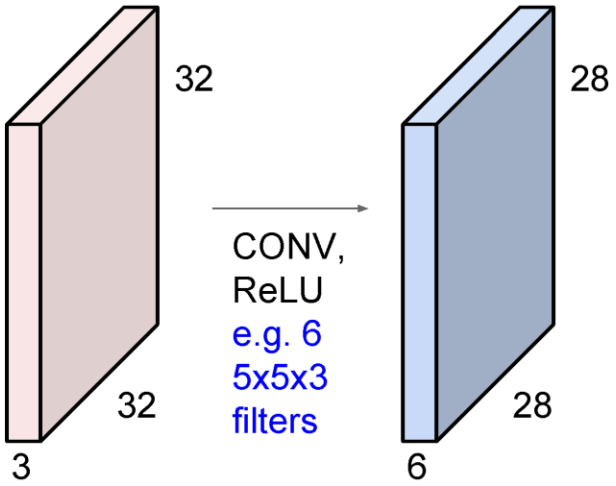
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



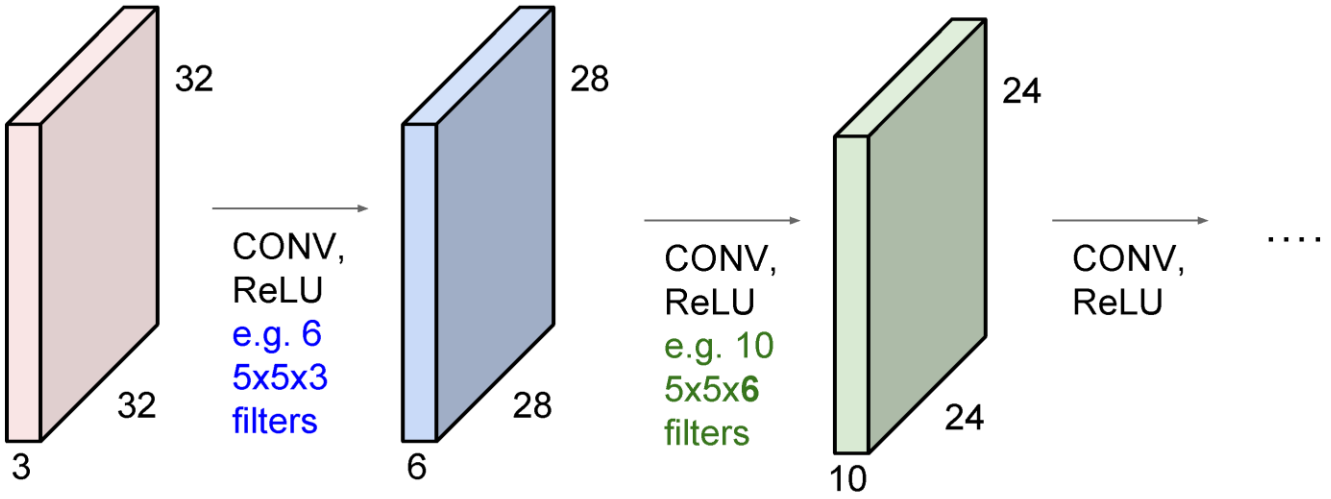
We stack these up to get a "new image" of size 28x28x6!

(nombre total de paramètres : 6 x (75 + 1) = **456**)

ConvNet est une séquence de couches de convolution, entrecoupées de fonctions d'activation.



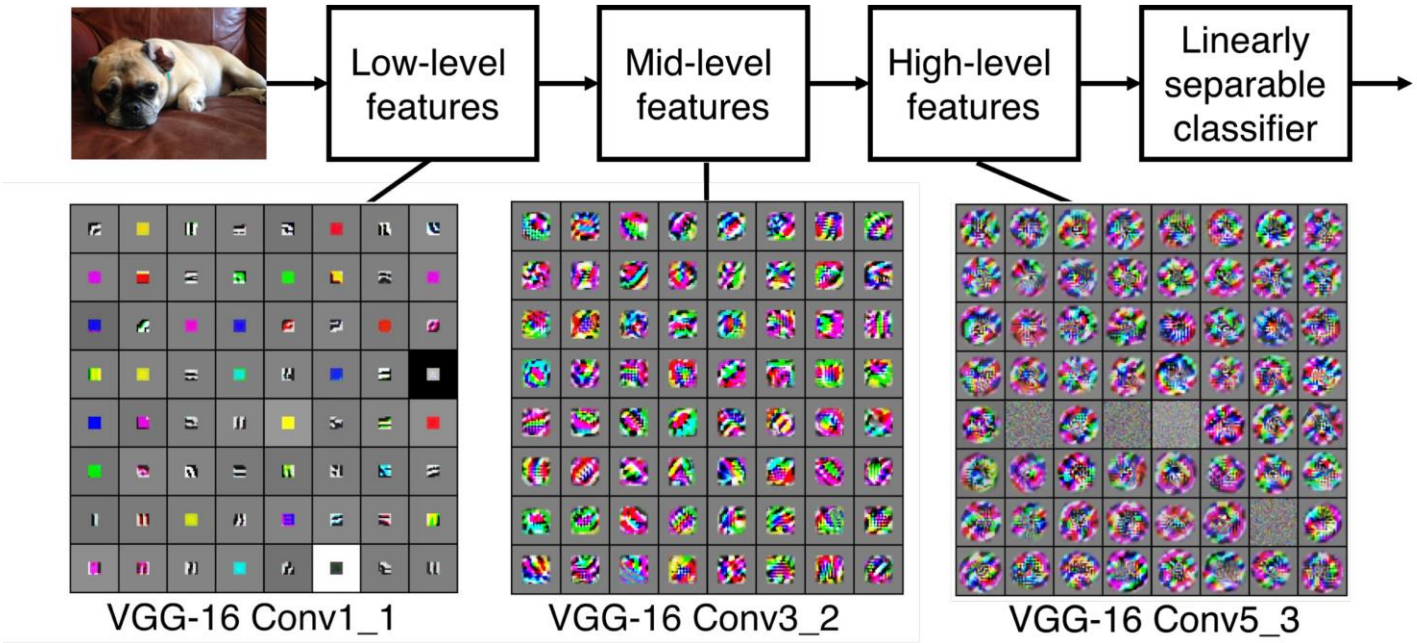
ConvNet est une séquence de couches de convolution, entrecoupées de fonctions d'activation.



Preview

[Zeiler and Fergus 2013]

Visualization of VGG-16 by Lane McIntosh. VGG-16 architecture from [Simonyan and Zisserman 2014].



Propriétés de la couche convolutive

UNIVERSITÉ CÔTE D'AZUR

- Nombre réduit de paramètres à apprendre par rapport à une couche entièrement connectée
- Préserve la structure spatiale - la sortie d'une couche convolutive a la forme d'une image.

[ConvNetJS demo: training on CIFAR-10]

ConvNetJS CIFAR-10 demo

Description

This demo trains a Convolutional Neural Network on the CIFAR-10 dataset in your browser, with nothing but Javascript. The state of the art on this dataset is about 90% accuracy and human performance is at about 94% (not perfect as the dataset can be a bit ambiguous). I used this python script to parse the original files (python version) into batches of images that can be easily loaded into page DOM with img tags.

This dataset is more difficult and it takes longer to train a network. Data augmentation includes random flipping and random image shifts by up to 2px horizontally and verically.

By default, in this demo we're using Adadelta which is one of per-parameter adaptive step size methods, so we don't have to worry about changing learning rates or momentum over time. However, I still included the text fields for changing these if you'd like to play around with SGD+Momentum trainer.

Report questions/bugs/suggestions to @karpathy.

https://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html

Vue d'ensemble

- Un réseau neuronal convolutif peut être considéré comme une fonction allant des images aux scores des classes
- Avec des millions de poids réglables...
- ... conduisant à un mappage très non linéaire entre les images et les caractéristiques / scores de classe.
- Nous réglerons ces poids en fonction de la précision de la classification sur les données d'entraînement...
- ... et nous espérons que notre réseau se généralisera à de nouvelles images au moment du test.

Vue d'ensemble

- Un réseau neuronal
  fonction allant des i
- Avec des millions de
- ... conduisant à un r
  les caractéristiques
- Nous réglerons ces
  classification sur les
- ... et nous espérons
  nouvelles images au