# Fonctionnement d'un système de classification à l'aide de réseaux de neurones

William PUECH – Université de Montpellier

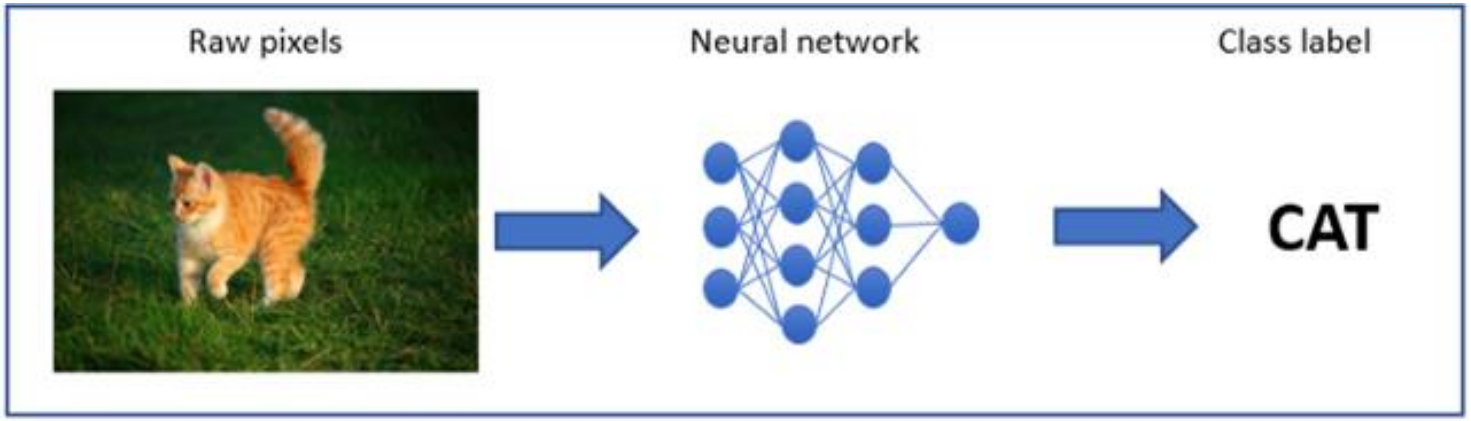
# Sommaire

- Fonctionnement d’un réseau de neurones
- Correction d’un réseau
- Réseaux de neurones à convolution
- Implémentation et Utilisation

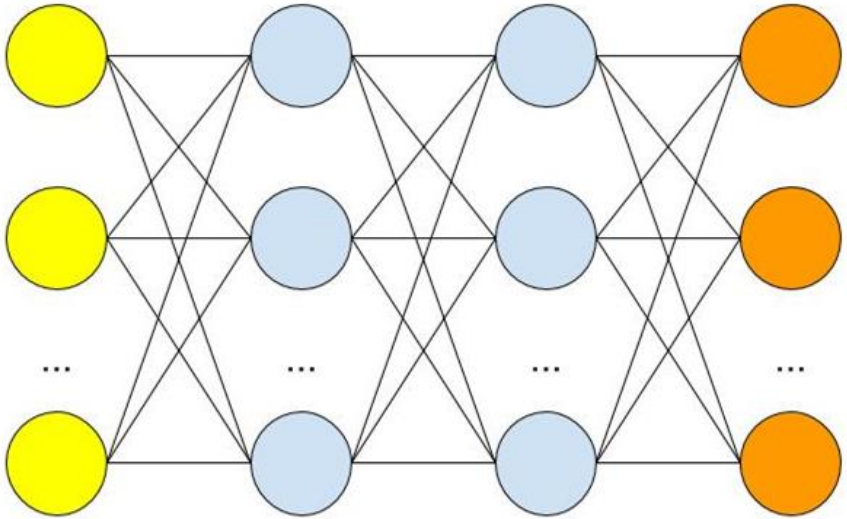
# 1

# Réseaux de neurones

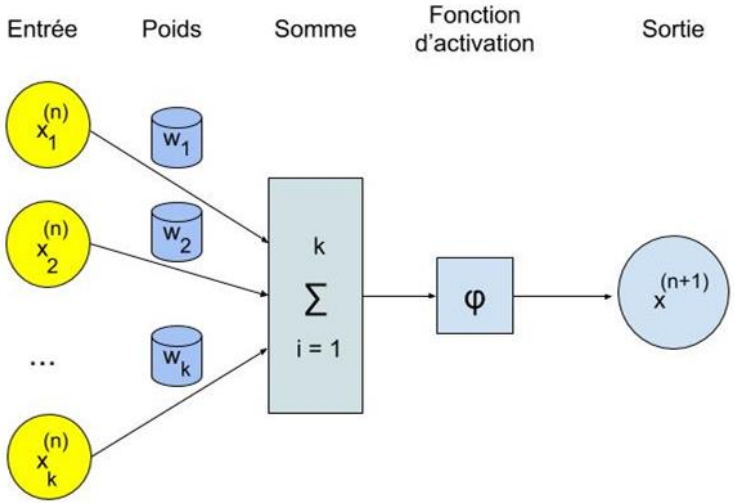
# Système de classification



# Architecture d’un réseau de neurones

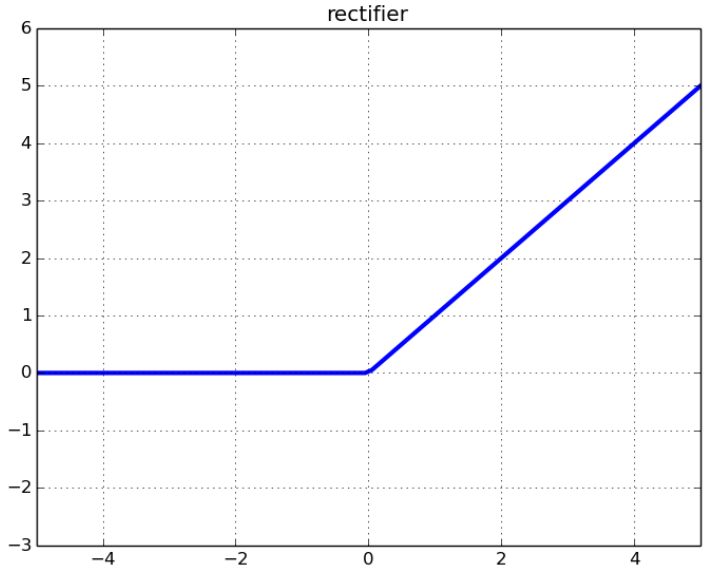


# Fonctionnement d'un neurone




$$x_j^{(n+1)} = g^{(n+1)}(h_j^{(n+1)}) = g^{(n+1)}(\sum_{k=1}^{K} w_{jk}^{(n+1)} x_k^{(n)})$$

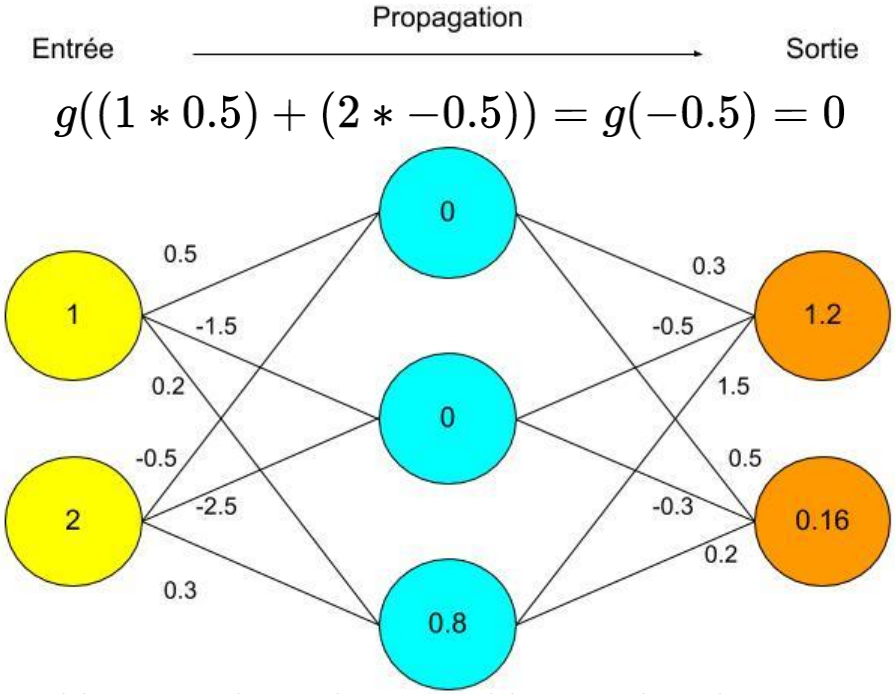
# La fonction ReLU




$$g(x) = x^{+} = \max(0, x)$$

$$g'(x) = \begin{cases} 1 \text{ si } x > 0 \\ 0 \text{ sinon} \end{cases}$$

# Propagation

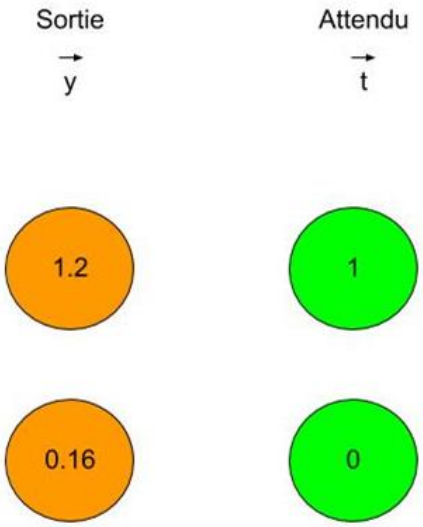


# 2

# Rétropropagation

## Améliorer notre modèle

# Calcul des erreurs



# Calcul des erreurs

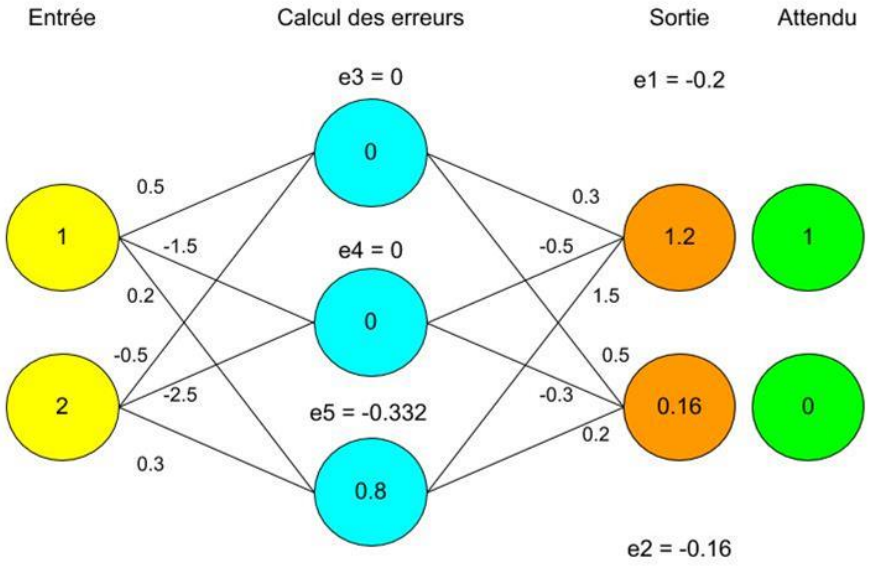
Erreur de sortie

$$e_i^{sortie} = g'\left(h_i^{sortie}\right)\left[t_i - y_i\right]$$

Erreur sur les couches inférieures

$$e_j^{(n-1)} = g'^{(n-1)}\left(h_j^{(n-1)}\right)\sum_i w_{ij} e_i^{(n)}$$

# Calcul des erreurs




$$e1 = g'(1.2) * (1 - 1.2)$$
$$e1 = 1 * (-0.2) = -0.2$$

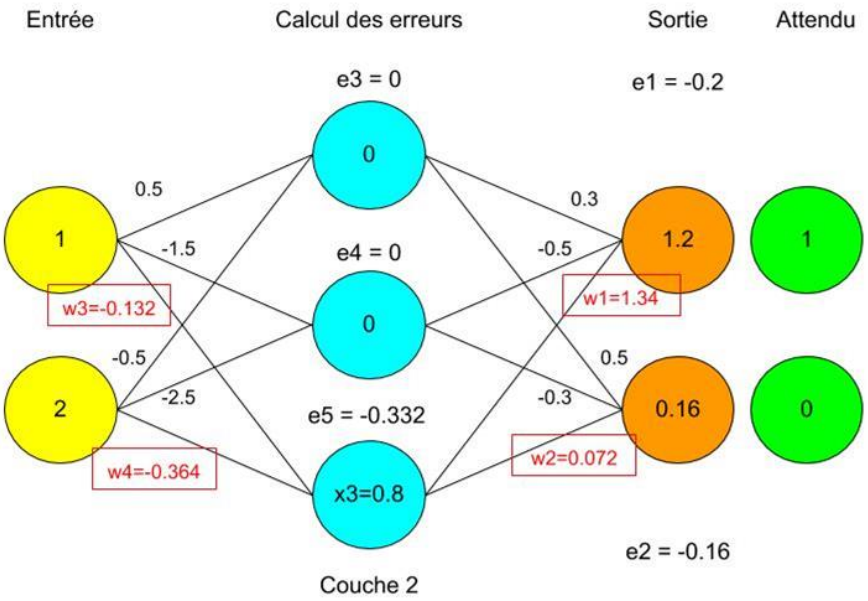
$$e5 = g'(0.8) * [(1.5 * (-0.2)) + (0.2 * (-0.16))]$$
$$e5 = 1 * [-0.3 - 0.032] = -0.332$$

# Correction des poids

$$w_{ij}^{(n)} = w_{ij}^{(n)} + \lambda e_i^{(n)} x_j^{(n-1)}$$

Avec λ le taux d'apprentissage

# Mise à jour des poids



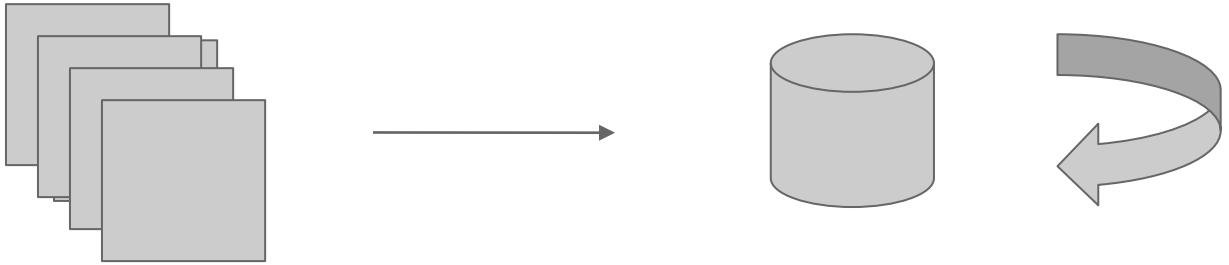

$$w_1 = w_1 + \lambda * e_1 * x_3^{(2)}$$

$$w_1 = 1.5 + 1 * (-0.2) * 0.8$$

$$w_1 = 1.5 - 0.16 = 1.34$$

Ici, λ est égal à 1

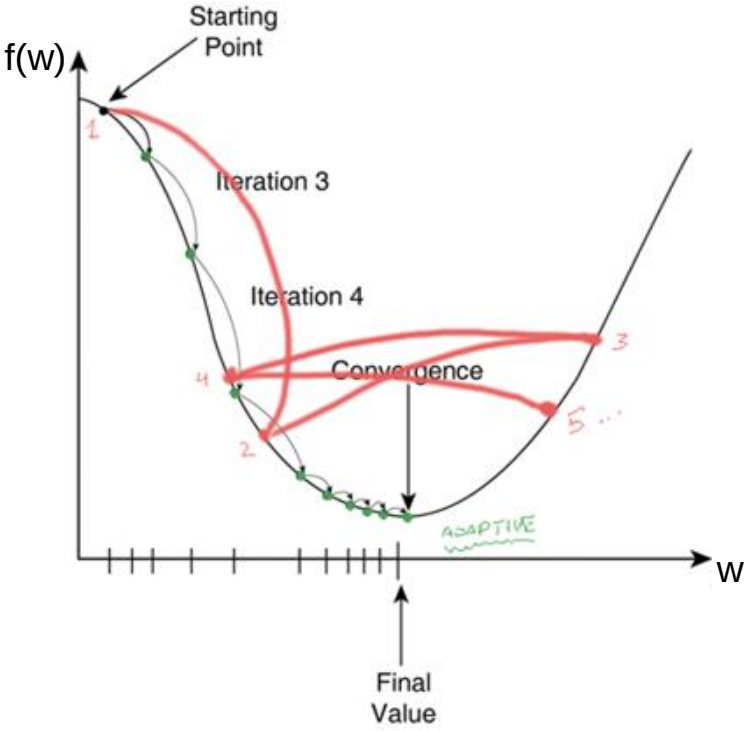
# Implémentation



Données Modèle Mise à jour

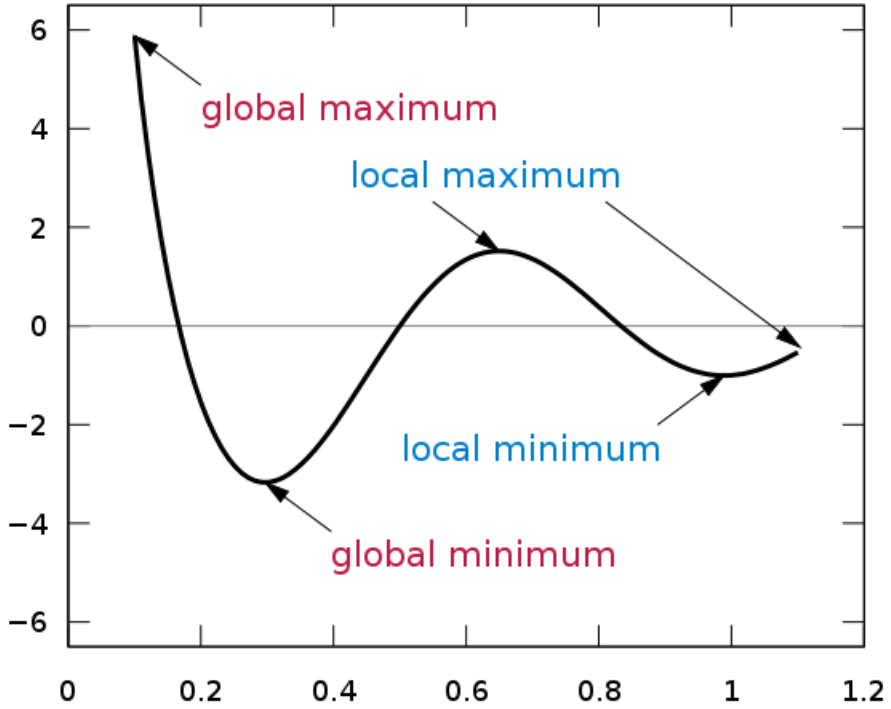
$$\overline{w} = \frac{1}{t} \sum_{i=0}^{t-1} w_i$$

# Convergence de l'algorithme




https://qph.fs.quoracdn.net/main-qimg-f3972c89625c0451f0ea92a4c0ea0728

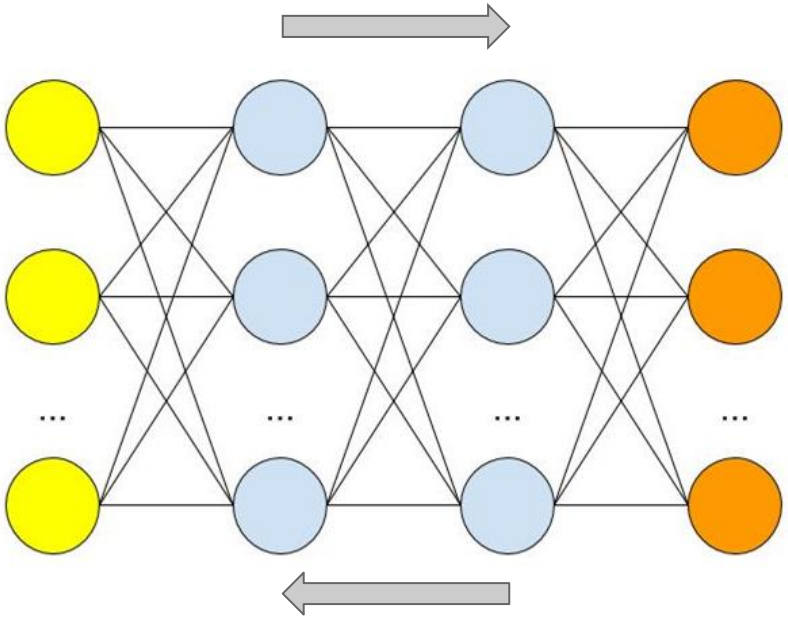
# Problème lors de la recherche




https://en.wikipedia.org/wiki/Backpropagation#/media/File:Extrema_example.svg

# Résumé

Propagation



Rétro-propagation

# 3

# Réseau de neurones à convolution

Classifier des images

# Architecture

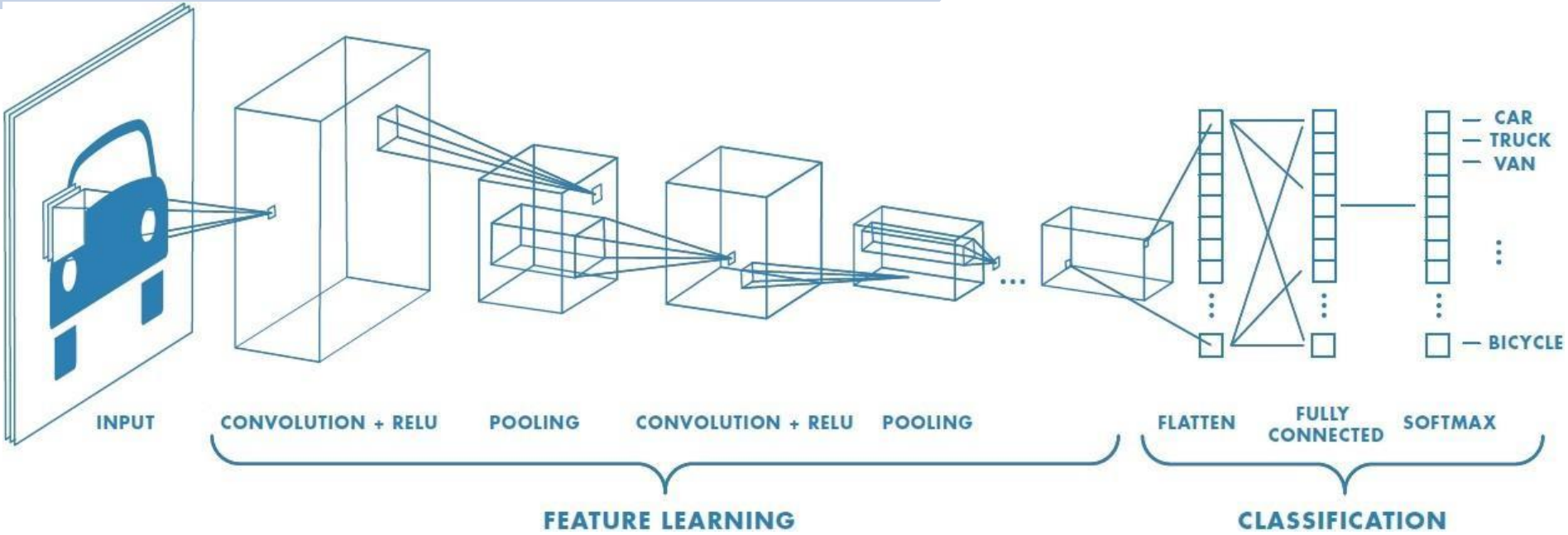
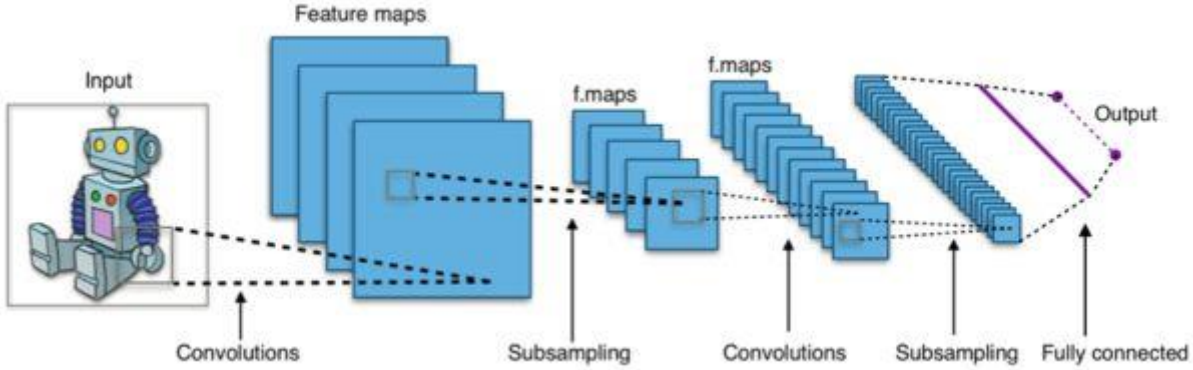





Illustration de “The MathWorks” Disponible sur : https://www.mathworks.com/discovery/convolutional-neural-network.html

# Réseau de neurones à convolution (CNNs)

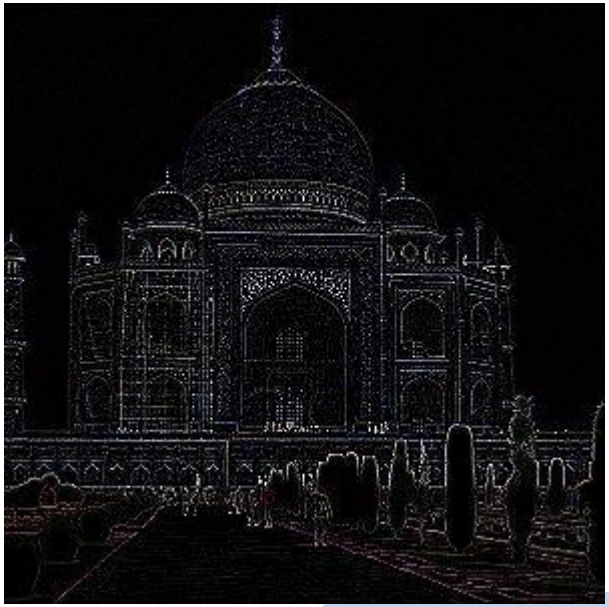
- Couche de Convolution
- Couche de Pooling
- Couche Entièrement Connectée



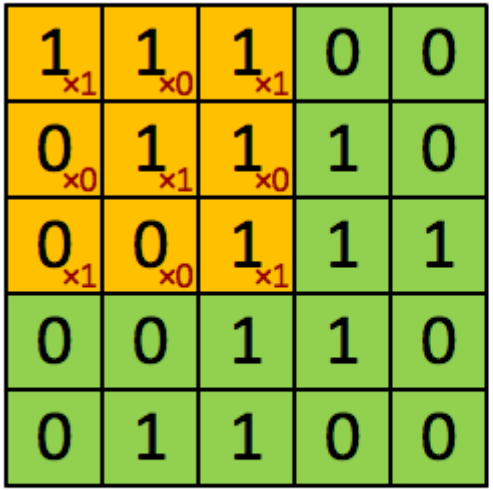
# Qu’est ce qu’une convolution ?

*

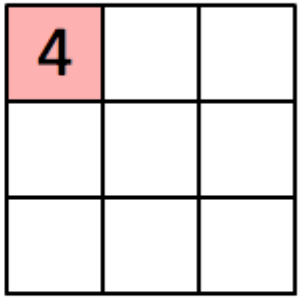
| | | | |
|---|---|---|---|
| 0 | 1 | 0 | |
| 1 | -4 | 1 | |
| 0 | 1 | 0 | |



# Couche de convolution




Image




Convolved Feature

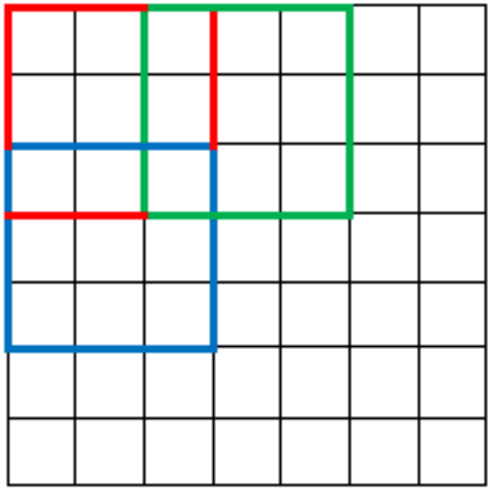
$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

# Paramètres de la couche de convolution

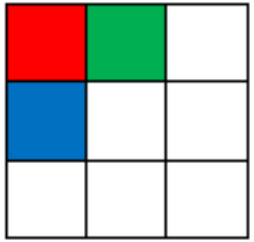
- Le pas de contrôle
- La marge

# Le pas de contrôle

7 x 7 Input Volume



3 x 3 Output Volume

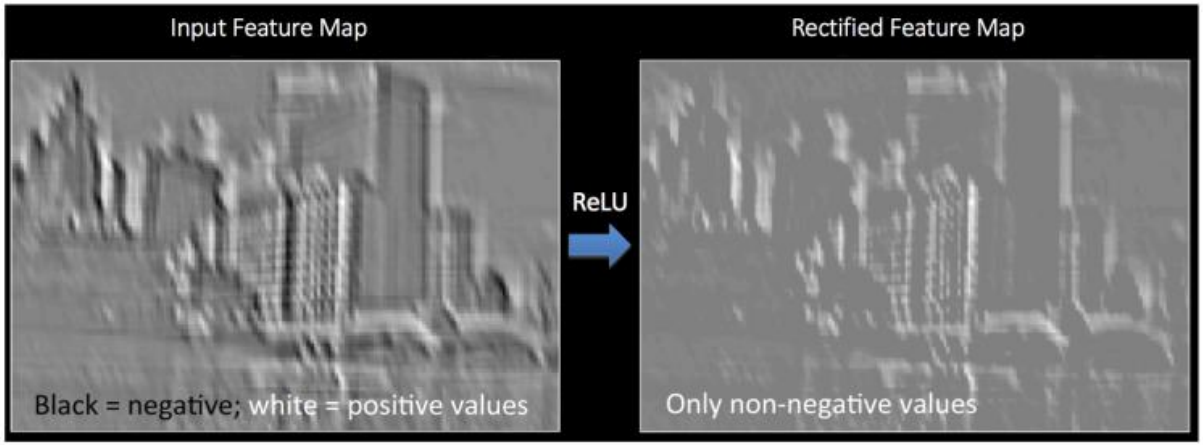


# La marge

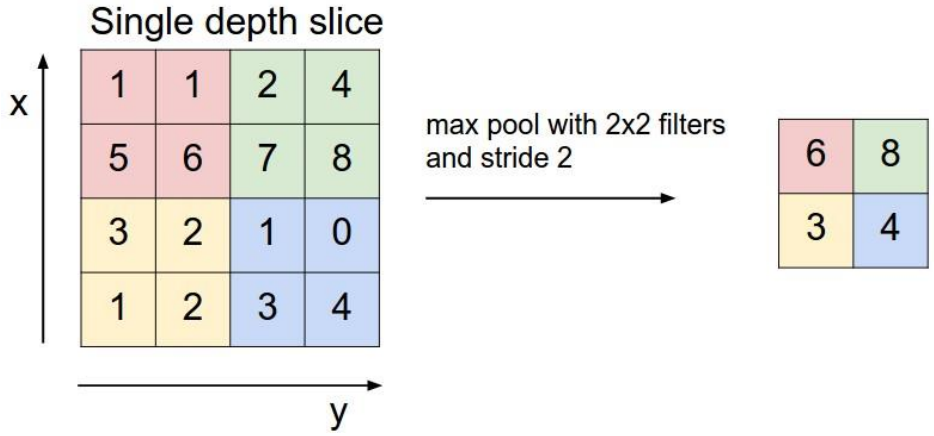
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 18 | 54 | 51 | 239 | 244 | 188 | 0 |
| 0 | 55 | 121 | 75 | 78 | 95 | 88 | 0 |
| 0 | 35 | 24 | 204 | 113 | 109 | 221 | 0 |
| 0 | 3 | 154 | 104 | 235 | 25 | 130 | 0 |
| 0 | 15 | 253 | 225 | 159 | 78 | 233 | 0 |
| 0 | 68 | 85 | 180 | 214 | 245 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

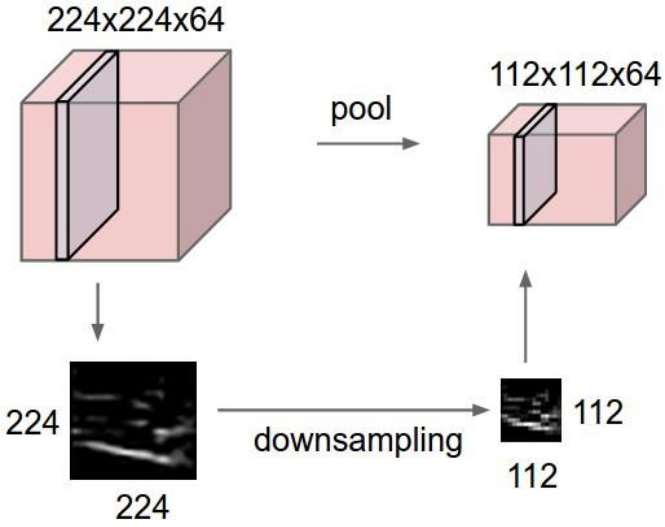
# ReLU - Rectified Linear Unit

$$g(x) = x^{+} = \max(0, x)$$

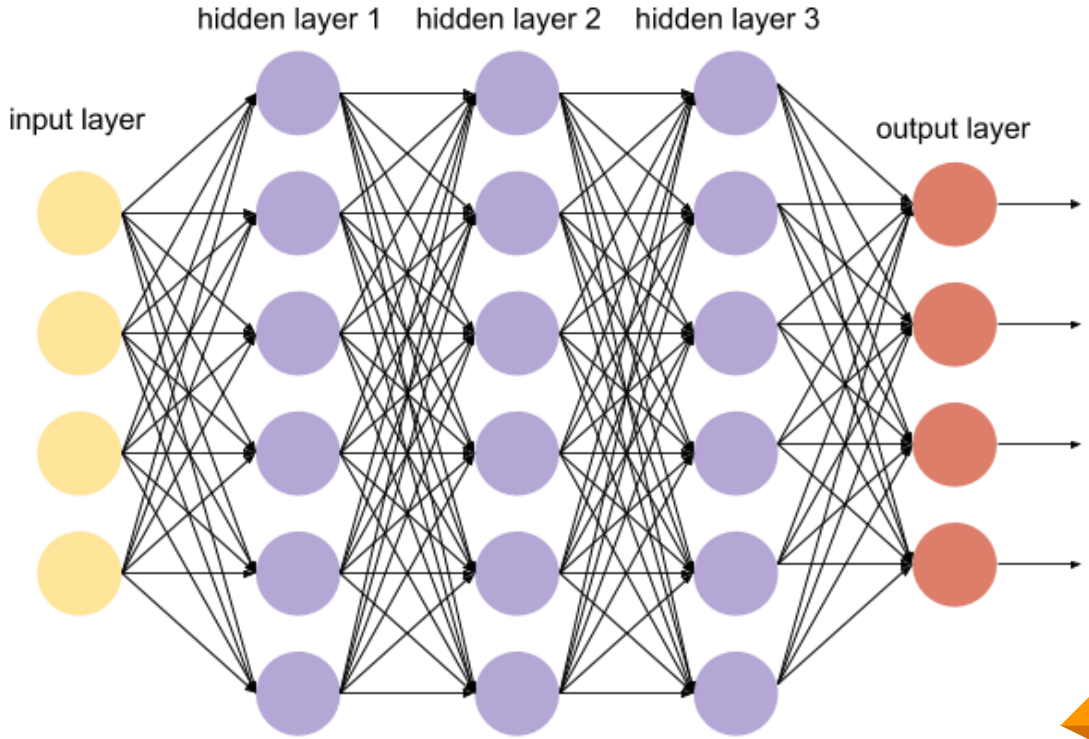


# Couche de Pooling



# Couche entièrement connectée



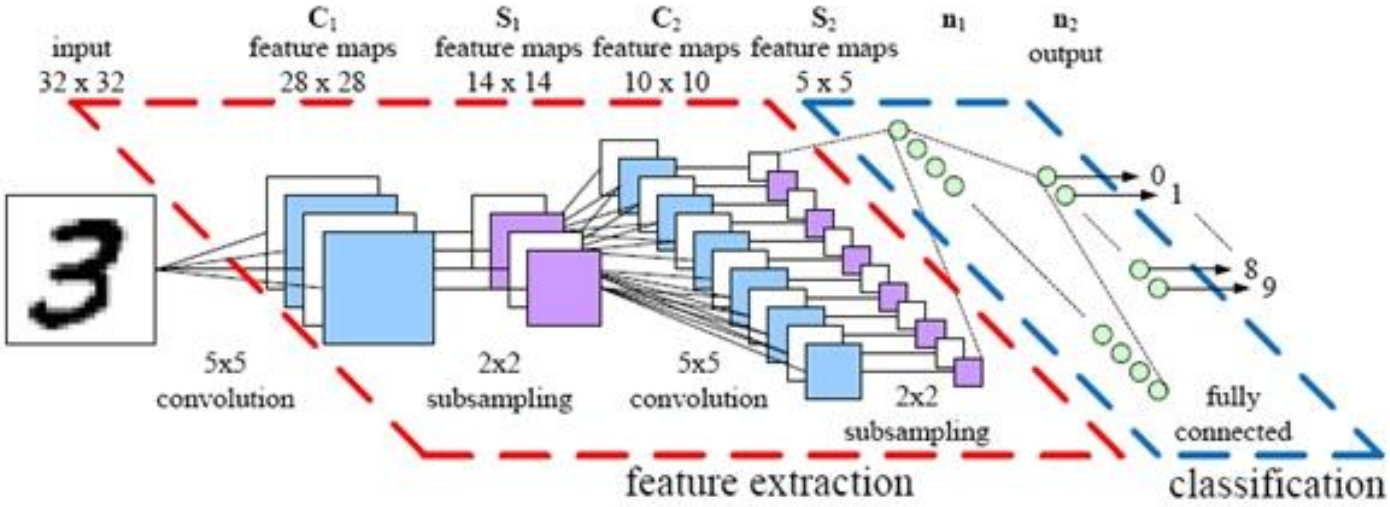
# Fonction Softmax

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ pour tout } j \in \{1, \dots, K\}.$$

z = (1 ; 3 ; 2,5 ; 5 ; 4; 2)

σ(z) = (0,011 ; 0,082 ; 0,050 ; 0,605 ; 0,222 ; 0,030)

# Récapitulatif



# 4

# Implémentation et Utilisation

# Fonctionnement de l’apprentissage de notre système

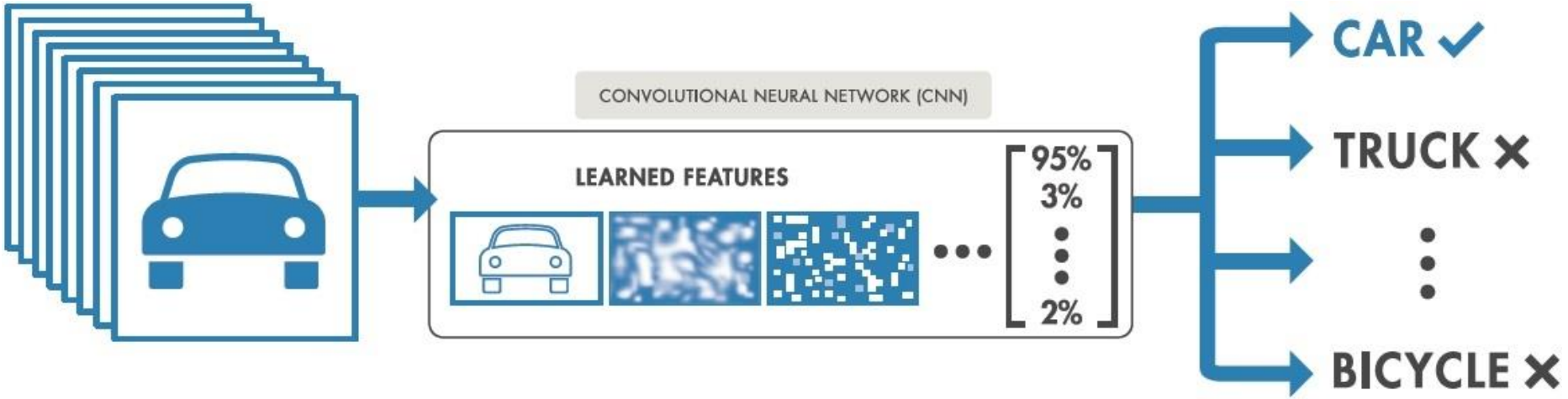
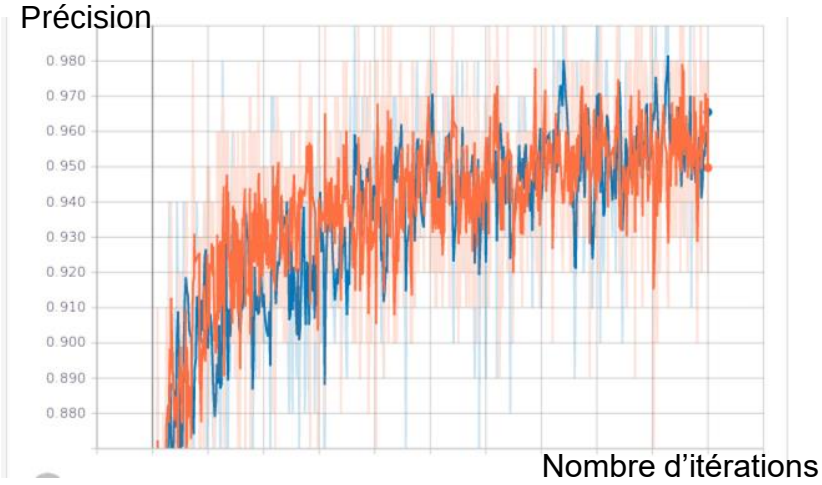



Illustration de “The MathWorks” Disponible sur : https://www.mathworks.com/discovery/convolutional-neural-network.html

# Visualisation avec TensorBoard

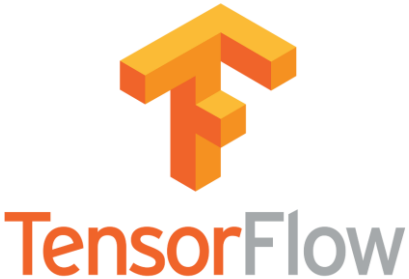



- Phase d'entraînement (1h30)
- Phase de validation (30 min)
- Phase d'utilisation (0.05 seconde)

# Technologies utilisées


Langage de programmation

Librairie traitement d’images
Gestions des fichiers




Librairie Python
Développée par Google

Création de système de
réseaux de neurones


Compute Unified Device
Architecture

Calcul parallèle sur
processeurs graphiques

# Inception v3

- Modèle pré-entraîné sur plus de 2 millions d'images sur ImageNet
- Performant sur la phase d'extraction des informations 86% de précision

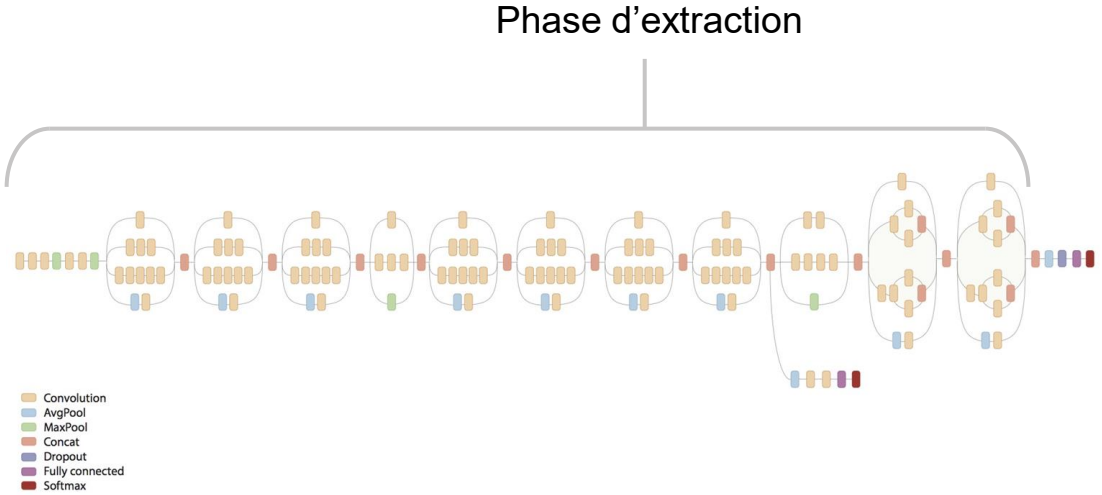



*Schéma de l'architecture de Inception v3*

ustration disponible sur : https://towardsdatascience.com/multi-label-image-classification-with-inception-net-cbb2ee538e30

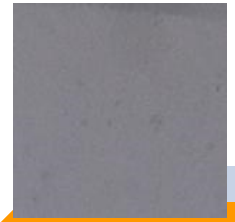
# Cas d’application - Détection de fourmis

# Cas d’application - Détection de fourmis

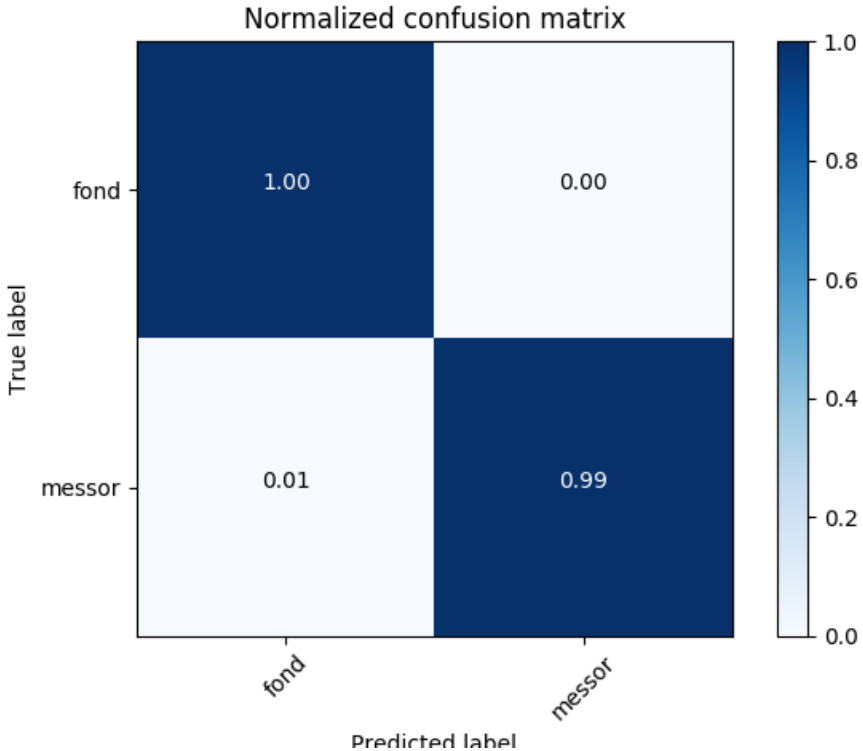
# Cas d’application - Détection de fourmis

Messor

Fond



# Validation de notre classification de fourmis



# Détection de fourmis

Image originale


Image originale superposée par une carte de chaleur

# Bibliographie


https://fr.wikipedia.org/wiki/R%C3%A9seau_neuronal_convolutif
https://fr.wikipedia.org/wiki/R%C3%A9seau_de_neurones_artificiels
https://www.tensorflow.org/
https://arxiv.org/pdf/1512.00567.pdf
https://www.infor.uva.es/~teodoro/neuro-intro.pdf
https://towardsdatascience.com/multi-label-image-classification-with-inception-net-cbb2ee538e30
https://www.mathworks.com/discovery/convolutional-neural-network.html
https://www.tensorflow.org/tutorials/layers
https://fr.wikipedia.org/wiki/R%C3%A9tropropagation_du_gradient

# Merci !

Avez-vous des questions ?