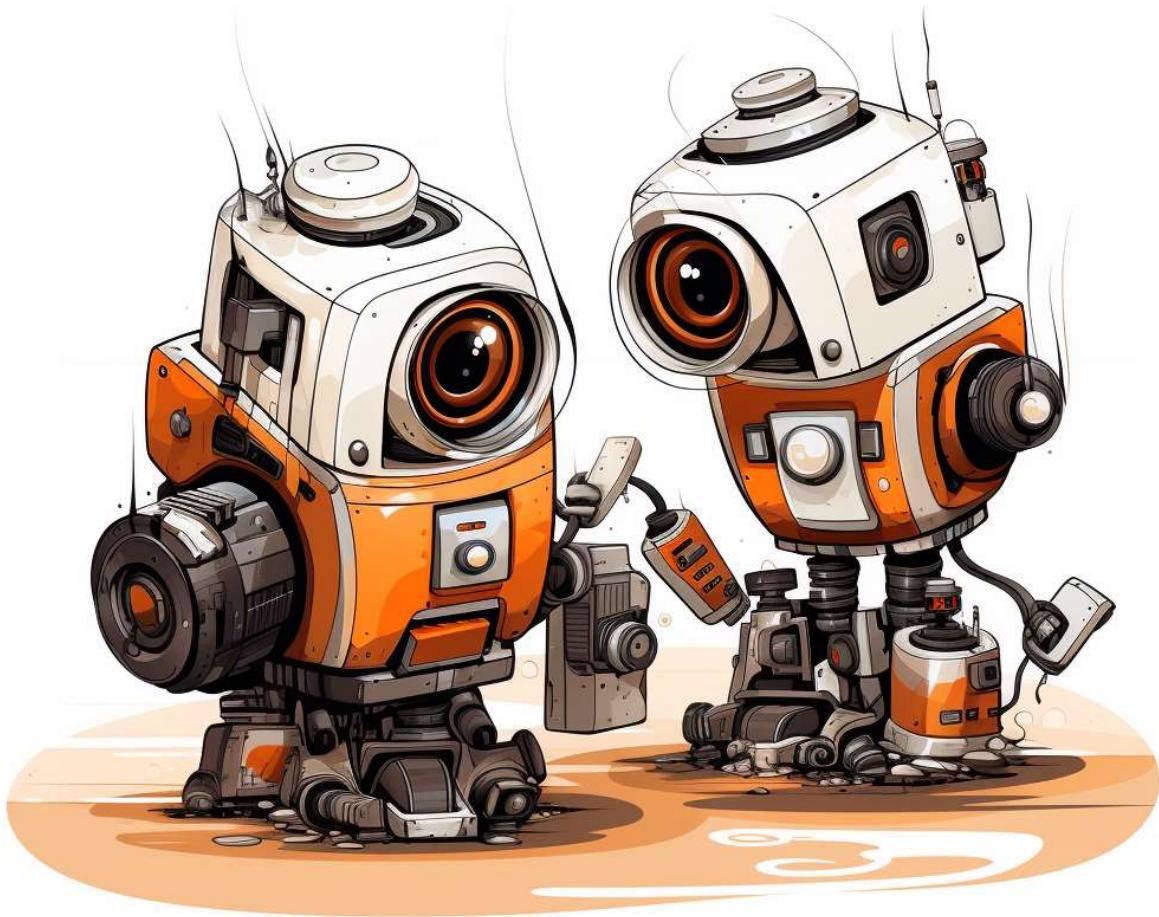




SRI 3A

Vision par ordinateur



philippe.joly@univ-tlse3.fr

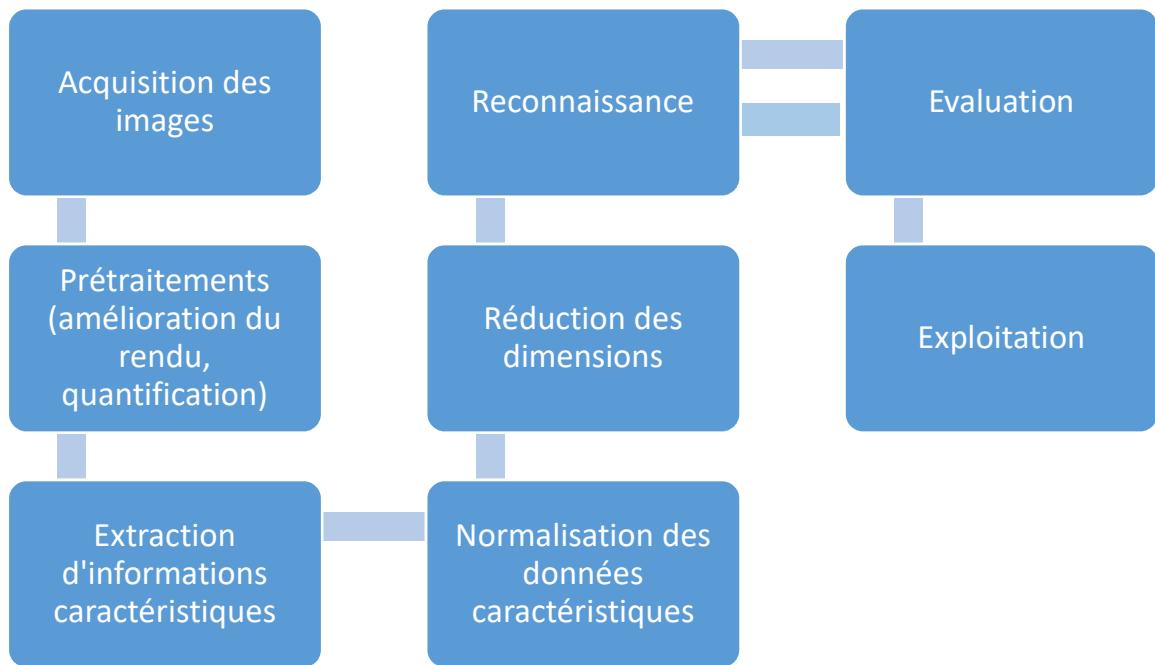
Table des matières

I.	Introduction.....	4
A.	Schéma général	4
B.	Objectif de la reconnaissance.....	4
1.	Classification.....	4
2.	Détection.....	4
3.	Identification	4
4.	Vérification	4
5.	Localisation.....	4
6.	Segmentation	4
7.	Regression	4
C.	Choix d'une stratégie de reconnaissance.....	5
1.	Fusion précoce	5
2.	Fusion tardive	5
3.	Méthodes par apprentissage	5
D.	Evaluation des résultats	5
1.	Base d'apprentissage (BA).....	5
2.	Précision	6
3.	Rappel.....	6
4.	F-mesure.....	6
5.	Accuracy (justesse)	6
6.	Intersection over Union (IoU).....	6
7.	Baseline	6
II.	Extraction d'informations caractéristiques	7
A.	Caractérisation de régions.....	7
1.	Couleur	7
2.	Texture	7
3.	Forme	17
B.	<i>Points d'intérêt</i>	20
1.	Principe général.....	20
2.	Détection des points d'intérêt	21
3.	Descripteurs locaux	24
III.	Adaptation des données	25
A.	Normalisation	25
B.	Réduction des dimensions.....	26
1.	Analyse en Composantes Principales	26

2.	Décomposition en Valeurs Singulières	26
3.	Feature selection.....	26
4.	Autres possibilités	26
IV.	Méthodes de reconnaissance.....	27
A.	Arbres de décision	27
B.	Random Forest	27
1.	Apprentissage.....	27
2.	Reconnaissance	28
C.	Boosting.....	28
1.	Principes généraux	28
2.	Exemple : ADABOOST (Boosting adaptatif).....	28
3.	Reconnaissance	29
4.	Mise en œuvre de l'apprentissage	29
5.	Cascades de Haar.....	31
D.	Classification Bayesienne	34
E.	Votes.....	34
1.	Objectifs et principes.....	34
2.	Transformée de Radon.....	37
3.	Extension aux formes circulaires.....	38
4.	Transformée de Hough généralisée	39
F.	Transformée en distance.....	42
1.	Distance, voisinage et connexité.....	42
G.	Graphes et Méthodes s'appuyant sur la logique	50
H.	Support Vector Machine	50
I.	Réseaux de neurone.....	50
1.	Réseaux de Neurones Numériques	50
2.	Architecture standard et mise en œuvre	52
3.	Choix des paramètres.....	52
4.	Neurones spécialisés	54
5.	Architectures spécialisées	54
6.	Architectures référencées en classification de concept dans les images	56
7.	Bases d'apprentissage	58
8.	Réseaux de Neurones à Spike.....	59

I. Introduction

A. Schéma général



B. Objectif de la reconnaissance

1. Classification

A quelle classe appartient l'observation ?

2. Détection

Oui ou non la classe d'objet C est-elle observable ?

3. Identification

Quelle est l'identité de l'individu observable ?

4. Vérification

Oui ou non l'observation appartient-elle à une classe donnée ?

5. Localisation

Où se trouve l'objet à détecter ?

6. Segmentation

Quelle est la région composée des pixels représentant l'objet, ou l'intervalle de temps pendant lequel un objet est visible ?

7. Regression

L'objectif est de prédire une mesure (continue) associée à une observation

C. Choix d'une stratégie de reconnaissance

1. Fusion précoce

Les caractéristiques sont concaténées les unes à la suite des autres (même si elles ne sont pas homogènes) et envoyées à un unique système de reco.

2. Fusion tardive

Plusieurs systèmes de reco primaires sont opérés sur tout ou partie des caractéristiques et mis en concurrence. Un système de reco secondaire reçoit en entrée les décisions prises par les systèmes de reco primaires ou intermédiaires (exemple : boosting, forêts aléatoires, etc)

3. Méthodes par apprentissage

Les méthodes par apprentissage reposent en général sur 2 algorithmes duaux : un algorithme calcule un « modèle » du concept, l'autre utilise ce modèle pour reconnaître un concept. L'erreur produite par le second algorithme est parfois utilisée par le premier algorithme pour améliorer le modèle.

a) Notion de « modèle »

Un algorithme construit une représentation numérique abstraite des concepts à reconnaître. Cette représentation numérique est appelée modèle. Elle est utilisée ensuite par un autre algorithme pour reconnaître le concept.

b) Notion d'état caché

Lorsqu'un modèle représente un concept sous la forme d'un graphe avec un « début » et une « fin », les états intermédiaires peuvent représenter un élément d'information explicite ou non. Lorsque ce n'est pas le cas, on parle d'états cachés – c'est l'algorithme d'apprentissage qui décide de leur signification (souvent abstrait et incompréhensible pour un humain).

c) Méthodes supervisées

Utilisation de données étiquetées pour le calcul du modèle.

d) Méthodes semi-supervisées

Utilisation de quelques étiquettes pour amorcer l'apprentissage et traiter les cas ambigus (ie : proches des frontières de décision – cas de l'apprentissage « actif »)

e) Méthodes non supervisées

Aujourd'hui, il s'agit souvent d'identifier des classes par clustering ou de produire des « embeddings » (ou représentations abstraites discriminantes)

f) Méthodes auto supervisées

Même objectif. Apprentissage sur des paires positives (une image et la même image transformée) et des paires négatives (2 images différentes). L'objectif est que le système de reco produise la même sortie pour les paires positives (erreur nulle) et une sortie distincte pour les paires négatives.

D. Evaluation des résultats

1. Base d'apprentissage (BA)

BA={(x_i, y_i)} où x_i est le $i^{\text{ème}}$ exemple et y_i est le label (= classe = annotation) de cet exemple.
L'ensemble des annotations est appelé « Vérité Terrain ».

Problèmes classiques :

- Déséquilibre des exemples positifs et des exemples négatifs
- Redondance (conduisant à du sur-apprentissage)
- Faiblesse de la représentation du « reste du monde »
- Coût / Temps nécessaire / faible intérêt de la tâche d'annotation
- Annotations peu fiables
- Volumétrie insuffisante

2. Précision

TP = vrai positif, TN = vrai négatif, FP = Faux positif, FN = faux négatif

$$Précision = \frac{TP}{TP + FP}$$

3. Rappel

$$Rappel = \frac{TP}{TP + FN}$$

4. F-mesure

$$F\text{-mesure} = \frac{2}{\frac{1}{précision} + \frac{1}{rappel}} = 2 \cdot \frac{précision \cdot rappel}{précision + rappel} = \frac{2 \cdot TP}{2 \cdot TP + FN + FP}$$

5. Accuracy (justesse)

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

6. Intersection over Union (IoU)

SegIA = segment identifié par l'algorithme de reconnaissance des formes ; SegVT = segment correspondant dans la vérité terrain.

$$IoU = \frac{SegIA \cap SegVT}{SegIA \cup SegVT}$$

Exercice : calculer le nombre de données traitées, la précision, le rappel, la F-mesure et l'accuracy pour :

TP	12	12	12
FP	36	12 (=36-24)	12
TN	8	8	48
FN	4	28 (=4+24)	28
Nb données			
Precision			
Rappel			
Moyenne			
F-mesure			
Accuracy			

7. Baseline

Il s'agit d'une méthode opérationnelle bien identifiée de reconnaissance des formes qui sert de référence pour la comparaison avec une méthode étudiée. Une baseline peut être :

- Une méthode naïve
- Une méthode de l'état de l'art
- La même méthode que celle qui est étudiée, mais avant amélioration
- Un « oracle » qui peut être :
 - Un générateur (par exemple aléatoire) de prédictions
 - Un humain (notamment pour la reconnaissance de concepts subjectifs)

II. Extraction d'informations caractéristiques

Caractéristiques = features mais pas embeddings

A. Caractérisation de régions

1. Couleur

a) *Histogrammes*

Cf. cours 2A SRI

b) *Espaces de couleur*

Cf. cours 2A SRI

2. Texture

a) *Transformée de Fourier – Filtres de Gabor*

→ Transformée de Fourier Discrète directe et inverse,

Dans le domaine discret, le calcul de la transformée de Fourier s'exprime par :

$$\hat{F}_\omega = \sum_{t=0}^{N-1} F_t e^{-2i\pi \frac{\omega t}{N}}$$

Le calcul de la transformée inverse est donné par :

$$F_t = \frac{1}{N} \sum_{\omega=0}^{N-1} \hat{F}_\omega e^{2i\pi \frac{\omega t}{N}}$$

Exemple :

X	1	3	8	4	5	1	2	6
TFD(x)	30	-1.1716 - 6i	-4 + 6i	-6.8284 + 6i	2	-6.8284 - 6i	-4 - 6i	-1.1716 + i
TFD(x)	30	6.1133	7.2111	9.09	2	9.09	7.2111	6.1133

Remarquons que :

- le premier coefficient de la TFD est réel. Il est égale à la somme des échantillon (en pratique c'est N.moyenne(x)). On appelle généralement ce coefficient dans les transformée fréquentielle, le coefficient « DC » ou « composante continue ».

Exercice : Remplir le tableau ci-dessous :

X	1	5	1	5	1	5	1	5
TFD(x)								
TFD(x)								

Transformée de Fourier 2D

La transformée de Fourier 2D d'une image est obtenue en appliquant la TFD sur chaque ligne indépendamment les unes des autres. Puis on applique la TFD sur chaque colonne du résultat obtenu.

→ Notions de Systèmes et de Filtres

Une fois qu'on a calculé le spectre, il est possible de produire un filtrage fréquentiel en annulant ou en atténuant les amplitudes de certaines des fréquences. On distingue les filtres :

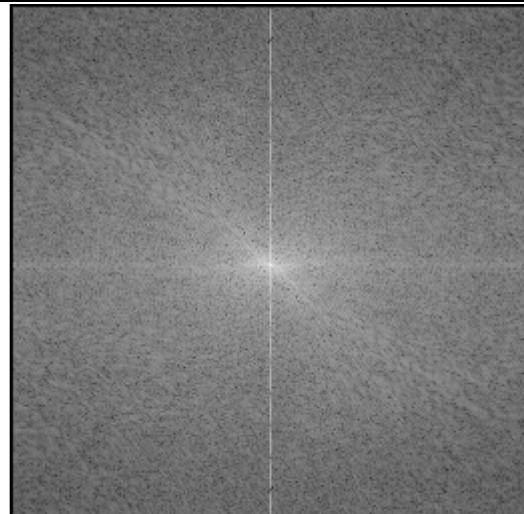
- passe-haut / coupe-bas : qui ne conserve que les hautes fréquences
- passe-bas / coupe-haut
- passe-bande / coupe bande : qui ne conserve (resp. coupe) que les informations d'un intervalle fréquentiel donné.

→ Représentation fréquentielle d'image

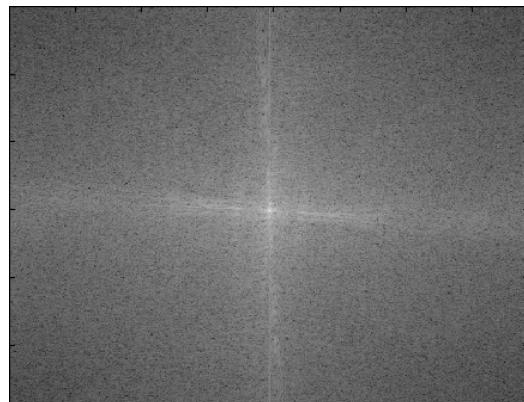
Il est souvent d'usage en traitement d'image d'effectuer une permutation des quadrants pour ramener les basses fréquences (BF) au centre) afin de simplifier les calculs de filtrage notamment.

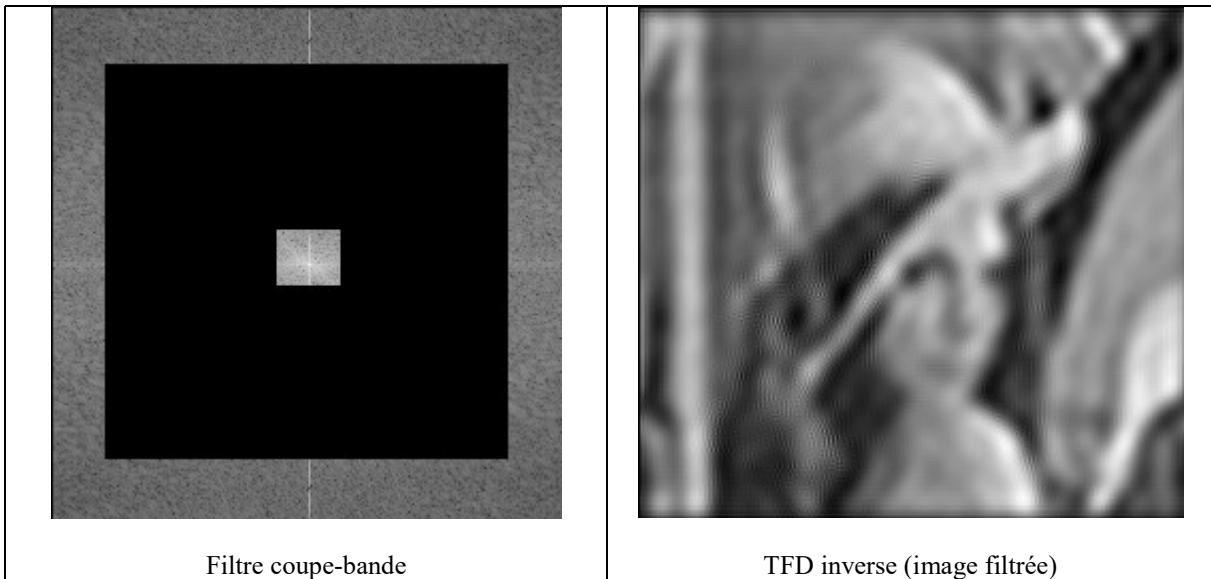


Image de départ (Lena)



TFD 2D
(log du module normalisé - BF au centre)





➔ Filtres de Gabor

Les paramètres du filtre sont $\mu_\theta, \sigma_\theta, \mu_\omega, \sigma_\omega$

$$\text{Un filtre s'exprime sous la forme : } G(\theta, \omega) = e^{\frac{(\theta - \mu_\theta)^2}{2\sigma_\theta^2}} \cdot e^{\frac{(\omega - \mu_\omega)^2}{2\sigma_\omega^2}}$$

Exemple de valeurs possibles pour les paramètres des filtres :

pour i allant de 0 à 4 (= 5 « grains » possibles)

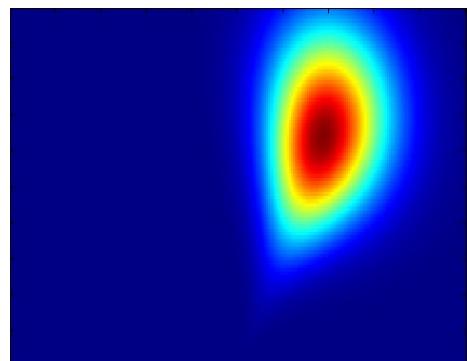
pour j allant de 0 à 5 (= 6 orientations possibles)

$$\mu_\omega(i) = 3/4 * (\max(F) - \min(F)) \cdot 2^{-i}$$

$$\mu_\theta(j) = j * \pi / 6$$

$$\sigma_\omega(i) = (\max(F) - \min(F)) \cdot 2^{-(i+1)} / \sqrt{8 \ln(2)}$$

$$\sigma_\theta(j) = (\pi / 12) / \sqrt{2 \ln 2}$$



b) *Transformée en Ondelette – Filtres de Haar*

➔ Transformée en ondelettes continues

$$W(a, b) = \langle f, \psi_{a,b} \rangle = \int_{t=-\infty}^{+\infty} f(t) \overline{\psi}_{a,b}(t) dt \text{ avec } \psi_{a,b}(t) = \frac{1}{\sqrt{|a|}} \psi\left(\frac{t-b}{a}\right).$$

Où $\psi(t)$ est une **ondelette mère** à définir. Le paramètre « a » exprime la « dilatation » (ou l'échelle) de l'ondelette mère. Plus l'échelle (temporelle) est petite » plus la fréquence analysée est élevée : on utilise une petite onde qui oscille donc plus. A l'inverse, une grande échelle permettra d'analyser des basses fréquences (l'onde oscille plus lentement).

« b » exprime sa translation, c'est-à-dire la position sur le signal où on calcule la décomposition en ondelettes.

Pour qu'une fonction puisse jouer le rôle d'ondelette mère, il faut qu'elle soit continue, à valeur complexe (ou réelle), et qu'elle vérifie les conditions suivantes :

- $\int_{t=-\infty}^{+\infty} \psi(t) dt = 0$ (ie, dans le domaine discret, si elle est réelle : $\sum \text{coeff} > 0 = -\sum \text{coeff} < 0$)
- $\int_{t=-\infty}^{+\infty} |\psi(t)|^2 dt < \infty$

La transformation inverse est obtenue avec :

$$f(t) = \frac{1}{c_\psi^2} \int_a \int_b W(a, b) \frac{1}{a^2} \psi\left(\frac{t-b}{a}\right) \partial b \partial a \quad \text{avec} \quad c_\psi = \sqrt{2\pi \int_{\omega=-\infty}^{+\infty} \frac{|\hat{\psi}(\omega)|^2}{|\omega|} d\omega}$$

Cette reconstruction n'est possible que si $c_\psi < +\infty$. Cette condition est appelée la **condition d'admissibilité**.

→ Transformée en Ondelette discrète - Algorithme de Mallat

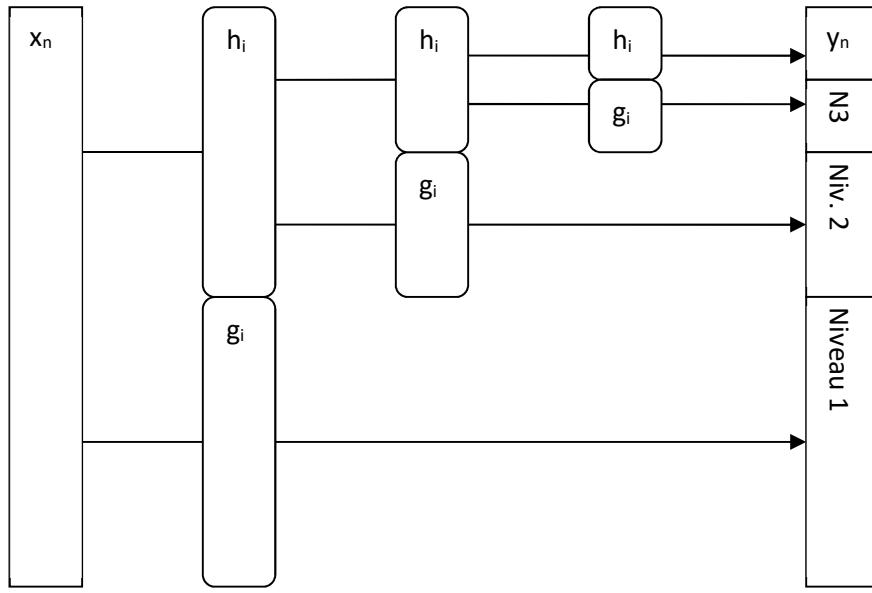
Le premier problème posé consiste à discréteriser l'espace des paramètres a et b , tout en conservant une expression continue du signal. Pour garantir la reconstruction du signal par transformée inverse, on utilise une « discréétisation logarithmique » du facteur d'échelle - la base du logarithme étant laissée au choix de l'utilisateur. Généralement, on prend un logarithme binaire, ce qui conduit à prendre pour a les valeurs 1, 2, 4, 8, 16, ... On parle alors de **décomposition dyadique** du signal.

On pose $a = a_0^j$ où a_0 dépend de la base logarithmique choisie (ie. 2) et où j exprime le niveau de la décomposition. On pose $b = k \cdot a_0^j \cdot b_0$ où k exprime le multiple du décalage de la fenêtre, et où b_0 est relatif au pas de progression de cette fenêtre. En général, on choisit $b_0 = 1$.

On peut exprimer alors la **fonction d'échelle** qui représente l'ondelette à différentes valeurs

$$\text{d'échelle et pour différentes valeurs de translation : } \psi_{j,k}(t) = a_0^{-j} \cdot \psi(a_0^{-j} t - kb_0).$$

La Transformée en Ondelettes Discrète s'exprime par un algorithme décomposant le signal en différentes sous-bandes fréquentielles. Ces bandes sont obtenues par la convolution de ce signal avec un filtre linéaire passe-bas et un filtre linéaire passe haut (complémentaire) à différentes échelles. Ces filtres font office d'ondelettes dans cette décomposition.



Les réponses impulsionnelles de h et g sont dépendantes l'une de l'autre (le filtrage doit être complémentaire). La relation qui relie h et g peut être :

$$g_{l-1-n} = (-1)^n \cdot h_n.$$

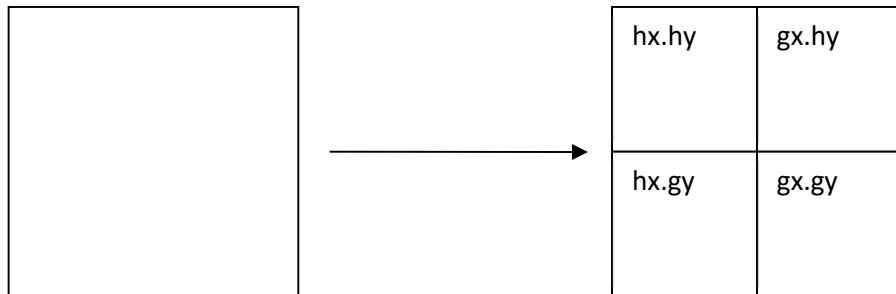
Les filtres qui vérifient cette relation sont appelés des **filtres miroirs en quadrature**.

Le nombre de niveaux (ie. le nombre d'itération du processus sur les basses-fréquences – ou encore l' « ordre ») peut être fixé a priori, ou dépendre d'un seuil sur le nombre de valeurs produites après sous-échantillonnage.

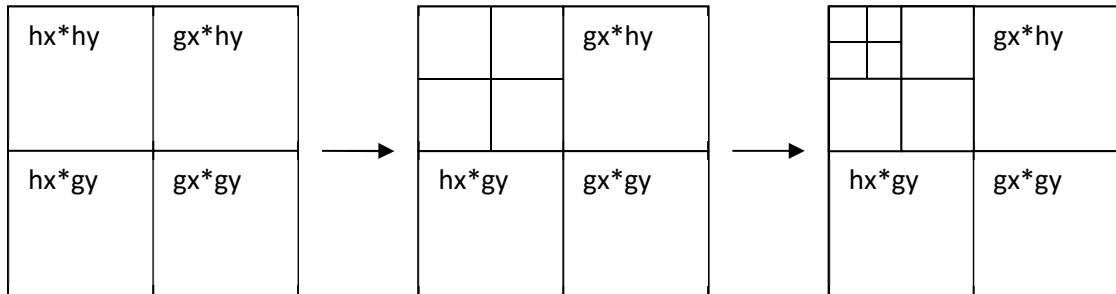
A chaque niveau de la décomposition correspond une résolution du signal (basse fréquence) qui est sous-échantillonné. On dit qu'on effectue une **analyse multirésolution**.

➔ Transformée en ondelettes 2D

On applique à l'image une convolution ligne par ligne avec les filtres gx et hx , puis sur les résultats de cette première convolution, une seconde convolution colonne par colonne avec les filtres gy et hy . Les résultats sont rangés dans une matrice ayant les mêmes dimensions que l'image de la manière suivante :



Le processus est répété sur l'image des basses fréquences obtenue dans la partie (hx.hy)



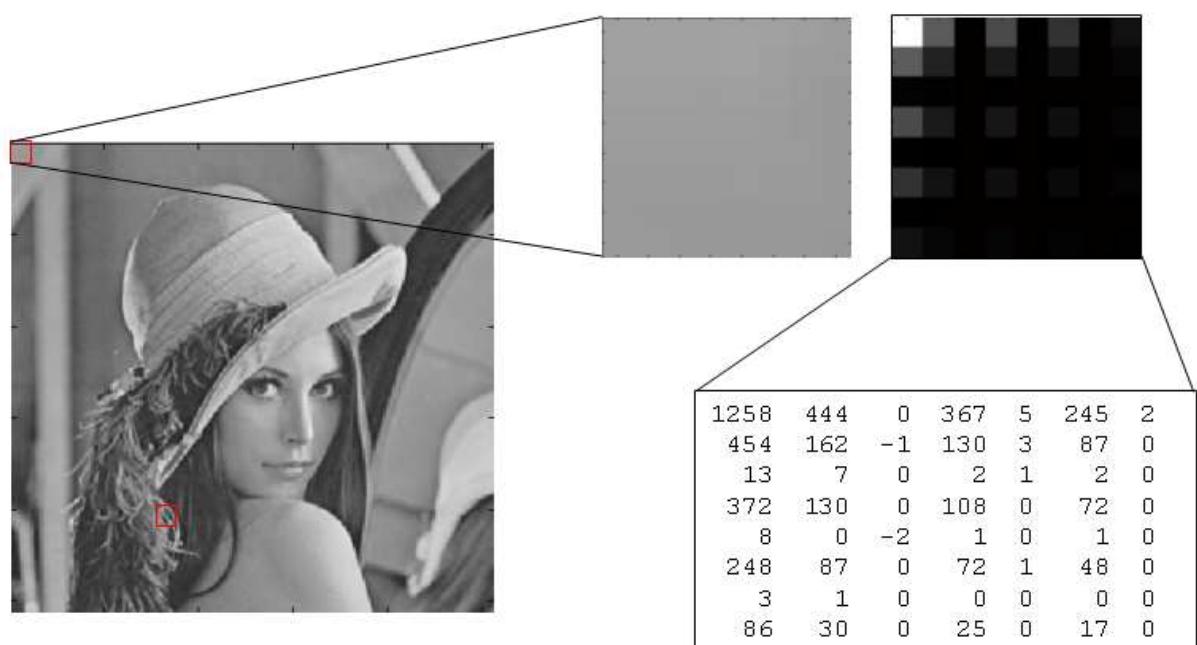
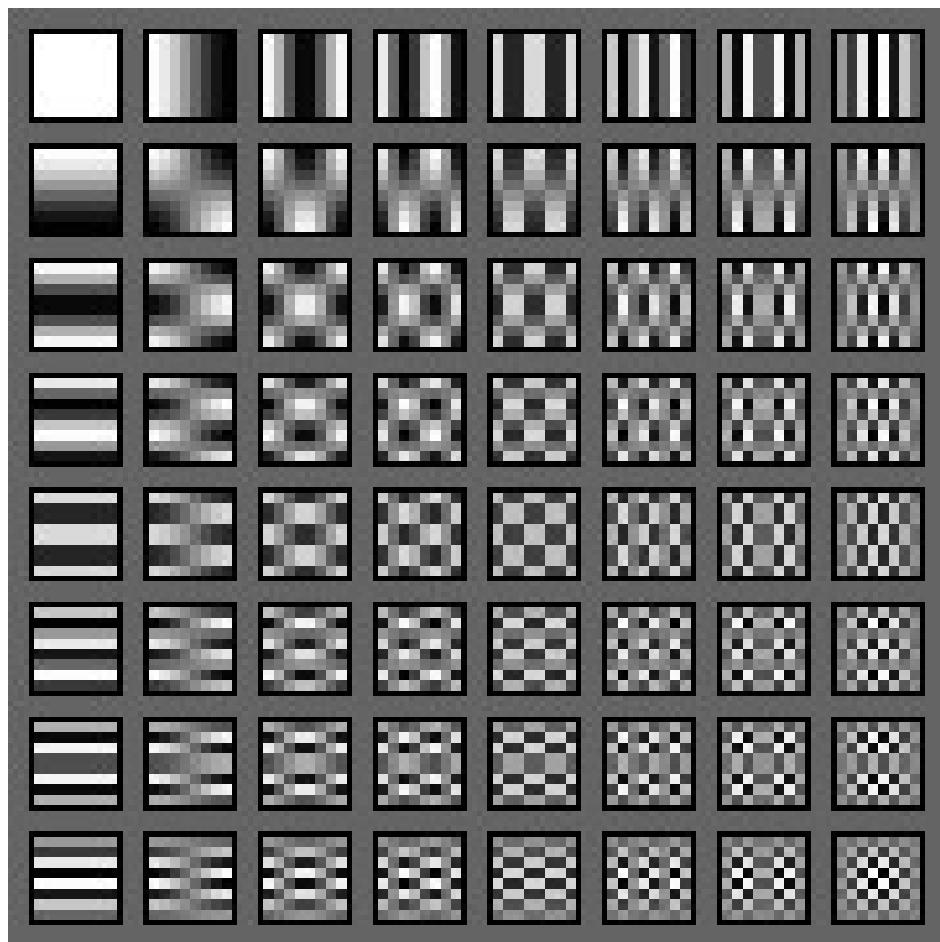
c) Transformée en Cosinus Discrète

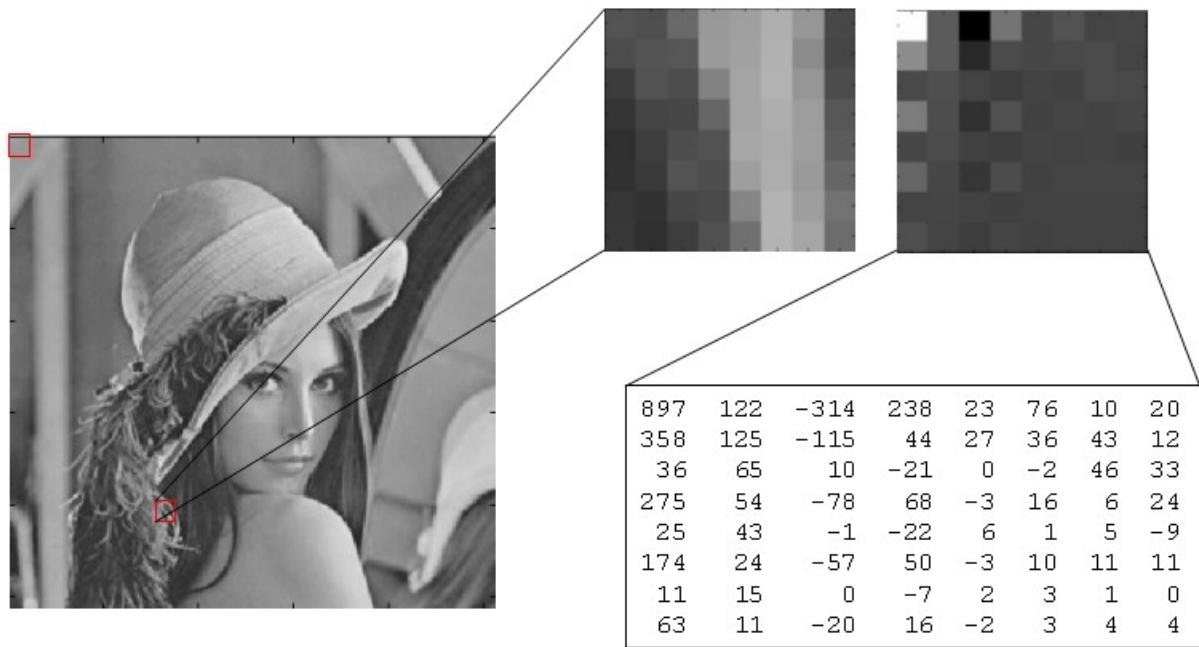
$$DCT(u,v) = \frac{2}{N} c(u).c(v) \sum_{y=0}^{N-1} \sum_{x=0}^{N-1} \cos\left(\frac{\pi}{N} u \left(x + \frac{1}{2}\right)\right) \cos\left(\frac{\pi}{N} v \left(y + \frac{1}{2}\right)\right) M(x,y)$$

$$\text{avec } c(a) = \begin{cases} \frac{1}{\sqrt{2}} & \text{si } a = 0 \\ 1 & \text{sinon} \end{cases}$$

La transformée inverse est obtenue avec :

$$M(x,y) = \frac{2}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} c(u).c(v).DCT(u,v) \cos\left(\frac{\pi}{N} u \left(x + \frac{1}{2}\right)\right) \cos\left(\frac{\pi}{N} v \left(y + \frac{1}{2}\right)\right)$$





d) Matrices de cooccurrence

La matrice de cooccurrence NxN ($N = \text{nb de niveaux de gris}$) est définie par :

$$m_R(i,j) = \frac{\#\{(x,y)(x',y')|(x,y)R(x',y') \wedge I(x,y)=i, I(x',y')=j\}}{\#\{(x,y)(x',y')|(x,y)R(x',y')\}}$$

R : relation spatiale entre 2 pixels (distance et orientation)

$I(x,y)$: niveau de gris à la position (x,y)

: nombre d'éléments

Les paramètres d'Haralick sont souvent utilisés pour caractériser la distribution des coefficients dans la matrice. Ils sont calculés à l'aide des 14 fonctions suivantes :

notations

$$m_x(i) : \text{somme de la ligne } i \quad m_{x+y}(k) : \text{somme de la } k^{\text{ième}} \text{ diagonale secondaire}$$

$$m_y(i) : \text{somme de la colonne } i \quad m_{x-y}(k) : \text{somme de la } k^{\text{ième}} \text{ diagonale principale}$$

moment angulaire d'ordre 2 $f_1 = \sum_i \sum_j m(i,j)^2$

contraste $f_2 = \sum_{n=0}^N n^2 \cdot m_{x-y}(n)$

corrélation $f_3 = \frac{\sum_i \sum_j (i \cdot j \cdot m(i,j) - \mu_x \mu_y)}{\sigma_x \sigma_y}$

variance $f_4 = \sum_i \sum_j (i - j)^2 \cdot m(i,j)$

moment des différences inverses $f_5 = \sum_i \sum_j \frac{1}{1 + (i - j)^2} \cdot m(i,j)$

moyenne des sommes $f_6 = \sum_{k=2}^{2N-1} k \cdot m_{x+y}(k)$

variance des sommes $f_7 = \sum_{k=2}^{2N-1} (k - f_6)^2 \cdot m_{x+y}(k)$

entropie de la somme $f_8 = - \sum_{k=2}^{2N-1} m_{x+y}(k) \cdot \log(m_{x+y}(k))$

entropie $f_9 = - \sum_i \sum_j m(i,j) \cdot \log(m(i,j))$

variance des différences $f_{10} = \sum_{k=0}^N (k - \mu_{x-y})^2 \cdot m_{x-y}(k)$

avec $\mu_{x-y} = \frac{1}{N} \sum_{k=0}^N m_{x-y}(k)$

entropie des différences : $f_{11} = -\sum_{i=0}^N m_{x-y}(i) \log(m_{x-y}(i))$

corrélation de l'information : $f_{12} = \frac{H_{XY} - H_{XY}}{\max(H_X, H_Y)}$

$$f_{13} = [1 - \exp(-2.0(H_{2XY} - H_{XY}))]^{\frac{1}{2}}$$

avec $H_{XY} = f_9$ $H_X = -\sum_{k=0}^N m_x(k) \log(m_x(k))$ $H_Y = -\sum_{k=0}^N m_y(k) \log(m_y(k))$

$$H_{1XY} = -\sum_i \sum_j m(i,j) \log(m_x(i)m_y(j)) \quad H_{2XY} = -\sum_i \sum_j m_x(i)m_y(j) \log(m_x(i)m_y(j))$$

corrélation maximum : $f_{14} = (2\text{ème des plus grandes valeurs propres de } Q)^{\frac{1}{2}}$

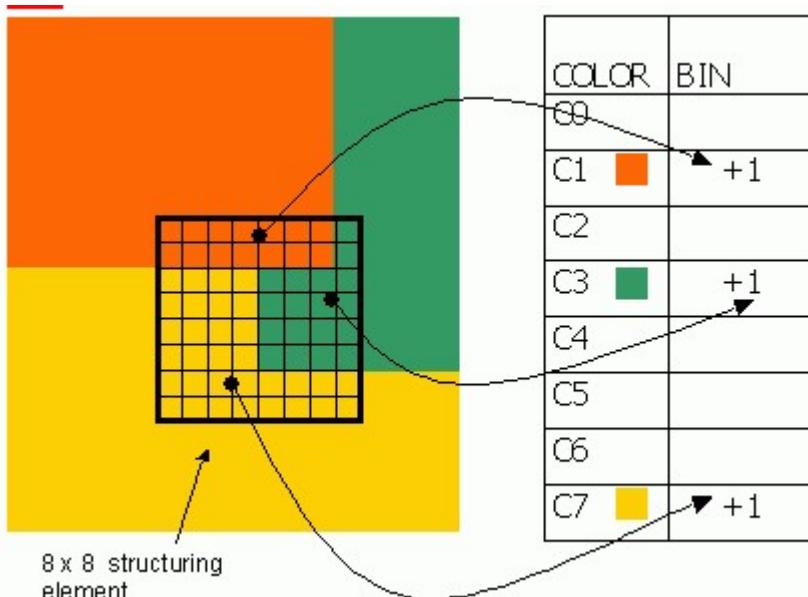
$$\mathcal{Q} = \{\mathcal{Q}(i,j)\}_{\substack{i=0 \dots N \\ j=0 \dots N}} \quad \mathcal{Q}(i,j) = \sum_{k=0}^N \frac{m(i,k)m(k,j)}{m_x(i)m_y(j)}$$

e) Structure de couleur

On crée un tableau comprenant autant d'entrée qu'il y a de couleurs dans la LUT.

Un élément structurant (ici un carré 8x8) balaye la région.

Pour chaque couleur différente C présente dans la région, $T[C] \leftarrow T[C] + 1$

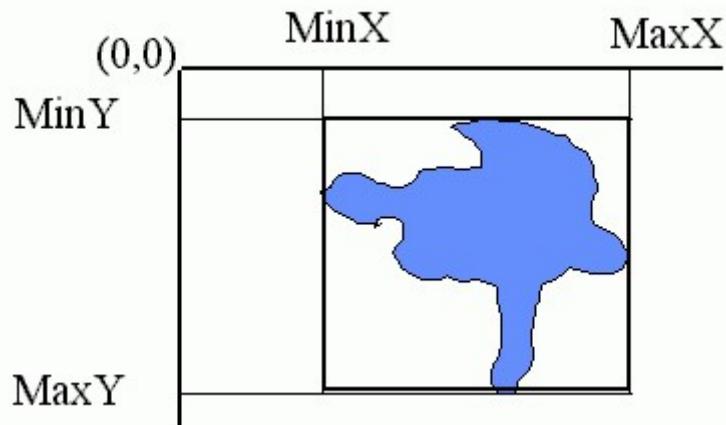


Lorsqu'une couleur est "compacte", elle est associée à une petite valeur dans le tableau par rapport au nombre de pixels concernés. Lorsqu'une couleur est éparsillée, elle présente une forte valeur dans le tableau par rapport au nombre de pixels concernés.

3. Forme

a) Caractérisation de la région complète

→ les coordonnées de la boîte englobante



→ Centre de gravité - centre géodésique

Soit $d_R(a,b)$ la distance géodésique dans la région R , c'est à dire la plus petite distance à l'intérieur de X entre le pixel a et le pixel b . On peut définir cette distance comme étant le chemin comportant le moins de pixels en passant de pixel à pixel par un des 4 voisins tout en restant à l'intérieur de R .

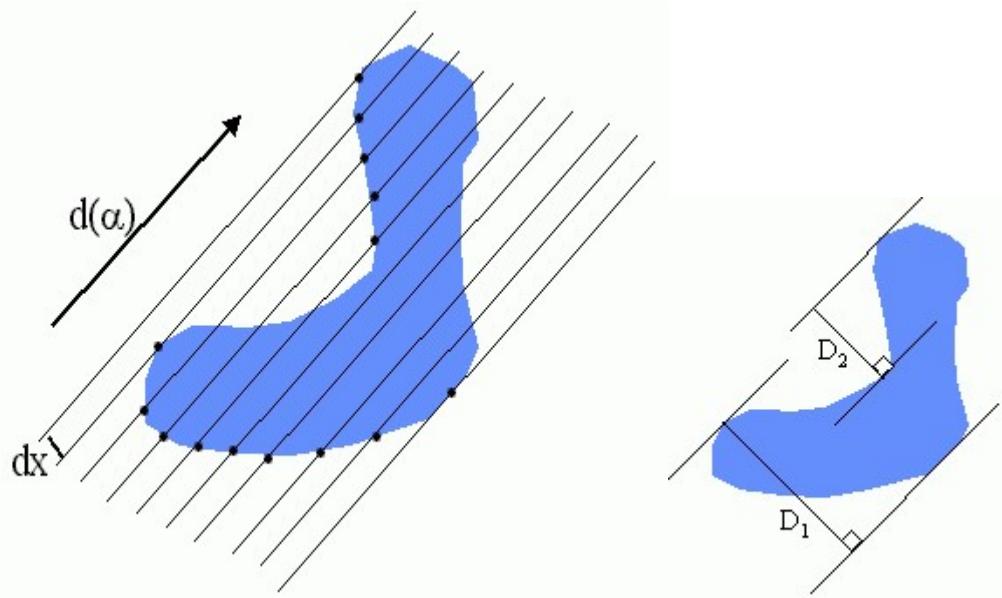
On définit la fonction de propagation P_R par : $P_R(a) = \max(d(a,b))$, pour tout b appartenant à R .

Le centre géodésique est alors le point produisant la valeur minimale de P_R .

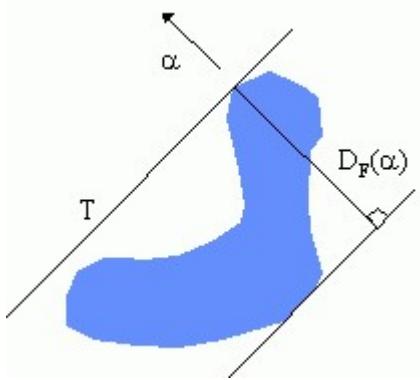
- l'aire
- le périmètre
- diamètre - diamètre géodésique - rayon géodésique

b) Caractérisation du contour

→ Le nombre d'intercepts (ou calcul de la variation diamétrique)



- ➔ Périmètre de Crofton
- ➔ Diamètre de Feret



- ➔ Facteur de compacité
- $FC = \text{Périmètre}(R)/\sqrt{4\pi \cdot \text{Aire}(R)}$

c) [VII.4 Moments](#)

On définit le moment cartésien d'ordre $p+q$ par :

$$m_{pq} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x^p y^q g(x, y) dx dy$$

$$m_{pq} = \sum_{y=0}^{NL-1} \sum_{x=0}^{NC-1} x^p y^q f(x, y)$$

$g(x,y)$ et $f(x,y)$ sont les fonctions images dans le cas continu et le cas discret. $x^p y^q$ est la "base". On définit l'ensemble complet des moments d'ordre n par : $\{m_{pq} / p+q \leq n\}$.

On montre que $\{m_{pq}\}$ est défini de manière unique par $f(x,y)$. L'utilisation des moments consiste donc en général à définir un sous-ensemble de $\{m_{pq}\}$ suffisant pour caractériser une région dans le cadre d'une application donnée.

Pour une image binaire, $f(x,y) = 1$ si le point appartient à la région et vaut 0 sinon.

On définit à partir des coordonnées des centres de gravité les moments centraux:

$$\mu_{pq} = \sum_{y=0}^{N_L-1} \sum_{x=0}^{N_C-1} (x - \bar{x})^p (y - \bar{y})^q \cdot f(x,y)$$

→ Les moments invariants

Ces moments sont invariants en translation, rotation et homothétie (Hu).

$$M(1) = \mu_{20} + \mu_{02}$$

$$M(2) = (\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2$$

$$M(3) = (\mu_{30} - 3\mu_{12})^2 + (3\mu_{21} - \mu_{03})^2$$

$$M(4) = (\mu_{30} + \mu_{12})^2 + (\mu_{21} + \mu_{03})^2$$

$$M(5) = (\mu_{30} - 3\mu_{12})(\mu_{30} + \mu_{12}) \left[(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2 \right] + (3\mu_{21} - \mu_{03})(\mu_{21} + \mu_{03}) \left[(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2 \right]$$

$$M(6) = (\mu_{20} - \mu_{02}) \left[(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2 \right] + 4\mu_{11}(\mu_{30} + \mu_{12})(\mu_{21} + \mu_{03})$$

$$M(7) = (3\mu_{21} - \mu_{03})(\mu_{30} + \mu_{12}) \left[(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2 \right] + (\mu_{30} - 3\mu_{12})(\mu_{21} + \mu_{03}) \left[(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2 \right]$$

Moments invariants en échelle :

$$\eta_{ij} = \frac{\mu_{ij}}{\mu_{00}^{\left(1+\frac{i+j}{2}\right)}}$$

→ Bitquads

Histogramme de motifs binaires 2x2 comportant au moins un '0' et au moins un '1'.

→ Descripteurs de Fourier

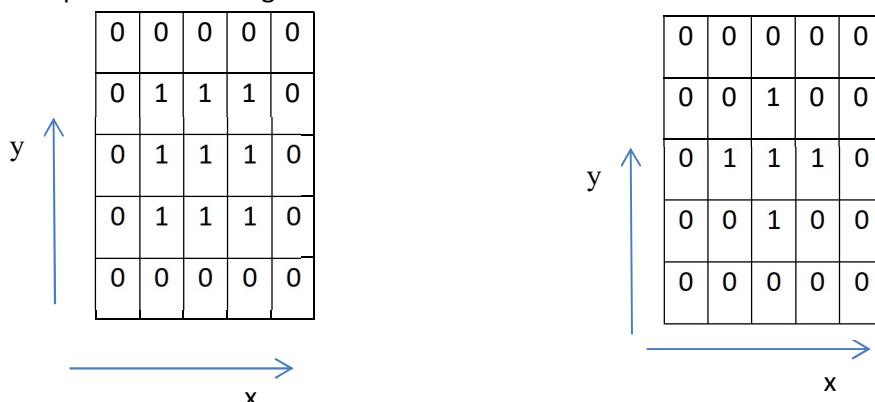
Coefficients fréquentiels issus de la transformée de Fourier discrète appliquée à la fonction de courbure k d'une forme. Cette fonction est définie comme étant la dérivée de la fonction angulaire qui donne en tout point selon une abscisse curviligne l'angle du contour de la forme avec l'axe des abscisses.

→ Curvature scale space

Tableau décrivant le nombre d'opérations de lissage à effectuer sur le contour d'une forme pour faire disparaître toutes les concavités.

Exercice : On considère les régions correspondant aux ‘1’ dans les images binaires ci-dessous :

Donnez pour ces deux régions :



1. Le périmètre
2. L’aire
3. Les coordonnées de leur centre de gravité
4. Les coordonnées de leur centre de gravité géodésique
5. Le nombre d’intercepts pour $t=\{0^\circ, 45^\circ, 90^\circ, 135^\circ\}$
6. Le diamètre de Feret pour $t=\{0^\circ, 45^\circ, 90^\circ, 135^\circ\}$
7. Le facteur de compacité
8. Les moments centraux $\mu_{10}, \mu_{11}, \mu_{01}$
9. Les bitquads
10. Le Curvature Scale Space

B. Points d’intérêt

1. Principe général

Objectif : mettre en correspondance 2 images ou caractériser un objet pour l’identifier dans toute image où il pourrait figurer.

Principe :

- Détection de zones de saillance dans les images : des points spécifiques qu’on peut retrouver même si l’image est passablement modifiée
- Association d’un descripteur à chaque point
- Appariement des points entre 2 images en fonction de la position des points et/ou de leur description

2. Détection des points d'intérêt

a) DéTECTEUR de Moravec (1980)

Les points de Moravec sont les maximums locaux de la fonction :

$$SM(x, y) = \min_{(a,b)} \sum_{u,v} (\omega(u, v) \cdot |I(x+u, y+v) - I(x+u+a, y+v+b)|)$$

Exemple de voisinage exprimé par ω pour un 4-V

u\w	-1	0	1
-1	0	1	0
0	1	1	1
1	0	1	0

Exercice : Appliquez cette méthode sur la matrice ci-dessous pour $(a, b) \in [-1,1]^2$ et un voisinage 8-V.

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

b) DéTECTEUR de Harris (1988)

La fonction calculée s'exprime par :

$$SH(x, y) = A.*B - C.^2 - k(A + B).^2$$

Avec :

- $A = m\left(\left(\frac{\delta I}{\delta x}\right)^2\right)$, $B = m\left(\left(\frac{\delta I}{\delta y}\right)^2\right)$, $C = m\left(\left(\frac{\delta I}{\delta x}\right)\left(\frac{\delta I}{\delta y}\right)\right)$,
- k à fixer expérimentalement
- $m(a)$ est la moyenne pondérée des valeurs dans le voisinage autour du point considéré

Exercice : Même question que précédemment sur la matrice ci-dessous – avec calcul de la moyenne non pondérée sur un support 4V. Déterminer les conditions sur k pour obtenir une « bon comportement de SH sur cet exemple.

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	0	0	0	0	0	0	0
0	0	0	0	5	5	5	5	5	5
0	0	0	0	5	5	5	5	5	5
0	0	0	0	5	5	5	5	5	5
0	0	0	0	5	5	5	5	5	5
0	0	0	0	5	5	5	5	5	5
0	0	0	0	5	5	5	5	5	5

c) SUSAN (1997)

Le détecteur SUSAN considère que les coins sont des points dont les voisins ont peu de valeurs similaires. On définit une zone circulaire autour d'un point (le noyau), à l'aide d'un structurant e 37 pixels.

Puis on considère les pixels de cette zone qui ont un niveau de gris proche de ce noyau. Ces pixels forment une zone appelée USAN ("Univalue Segment Assimilating Nucleus"). En chaque point du disque, on calcule :

$$c((x_0, y_0), (x, y)) = e^{-\left(\frac{I(x_0, y_0) - I(x, y)}{t}\right)^6}$$

où (x_0, y_0) est le nucléus, (x, y) est le point testé et t est le seuil sous lequel on considère que deux pixels ont une couleur proche. Puis on calcule :

$$n(x_0, y_0) = \sum_{(x, y)} c((x_0, y_0), (x, y)).$$

$$R(x_0, y_0) = \begin{cases} g - n(x_0, y_0) & si \quad n(x_0, y_0) < g \\ 0 & sinon \end{cases}$$

Les maximums locaux de R sont des coins ou des contours.

Pour ne retenir que des coins, il faut appliquer 2 filtrages supplémentaires. Pour chaque point on calcule le « centre de gravité des USANs ». On élimine alors :

- les points dont les centres de gravité sont trop proches
- les points dont le segment reliant au centre de gravité et jusqu'au centre n'appartiennent pas à l'USAN.

Exercice:

Question 1 : Donner une valeur (approximée) de la fonction R dans la zone grisée des deux matrices ci-dessous :

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

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

Question 2 : Identifier le ou les points d'intérêt sur les résultats obtenus à la question 1 sur chaque matrice.

d) SIFT (2004)

(Scale Invariant Feature Transform)

Ces descripteurs sont produits en 2 phases :

- l'extraction de points clés
- le filtrage des points candidats

e) SURF

(Speeded Up Robust Feature)

Principe identique aux SIFT avec accélération des calculs.

f) Squelette

3. Descripteurs locaux

a) IV.1 SIFT

Chaque niveau produit un certain nombre de points clés. Pour chaque point clé, on produit un descripteur sous la forme de 16 histogrammes de 8 bins de la manière suivante :

- on prend une fenêtre de 16 par 16 pixels, centrée sur le point d'intérêt
- on coupe cette fenêtre en 16 sous-fenêtres disjointes de 4 sur 4 pixels.
- Dans chaque sous-fenêtre, en chaque point on calcule l'amplitude et la direction du gradient.
- Pour chaque point d'intérêt, on produit un descripteur de $4 \times 4 \times 8 = 128$ valeurs. Ce vecteur est normalisé par sa norme maximale théorique.

Pour la comparaison des SIFT, on utilise 2 métriques : distance en cosinus et la distance angulaire :

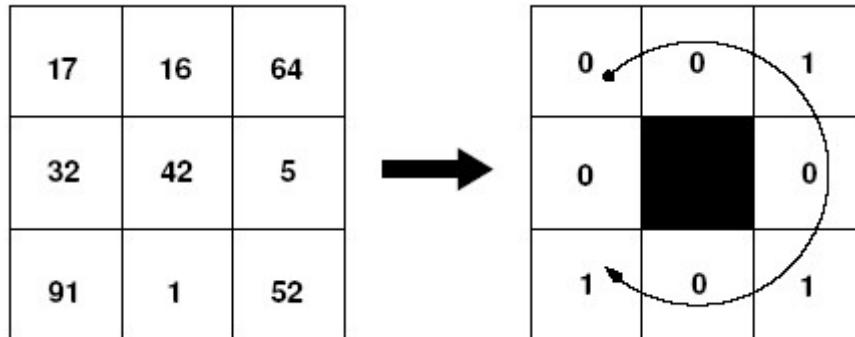
- $\text{Cos}(V_o, V_r) = V_o \cdot V_r / (|V_o| \cdot |V_r|)$
- $\text{Dangle} = \cos^{-1}(V_o \cdot V_r / (|V_o| \cdot |V_r|))$

b) SURF

A peu près identique aux SIFT avec :

- La recherche d'une orientation « invariante » à la rotation (estimation de la direction du plus fort gradient à l'échelle retenue)
- La production d'un descripteur de 4 valeurs pour chaque 4x4 sous-regions d'une fenêtre centree sur le point dont la taille depend de l'echelle soit 64 valeurs. Les 4 valeurs sont : la somme des dérivées locales en x et en y ; la somme des valeurs absolues des dérivées locales en x et en y.

Pour identifier si une image I correspond à une image M indexée, on « apparie » chaque descripteur d_I de la première à un descripteur de la seconde. On cherche le descripteur d_M le plus similaire. Si la distance euclidienne est inférieure à un seuil, on établit la correspondance.



00101010

Un descripteur LBP est invariant par une variation monotone de la valeur des pixels, ce qui est intéressant pour résister aux changements d'illumination.

d) Descripteurs locaux issus de squelettes

1. On assimile le squelette d'une forme à un graphe. Les nœuds et les feuilles sont pris comme étant des points d'intérêt.
2. Les descripteurs couramment associés à ces points sont :
 - a. Le nombre d'érosions nécessaires pour obtenir le nœud (ie épaisseur locale de la forme ou encore rayon de la boule maximale centrée sur le nœud)
 - b. Degré du nœud
 - c. Coordonnées polaires des points dans un repère centré sur la forme dont l'axe des abscisses est l'orientation principale de la forme (invariance à l'orientation). La coordonnée radiale peut être normalisée par rapport à la longueur de l'objet projeté sur cet axe principal. (invariance à l'échelle)

III. Adaptation des données

A. Normalisation

Les caractéristiques présentant un grand écart type ou qui s'expriment sur un grand intervalle de valeur par rapport aux autres ont de fort risque d'être privilégiées par les systèmes de décision, même si elles sont moins discriminantes. Pour éviter ce problème, il peut être utile de normaliser les données en entrée du système de reconnaissance des formes.

a) Normalisation de l'amplitude de variation des Features

$$F_{norm}(i) = \frac{F(i) - (\min_{j=1}^N F(j))}{(\max_{j=1}^N F(j)) - (\min_{j=1}^N F(j))}$$

b) *Normalisation « standard »*

$$F_{norm}(i) = \frac{F(i) - F_{moyenne}}{F_{écart-typ}}$$

B. Réduction des dimensions

Plus le nombre de caractéristiques à traiter par exemple est grand, plus le temps de calcul du modèle – et potentiellement de la décision – est important. Dans certains cas, la dimension de l'espace des caractéristiques peut rendre un système de reconnaissance inapplicable. Pour limiter ce problème, toujours en amont du système de reconnaissance, il peut être utile d'appliquer une des méthodes de réduction de dimension suivantes :

1. Analyse en Composantes Principales

Exemple de l'utilisation de l'ACP pour l'identification de personnes par le visage : les « eigenfaces ».

Algo de l'ACP appliqué à la reconnaissance des formes

Exercice :

Soient 4 images de visage fortement quantifiées ... sur 3 pixels. $V1=[1\ 1\ 4]$; $V2=[1\ 4\ 1]$; $V3=[4\ 1\ 1]$; $V4=[3\ 3\ 3]$. On apprend un modèle du visage à l'aide des vecteurs $V1$, $V2$ et $V3$. On teste si le visage $V4$ correspond bien à la même personne. Pour cela, on reconstruit le visage à l'aide du premier vecteur propre. On estime que le visage est reconnu si l'erreur quadratique de reconstruction est strictement inférieure à 1. $V4$ représente-t-il le visage de la personne à reconnaître ?
(Rq : les valeurs données permettent de simplifier les calculs.)

2. Décomposition en Valeurs Singulières

Exercice : Soient 3 exemples de visage $V1=[1\ -1]$, $V2=[0\ 0]$, $V3=[-1\ 1]$ et un visage dont on veut déterminer l'identité $V4=[1\ 1]$.

Déterminer U , S et V . On reconstruit le visage $V4$ à l'aide uniquement de la première valeur singulière. Si l'erreur quadratique moyenne est inférieure à 1, on identifie qu'il s'agit de la même personne. Est-ce le cas ?

3. Feature selection

Principe général : plus on dispose de caractéristiques en entrée, et le coût de calcul est élevé.
L'objectif est d'établir le meilleur compromis entre accuracy et coût de calcul. Soit N , le nombre de caractéristiques en entrée d'un classifieur. On établit une métrique sous la forme d'un rapport entre accuracy et coût de calcul de ce classifieur. Puis on teste toutes (ou un sous-ensemble de toutes) les combinaisons possibles de caractéristiques avec cette métrique pour identifier un meilleur choix.
Pistes : algorithme glouton, test de l'élimination d'abord des caractéristiques les plus coûteuses à produire, élimination incrémentale (élimination de la caractéristique à plus faible impact sur l'accuracy, puis de la seconde, etc ou, à l'inverse, sélection de la feature à plus fort impact, puis ajout de la seconde, etc.)

4. Autres possibilités

Il existe d'autres méthodes de réduction de dimension plus rarement utilisées telles que l'analyse discriminante factorielle, l'analyse discriminante linéaire, etc.

IV. Méthodes de reconnaissance

A. Arbres de décision

On désigne la base d'apprentissage par $\{(x_i, y_i)\}$. On désigne par f la fonction qui extrait de l'exemple x un vecteur de description. $f(x, k)$ désigne la k ème valeur du vecteur de description de x . On crée un arbre où à chaque nœud correspond un sous ensemble de la base d'apprentissage. La racine de l'arbre est associé à toute la base – les feuilles sont associées à des sous parties telles que les y_i ont autant que faire se peut la même valeur.

B. Random Forest

1. Apprentissage

Algorithme

- Construire N arbres (classificateurs faibles) :
 - Créer la racine de l'arbre associée à la base d'apprentissage $\{(x_i, y_i)\}$
 - Tant que le critère d'arrêt⁽¹⁾ sur les feuilles de l'arbre n'est pas vérifié
 - Sélectionner une feuille F qui pénalise le critère d'arrêt et dont l'ensemble des données associé peut être partitionné
 - Tirer aléatoirement une valeur de $k^{(2)}$
 - Déterminer $N^{(3)}$ seuils qui minimisent l'erreur de partition des y_i pour tous les $f(x_i, k)$
 - Construire $N+1$ sommets reliés chacun à F . Associer à ces sommets les partitions de l'ensemble de F . Associer à chaque arrête la valeur de k et le seuil retenu.
 - Fin tant que
 - Associer à chaque feuille la classe majoritaire des y_i qu'elle représente

(1) Le critère d'arrêt peut être :

- la hauteur de l'arbre (distance de la racine à la plus éloignée de ses feuilles),
- la pureté moyenne des ensembles des feuilles : $(\text{nombre max de } y_i \text{ identiques}) / (\text{nombre de } y_i)$
- l'erreur de prédiction est inférieure à un seuil donné : c'est le cas pour les Random Forest. Le principe consiste ici à mettre en place un principe de validation croisée : une partie de la base d'apprentissage est utilisée pour déterminer les seuils, l'autre partie est utilisée pour calculer le critère.

(2) Le tirage aléatoire est uniforme – toutes les caractéristiques sont équi-probables – et « avec remise ». Cela suppose qu'une caractéristique peut être sélectionnée plusieurs fois. Mais il peut être possible d'améliorer la discriminante ou la performance de l'arbre en optant pour une sélection aléatoire non-uniforme... Dans le cas des random forests, les arbres peuvent être construits en sélectionnant un nombre petit (devant le nombre de caractéristiques) m de caractéristiques sur lesquelles portent la décision. Les arbres sont ainsi spécialisés au traitement de sous-parties de l'information.

(3) Typiquement, dans le cas d'un arbre binaire (décision de type Vrai / Faux), il faut un seul seuil pour séparer en 2 classes => N=1. Pour 3 classes, il faut 2 seuils ; pour 4 classes, il faut 3 seuils, etc

2. Reconnaissance

Pour chaque arbre, soit x la donnée à tester. On calcule $f(x)$ et on compare les valeurs du vecteur de description à chaque seuil retenu pour chaque caractéristique k choisie lors de la construction de l'arbre.

Retenir la classe majoritairement renvoyée par les arbres comme réponse du classifieur global.

Valeurs typiques : blocs de 24x24 – 10000 images de visages vs 10000 contrexemples – arbres de 100 sommets environ -> perf voisines à ADABOOST, mais temps d'apprentissage passant d'une journée à une minute sur un CPU identique.

Exercice :

On considère un ensemble de 9 images dont 5 seulement représentent des visages. On annote manuellement ces images en indiquant par un '1' la présence d'un visage et par un '0' le cas contraire. On effectue sur ces images 3 mesures k_1 , k_2 et k_3 reportées dans le tableau ci-dessous :

X	1	2	3	4	5	6	7	8	9
Y	0	0	0	0	1	1	1	1	1
K1	1	2	3	1	2	3	1	2	3
K2	1	1	1	1	1	2	2	2	2
K3	2	2	2	1	1	1	1	1	1

Question 1 : Construire une « random forest » de 3 arbres équilibrés (1 racine 2 sommets intermédiaires et 4 feuilles) en utilisant le critère de pureté et un générateur de nombre aléatoire renvoyant successivement les valeurs 1 2 3 2 3 1 3 1 2.

Question 2 : Donnez l'expression du classifieur fort produit selon l'algorithme ADABOOST avec 2 classifieurs faibles. Pour simplifier les calculs, on considerera que tous les poids sont initialisés à 1/9 lors de la première boucle.

Question 3 : On dispose de 3 images à tester. Les valeurs de $[k_1 \ k_2 \ k_3]$ sur chacune d'elles sont [1 1 1], [2 2 2], [3 3 1]. Quelle décision est renvoyée par ADABOOST et la Forêt aléatoire sur ces exemples.

C. Boosting

1. Principes généraux

- Algorithme glouton pour la recherche et la pondération des caractéristiques en fonction de leur utilité pour la détection lors de l'apprentissage
- Repose sur la définition de « classifieurs faibles » = arbre de décision de hauteur 1.
- Repose sur la définition d'un classifieur fort = fusion tardive des décisions des classifieurs faibles pondérées en fonction de leur fiabilité.

2. Exemple : ADABOOST (Boosting adaptatif)

Définition d'un classifieur faible :

$$h(x, f, p, \theta) = \begin{cases} 1 & \text{si } p \cdot f(x) < p \cdot \theta \\ 0 & \text{sinon} \end{cases}$$

x est une image, f est une caractéristique mesurée sur l'image, θ est le seuil, p est la polarité ($\in \{+1, -1\}$) \Leftrightarrow permet de définir si θ est un seuil haut ou un seuil bas)

Apprentissage

Etant donnés des couples de la base d'apprentissage $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ où x_i est une image et où y_i est la connaissance liée à la présence d'un concept à détecter dans une image (= 0 ou 1). Soit T le nombre de classificateurs faibles à construire.

- Initialiser les poids $\omega_{0,i} = \begin{cases} \frac{1}{2m} & \text{si } x_i \text{ est un des } m \text{ exemples positifs} \\ \frac{1}{2l} & \text{si } x_i \text{ est un des } l \text{ exemples négatifs} \end{cases}$
- Pour t allant de 1 à T
 - Normaliser les poids $\omega'_{t,i} = \frac{\omega_{t,i}}{\sum_j \omega_{t,j}}$ pour tout i , puis $\omega_{t,i} = \omega'_{t,i}$ pour tout i .
 - Sélectionner le classificateur faible qui minimise l'erreur de classification :
$$\varepsilon_t = \min_{f,p,\theta} \sum_i \omega_{t,i} |h(x_i, f, p, \theta) - y_i|$$

(cf plus loin pour la sélection de f , p et θ)

$$h_t(x) = h(x, f_t, p_t, \theta_t) \text{ avec } (f_t, p_t, \theta_t) = \arg \min_{f,p,\theta} \sum_i \omega_{t,i} |h(x_i, f, p, \theta) - y_i|$$
 - Mise à jour des poids : $\omega_{t+1,i} = \omega_{t,i} \cdot \beta_t^{1-e_i}$

$$e_i = \begin{cases} 0 & \text{si } x_i \text{ est classé correctement} \\ 1 & \text{sinon} \end{cases}$$

Avec $\beta_t = \frac{\varepsilon_t}{1 - \varepsilon_t}$
- Fin Pour

3. Reconnaissance

Le classificateur fort final est obtenu par fusion des décisions des classificateurs faibles :

$$C(x) = \begin{cases} 1 & \text{si } \sum_{t=1}^T \alpha_t \cdot h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \alpha_t \\ 0 & \text{sinon} \end{cases} \quad \text{avec } \alpha_t = \ln\left(\frac{1}{\beta_t}\right)$$

T détermine la qualité attendue du classificateur (plus T est grand, plus le classificateur fort est précis). Dans la pratique, T détermine le nombre de caractéristiques utilisées par le classificateur.

4. Mise en œuvre de l'apprentissage

Pour la recherche du classificateur faible optimal minimisant l'erreur de classification :

Calculer les valeurs d'une caractéristique donnée sur toutes les images annotées

Ordonner les valeurs de la caractéristique

Définir le meilleur seuil θ pour séparer au mieux les exemples positifs et les exemples négatifs.

En déduire la valeur de p .

Exemple :

$$f = [53 \quad 45 \quad 27 \quad 18 \quad 12 \quad 5 \quad -1 \quad -8 \quad -15 \quad -23 \quad -34]$$

$$y = [1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0]$$

On teste toutes les valeurs de theta correspondant aux centres des intervalles définis par les valeurs ordonnées de f :

Entre 53 et 45 = 4 erreurs si $p < 0$ et 7 erreurs si $p > 0$

Entre 45 et 27 = 3 erreurs si $p < 0$ et 8 erreurs si $p > 0$

Entre 27 et 18 = 4 erreurs si $p < 0$ et 7 erreurs si $p > 0$

Entre 18 et 12 = 3 erreurs si $p < 0$ et 8 erreurs si $p > 0$

Entre 12 et 5 = 2 erreurs si $p < 0$ et 9 erreurs si $p > 0$

Etc...

⇒ on prend theta entre 12 et 5 (8 par exemple) et $p < 0$ pour cette feature.

Remarque : on suppose dans cet exemple que les poids associés à chaque exemple sont égaux pour calculer l'erreur. Ce n'est pas le cas en général. Pour implémenter cette méthode de manière efficace en prenant en compte une pondération des erreurs, le seuil est choisi comme minimisant le coût défini par :

$$e = \min (S^+ + (T^- - S^-), S^- + (T^+ - S^+))$$

où :

- T^+ est la somme des poids des exemples positifs (1),
- T^- est la somme des poids des exemples négatifs (0),
- S^+ est la somme des poids des exemples positifs jusqu'à la position du seuil testé,
- S^- la somme des poids des exemples négatifs jusqu'à la position du seuil testé.

Retour à l'exemple précédent en considérant que tous les poids sont à 1 : pour θ testé = 8, on a

- $T^+ = 5$ (constant),

- $T_- = 6$ (constant),
- $S_+ = 4$,
- $S_- = 1$.
- $\Rightarrow e = \min(4 + (6-1), 1 + (5-4)) = \min(9, 2) = 2$

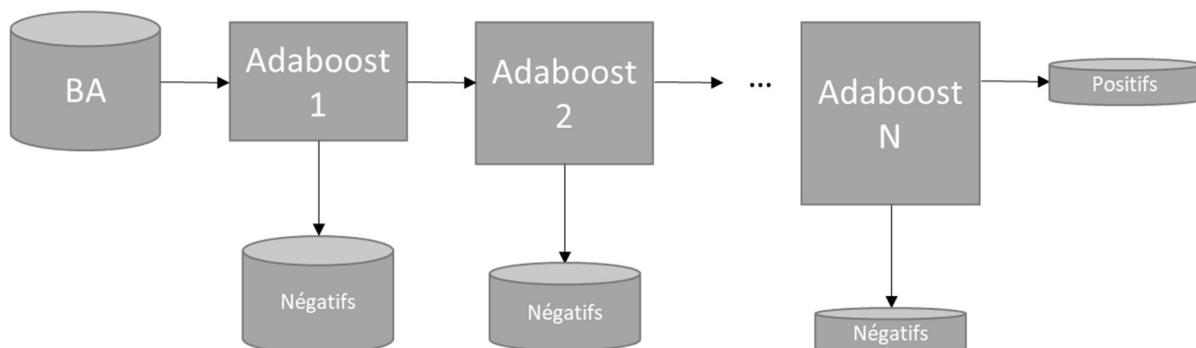
- Problème 1 : K caractéristiques et N exemples pour l'apprentissage $\Rightarrow K \times N$ valeurs à comparer et trier (Typiquement, pour un détecteur de visages avec des performances acceptables, $T=200$, $K=160000$ pour un bloc de 24×24 pixels et $N = 10000$ blocs minimum $\Rightarrow 1,6 \cdot 10^9$ valeurs à trier 200 fois en phase d'apprentissage... \Rightarrow apprentissage très long, mais ce n'est « pas grave »...)
- Problème 2 : Avec $T=200$, Il y a 200 caractéristiques à extraire de l'image, puis à comparer avec des seuils pour obtenir les décisions faibles qu'il faut ensuite pondérer pour prendre la décision finale \Rightarrow Calcul bien plus limité que celui de l'apprentissage mais encore prohibitif pour une exploitation en temps interactif (affichage d'un rectangle autour des visages visibles dans l'affichage d'une sortie caméra sur un smartphone par exemple) \Rightarrow mise en place d'une cascade de classificateurs.

5. Cascades de Haar

Le principe repose sur l'hypothèse qu'une image à une très faible probabilité de représenter le concept à détecter (si ce n'est pas le cas, cette méthode n'a aucun intérêt). L'idée est de construire des classificateurs peu coûteux et très fiables pour éliminer les cas clairement négatifs (et supposés être majoritaires) et de n'appliquer les classificateurs plus coûteux que sur le petit pourcentage des cas positifs ou « moins clairement » négatifs.

1. Construction d'un classifieur fort à partir de 2 classificateurs faibles ($T=2$) et modification progressive du seuil de décision pour retenir tous les exemples positifs (quitte à avoir beaucoup de faux positifs). Rejet d'une certaine quantité de vrais négatifs.
2. On augmente T et on construit un nouveau classifieur fort qu'on applique sur les exemples retenus à l'itération précédente. On adapte le seuil pour retenir tous les exemples positifs et on élimine ainsi une nouvelle proportion de cas négatifs. Si la proportion de cas négatifs éliminée est inférieure à un seuil, on recommence en 2.
3. On réitère le processus jusqu'à ce que la proportion de vrais positifs dans les exemples retenus dépasse un certain seuil.

Par exemple, pour obtenir les mêmes performances qu'un seul classifieur fort avec $T=200$, 160.000 caractéristiques sur une image, on construit une cascade de 38 niveaux pour la détection de visages où le dernier niveau a un $T=40$ (environ). (Version 2023 d'OpenCV : 24 étages / 9 features au premier étage et 200 features au dernier étage)



3 paramètres à fixer :

F_{cible} = taux de faux positifs admissible dans les résultats

f = pourcentage de réduction du taux de faux positifs entre 2 niveaux consécutifs de la cascade

d = pourcentage d'augmentation du taux de vrais positifs entre 2 niveaux de la cascade

Algorithme :

$F_0 = 1$

$D_0 = 1$

$i = 0$

P = sous-ensemble de la BA composé des cas positifs

N = sous-ensemble de la BA composé des cas négatifs

Tant que $F_i > F_{cible}$

$i = i + 1$

$n_i = 0$

$F_i = F_{i-1}$

 Tant que $F_i > f * F_{cible}$

$n_i = n_i + 1$

 Utiliser P et N pour entraîner un classifieur fort avec $T=n_i$ descripteurs de Haar à l'aide de l'algorithme Adaboost

 Evaluer la cascade avec le classifieur fort ainsi produit pour déterminer D_i = taux de vrais positifs de ce classifieur

 Diminuer le seuil de décision du classifieur fort de manière à ce que $D_i \geq d * D_{i-1}$

 Evaluer F_i avec le classifieur fort ainsi obtenu

 Fin tant que

$N = \{\}$

 Si $F_i > F_{cible}$ alors évaluer la cascade sur l'ensemble des exemples négatifs et ajouter les faux positifs dans N

Fin tant que

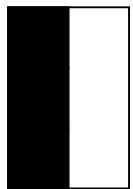
Commentaires :

- le premier while permet d'estimer chaque classifieur de la cascade. Dans l'implantation OpenCV, le premier while a bouclé 38 fois.

- Le second while permet de définir 1 classifieur fort de la cascade. Dans l'implantation, le premier classifieur fort comporte 2 features

Pourquoi parle-t-on de cascade de « Haar » ? Parce que la méthode a été mise au point pour sélectionner les filtres inspirés des filtres de Haar parmi les plus pertinents sur des centaines de millier pour la détection d'un concept. Pour construire ces filtres :

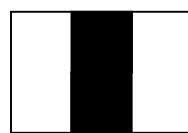
- On définit des « motifs de base » rectangulaires comportant des sous-zones rectangulaires blanches et noires. La surface des zones noires doit être égale à la surface des zones blanches.



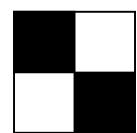
A



B



C



D

Pour chaque motif :

- On identifie dans un bloc d'image toutes les zones rectangulaires correspondant au format du motif. Une telle zone sera appelée « filtre de haar » et correspondra à 1 feature.
- On calcule la somme des coefs. de la zone blanche – coefs de la zone noire pour obtenir la valeur de la feature.

Sur un bloc de 24x24 pixels, les 4 motifs ci-dessus permettent de générer environ 160.000 features

Exercice : calculer toutes les features sur le bloc 3x3 ci-dessous à l'aide des 4 motifs proposés :

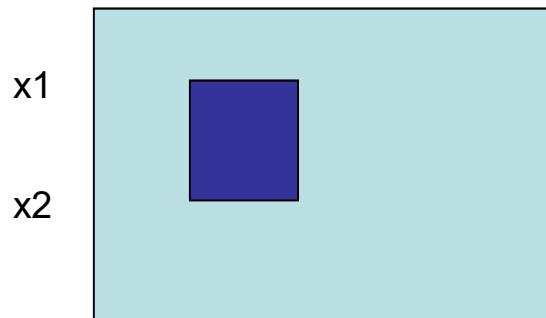
1	0	1
0	0	0
0	1	0

— Ces features sont rapides à calculer grâce à l' « image intégrale »

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y')$$

— Calcul de la somme des coef d'un bloc

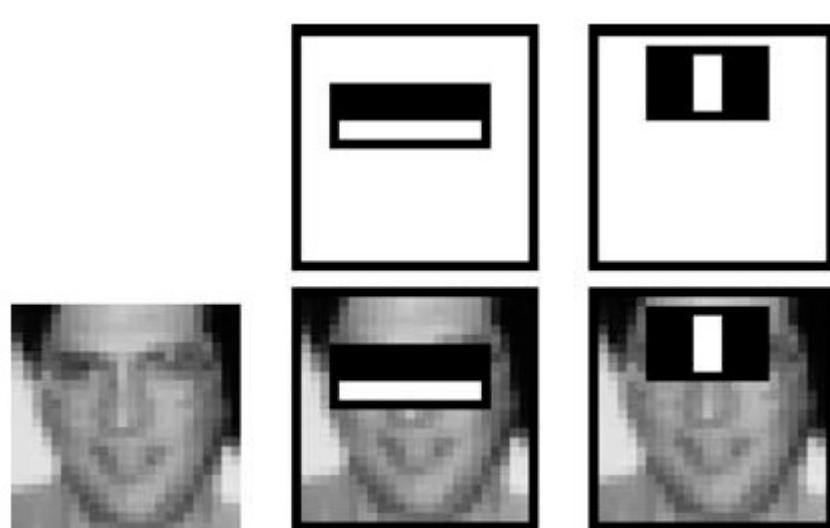
$y_1 \quad y_2$



Somme_bloc =

$$ii(x_2, y_2) + ii(x_1, y_1) - (ii(x_1, y_2) + ii(x_2, y_1))$$

Lorsqu'on applique une cascade de Haar sur des blocs 24x24 représentant des visages, le premier niveau de la cascade va retenir les 2 features suivantes :



D. Classification Bayesienne

Exemple : Modèles de Markov

E. Votes

Exemples : knn, Hough

1. Objectifs et principes

On dispose d'un modèle paramétrique supposé représenter la manière selon laquelle se distribue des observations. Le but est de déterminer la valeur des paramètres qui permet d'ajuster le modèle aux observations réelles – il s'agit d'un problème de régression. La transformée de Hough est un « algorithme glouton » qui consiste à estimer ces paramètres par une méthode de vote.

A l'origine, cette méthode a été développée pour « vectoriser » les contours dans une image binaire (après seuillage des gradients par exemple). Les contours des objets sont représentés par des courbes décrites à l'aide de fonctions simples (droites, arcs de cercles, ...) modélisées par leurs paramètres.

On transforme l'image dans l'espace des paramètres et on identifie la courbe dans cet espace.

Par exemple, les droites passant par un point (x,y) ont toutes une équation de la forme $y=m.x+c$

Algorithme

Etape 1. Modélisation

- une droite est représentée par une équation paramétrique
- on discrétise l'espace des paramètres (appelé espace de Hough) (on établit un pas pour chacune des 2 dimensions).
- Chacun des points (appelé « accumulateur ») de l'espace discrétisé de Hough se voit attribué le score de 0.

Etape 2. Reconnaissance

- Pour chacun des points de l'image des contours :
 - o On établit l'équation paramétrique correspondante dans l'espace de Hough
 - o On ajoute 1 au score des points de l'espace de Hough dont les coordonnées vérifient l'équation
- Les droites correspondant à des contours sont données par les points de l'espace de Hough ayant le plus grand coefficient.

Il existe une variante possible pour l'étape 2 :

- Pour chacun des accumulateurs de l'espace de Hough, on détermine l'équation paramétrique correspondante.
- On incrémente l'accumulateur pour chaque pixel à 1 de l'image dont les coordonnées vérifient l'équation.

Formellement, la Transformée de Hough conduit à construire une représentation $H(a,b)$ définie par

$$H(a,b) = \sum_x \sum_y I(x,y) \delta(ax + b - y) \text{ où } \delta(ax + b - y) \text{ est le symbole de Kronecker (i.e.}$$
$$\delta(x) = \begin{cases} 1 & \text{si } x = 0 \\ 0 & \text{sinon} \end{cases}$$

Exercice: Principes de base de la transformée de Hough

Question 1 : On considère une image de 5×5 pixels repérés par leurs coordonnées comprises entre 0 et 4. Soit P_1 le pixel de coordonnées $(2,3)$. On utilise les paramètres a et b pour représenter l'équation d'une droite sous la forme $y = a.x + b$. Donner une équation de la droite représentant le point P_1 (i.e. l'ensemble des droites passant par ce point) dans l'espace paramétrique associé.

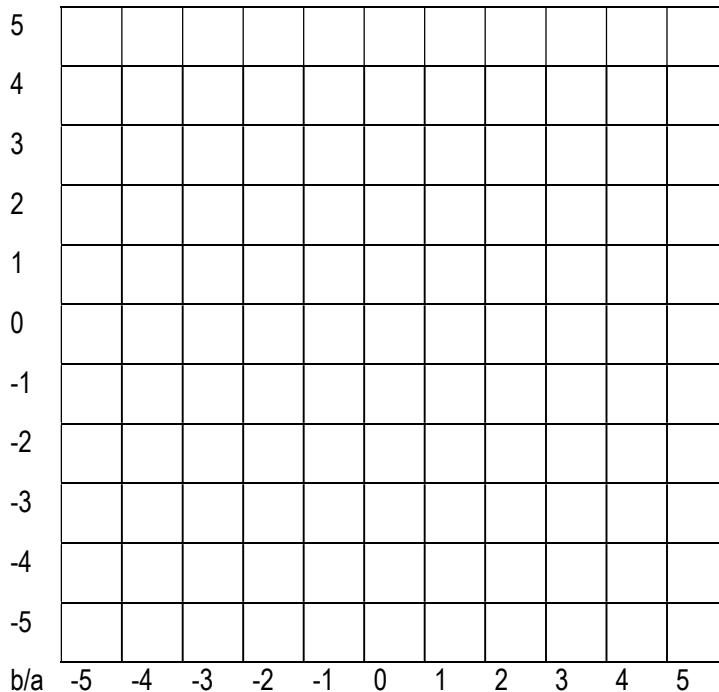
Question 2 :

Soi P_2 , le point de coordonnées $(1,1)$. Déterminer le point d'intersection de la droite définie à la question 1 avec celle représentant le point P_2 dans l'espace des paramètres. En déduire l'équation de la droite passant par P_1 et P_2 . Vérifier le résultat.

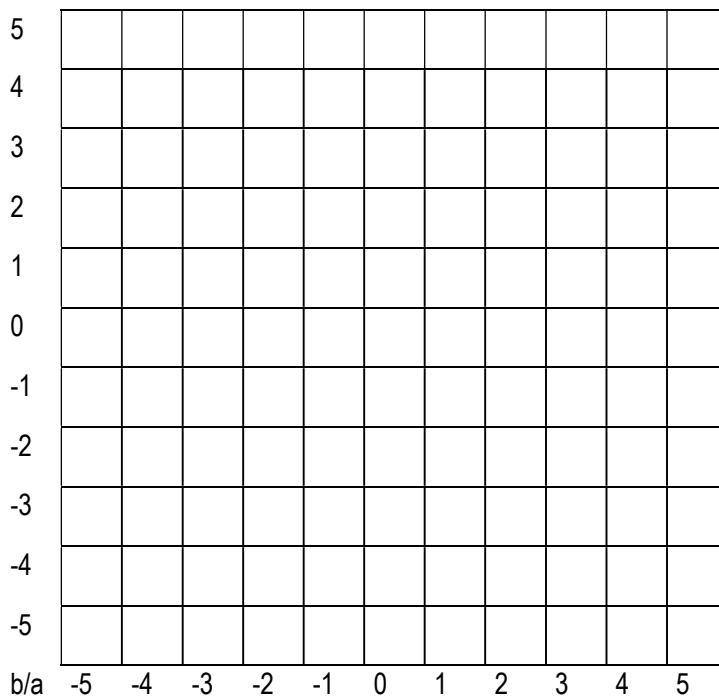
Exercice 2 : Mise en oeuvre

L'espace de Hough est représenté par une matrice 11×11 où le coefficient de pente a et le coefficient de translation b peuvent prendre des valeurs entières comprises entre -5 et 5 . Donner la configuration de l'espace de Hough lorsque les points de contours de l'image sont les suivants :

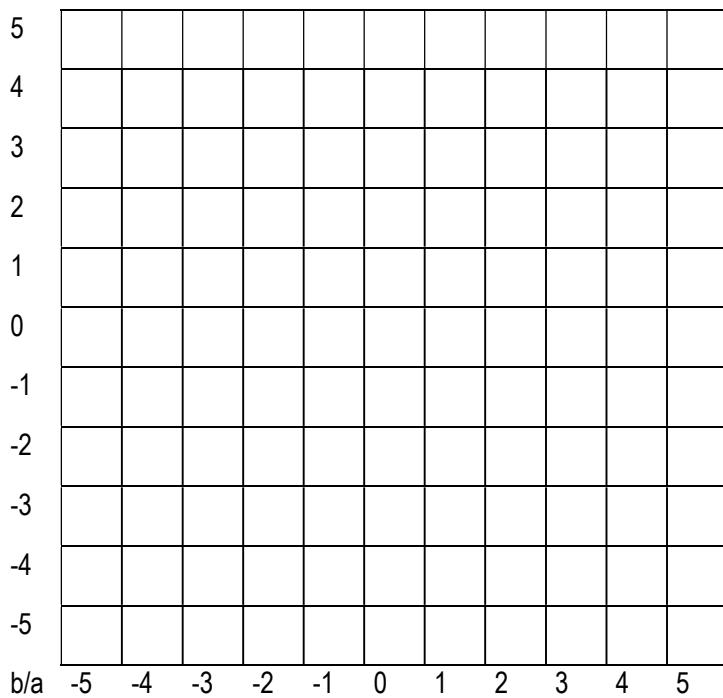
- 1) P1(1,0), P2(2,2), et P3(3,4)



- 2) P1(1,0), P2(2,2), et P3(0,0)



3) P1(1,0), P2(1,2), et P3(1,4)

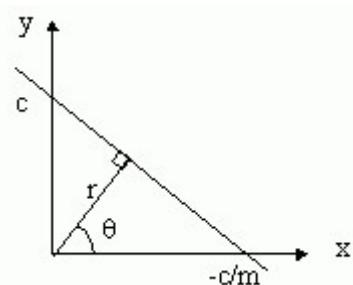


2. Transformée de Radon

Une modélisation des droites sous la forme $y=ax+b$ pose le problème de la discréétisation des paramètres a et b (valeur potentiellement infinie, progression non-linéaire). Pour palier ce problème, on utilise très fréquemment une représentation en coordonnées polaires : $x \cos\theta + y \sin\theta = \rho$

$0 < \rho <$ diagonale de l'image

$0 < \theta < \pi$



On exprime ainsi une extension de la transformée de Hough appelée Transformée de Radon sous la forme :

$$R(\rho, \theta) = \sum_x \sum_y I(x, y) \delta(x \cos \theta + y \sin \theta - \rho)$$

La Transformée de Radon a été développée initialement pour modéliser les informations disponibles dans les images de tomographie (rayons X). Selon le type d'étude, $I(x, y)$ peut être une image en niveaux de gris. A partir des informations disponibles dans R , il est possible de reconstruire l'image de départ par la « projection » inverse :

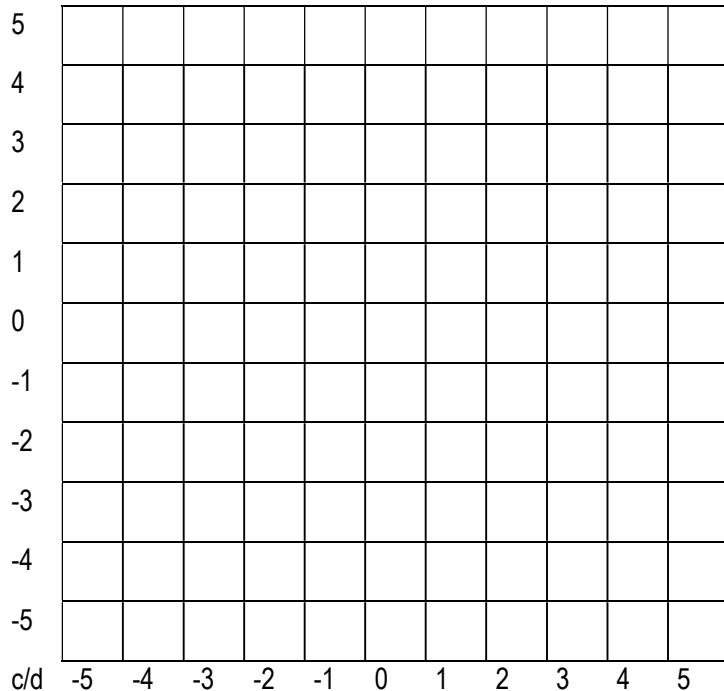
$$I(x, y) = \sum_{\theta} R(x \cos \theta + y \sin \theta, \theta)$$

3. Extension aux formes circulaires

$$(x - a)^2 + (y - b)^2 = r^2 \quad \Rightarrow 3 \text{ paramètres : } a, b, r$$

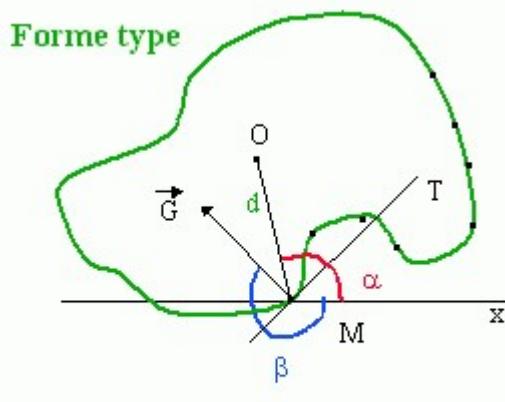
Exercice : On s'intéresse à des contours de forme circulaire de rayon 2 que l'on décrit à l'aide de l'équation $(x - c)^2 + (y - d)^2 = 2^2$ où c et d définissent l'espace des paramètres. Donner pour les trois points suivants, la configuration de l'espace de Hough associé :

P1(0,2), P2(2,0), P3(4,2)



4. Transformée de Hough généralisée

Modélisation



O : centre de gravité

MT : tangente au contour en M

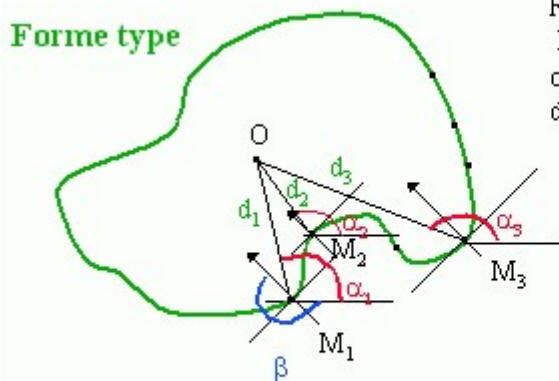
$G \rightarrow$: normale à MT
(direction du gradient)

$$d = \overline{MO}$$

α : angle sous lequel est vu le centre de gravité

β : angle de la normale au contour avec l'horizontale

Modèle = liste de triplets (β, α, d)



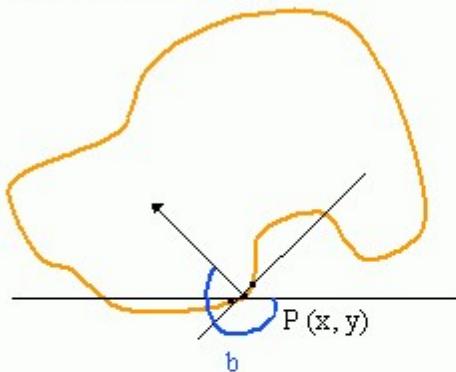
Remarque :

Plusieurs points du contour-modèle ont le même angle β , avec des valeurs différentes pour α_i et d_i .

Pour représenter le modèle, on regroupera les triplets (β, α, d) ayant la même valeur β .

Reconnaissance

Forme à reconnaître



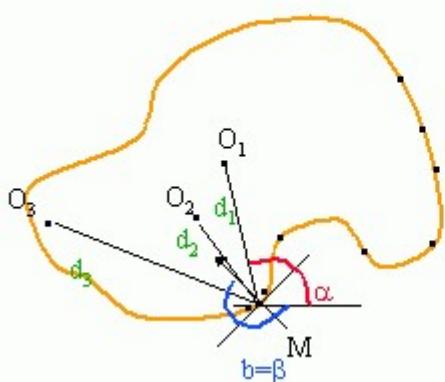
L'espace des paramètres est celui des coordonnées du centre de gravité O

Pour chaque point P, on calcule l'angle **b** que fait la normale au contour en P avec l'horizontale (on estime la direction du contour à l'aide des 2 points adjacents à P sur le contour).

On parcourt la forme modèle et on cherche les triplets (b, α, d) . Ce sont les points de contours du modèle qui ont une normale parallèle à celle de P. On peut donc dire que la portion de contour centrée sur P vote pour un centre de gravité dont les coordonnées sont données par α et d : $x_0 = x + d \cdot \cos(\alpha)$
 $y_0 = y + d \cdot \sin(\alpha)$

Si beaucoup de points de la forme votent pour ce même point, la forme sera reconnue.

Forme à reconnaître



L'angle b au point M de la forme correspond à l'angle β du modèle. Donc si M appartient à une forme correspondant au modèle, il vote pour des centres de gravité qui seraient en O_1 ou O_2 ou O_3 .

Algorithme

A- Modélisation

- On regroupe les triplets du modèle ayant la même valeur β et on les range dans un tableau T dont les lignes sont indiquées par les différentes valeurs de β .

T	β_1	(α_{11}, d_{11})	(α_{12}, d_{12})	\dots	(α_{1j}, d_{1j})
	β_2	(α_{21}, d_{21})	(α_{22}, d_{22})	\dots	(α_{2k}, d_{2k})
	β_n	(α_{n1}, d_{n1})	(α_{n2}, d_{n2})	\dots	(α_{nj}, d_{nj})

B- Reconnaissance

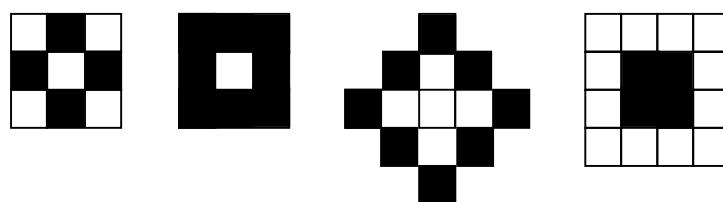
- On initialise un tableau A [$x_{min}, x_{max}, y_{min}, y_{max}$]
- Pour chaque point P(x,y) du contour,
 - on calcule l'angle b , ce qui fournit une entrée ($b=\beta$) dans la table T
 - pour chaque couple de la ligne β de T,
 - calculer $x_0 = x + d \cdot \cos(\alpha)$
 - $y_0 = y + d \cdot \sin(\alpha)$
 - incrémenter A [x_0, y_0]
- Déterminer le couple (x,y) qui maximise A[x, y]

illustration : Patrice Dalle

Exercice : Le tableau suivant ne contient qu'une partie seulement des données permettant de modéliser une forme (les angles sont exprimés en degré) :

β	liste des couples (α, d)
0	(0,1)
90	(90,1)
180	(180,1)
270	(270,1)

Compte-tenu de ces informations, quelle est (quelles sont) parmi les formes suivantes celle(s) qui correspond(ent) au modèle complet ? On considère que le contour de la forme est représenté par les pixels en noir.



Exercice : Une image représentant un circuit imprimé montre en particulier des composants carrés. On suppose qu'idéalement ces composants apparaissent sous la forme de carrés de 3x3 pixels dont les côtés sont alignés selon les axes verticaux et horizontaux (cf. matrice ci-dessous). Afin de détecter leur contour et leur position, on choisit d'utiliser la transformée de Hough. Donner une expression du modèle. Quelles sont les valeurs dans l'espace de Hough lorsque l'image des contours est celle correspondant à la matrice ci-dessous (précisément un carré de 3x3) ?

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

Vous indiquerez la nature des paramètres utilisés et les raisons qui motivent les différents choix que l'on peut être amené à faire pour implémenter un algorithme de reconnaissance de ces contours carrés, basé sur la transformée de Hough.

F. Transformée en distance

1. Distance, voisinage et connexité

a) Les k-distances

On considère une image comme étant un graphe où les sommets sont les pixels et où les arêtes joignent des pixels voisins. La notion de k-voisins désigne la relation d'adjacence définissant le voisinage (4- ou 8-voisins). Un k-chemin est un chemin de n pixels xi adjacents (au sens des k-voisins) reliant un pixel p à un pixel q où $x_0 = p$ et $x_n = q$.

Remarques :

- Un k-chemin de longueur n traverse donc n+1 pixels et n arêtes.
- Un k-chemin de longueur 1 est composé de 2 pixels voisins.
- Tout 4-chemin est un 8-chemin.

La k-distance entre deux pixels p et q est la longueur du plus petit k-chemin entre ces deux pixels. C'est une extension de la définition de la distance géodésique étendue à toute l'image (et non plus seulement à une région).

Une k-distance vérifie les axiomes de définition des distances :

- $d_k(p,p) = 0$;
- $d_k(p,q) > 0$ si p différent de q ;
- $d_k(p,q) = d_k(q,p)$;
- $d_k(p,r) \leq d_k(p,q) + d_k(q,r)$.

On a :

- $d_4(p,q) = |i-i'| + |j-j'|$ pour $p = (i,j)$ et $q = (i',j')$
- $d_8(p,q) = \max(|i-i'|, |j-j'|)$

La distance euclidienne définie par $d_e(p,q) = \sqrt{((i-i')^2 + (j-j')^2)}$ ne se prête pas nécessairement bien à toutes les mesures de distance dans l'espace discret. Par exemple, la distance géodésique introduite dans le cours 6 permet de prendre en compte la connexité d'une forme.

Exercice : Etant donné deux pixels $p(i,j)$ et $q(i',j')$. Indiquer laquelle des trois distances d_4 , d_8 et d_e renvoie la valeur la plus grande et laquelle renvoie la valeur la plus petite. Justifier en identifiant l'ensemble des pixels équidistants d'un pixel p central au sens des 3 distances.

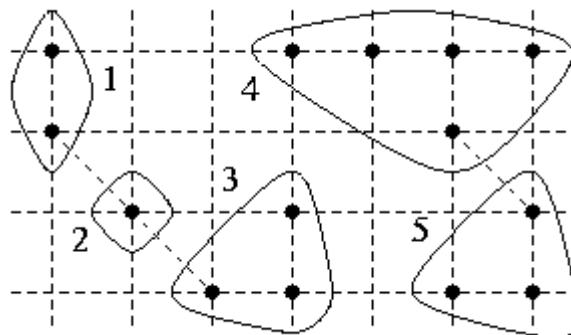
b) Les k -connexité

Une région X est dite k -connexe s'il existe un k -chemin inclus dans X reliant tout couple de points (p,q) de X .

Toute région 4-connexe est 8-connexe.

On appelle une composante k -connexe de X une partie k -connexe maximale de X . Si X est non-vide, les composantes k -connexes de X forment une partition de celui-ci, c.à.d. elles sont non-vides, mutuellement disjointes, et recouvrent X . Pour un pixel p de X , la composante k -connexe de X contenant p est la réunion de toutes les parties k -connexes de X contenant p .

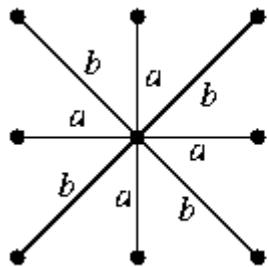
Une composante 8-connexe de X est la réunion d'une ou plusieurs composantes 4-connexes de X reliées entre elles par des connexions diagonales.



Dans cet exemple, les composantes 4-connexes sont les ensembles numérotés de 1 à 5, tandis que les composantes 8-connexes sont d'une part la réunion des ensembles 1 à 3, d'autre part la réunion des ensembles 4 et 5.

c) Chanfrein 3x3

Pour réduire l'écart entre la distance euclidienne et les k-distances, on propose de pondérer l'"adjacence" entre un pixel et ses k-voisins à l'aide de deux poids différents :



<i>b</i>	<i>a</i>	<i>b</i>
<i>a</i>	0	<i>a</i>
<i>b</i>	<i>a</i>	<i>b</i>

Exercice : Proposer une valeur pour *a* et *b* de manière à retrouver la 4-distance et la 8-distance. Déterminer l'ensemble des points équidistants d'un pixel *p* par la distance de chanfrein.

Cette matrice de coefficients est appelée le "masque de chanfrein". On étend ici l'idée mise en œuvre pour définir les filtres de Prewitt, Sobel et Frei-Chen.

Des contraintes sur les valeurs de *a* et *b* ont été définies (sous le nom de "Conditions de Montanari") pour garantir des propriétés minimales de la distance comme la convexité du cercle unité. Ces contraintes sont : $a > 0$ et $a \leq b \leq 2a$.

d) Chanfrein de Borgefors

Borgefors a ajouté à ces conditions deux critères :

- la réduction de l'écart entre distance de chanfrein et distance euclidienne.
- les coefficients du masque de chanfrein doivent être des nombres rationnels ayant de petits numérateurs et dénominateurs entiers (pour des raisons de vitesse de calcul).

Le premier critère est vérifié de façon optimale en choisissant $a = 0,95509$ et $b = 1,336930$.

L'approximation de ces valeurs par des rationnels "simples" peut être la suivante : $a = 1$ et $b = 4/3$.

Pour réduire le nombre de calculs, on prend $a=3$ et $b=4$ pour obtenir une estimation du triple de la distance euclidienne. On parle alors de $d_{3,4}$ de Borgefors.

Si on utilise un masque 5x5 pour le calcul de la distance de chanfrein, on montre qu'il est nécessaire d'estimer la valeur de 5 coefficients différents, ce qui, en réalité, conduit à estimer 3 coefficients effectifs : *a*, *b* et *c*.

$$\begin{array}{ccccccccc}
 e & c & d & c & e & & 2b & c & 2a & c & 2b \\
 c & b & a & b & c & & c & b & a & b & c \\
 d & a & 0 & a & d & \text{soit} & 2a & a & 0 & a & 2a \\
 c & b & a & b & c & & c & b & a & b & c
 \end{array}$$

e c d c e 2b c 2a c 2b

En fixant $a=1$, puis en prenant $b= 7/5$ et $c=11/5$, les critères de Borgefors sont satifaits. En multipliant par 5 ces coefficients, on calcul la d_{5,7,11} de Borgefors.

e) Mise en œuvre

On définit ainsi la distance $d(p,X)$ comme le minimum de la distance de p aux pixels de X :

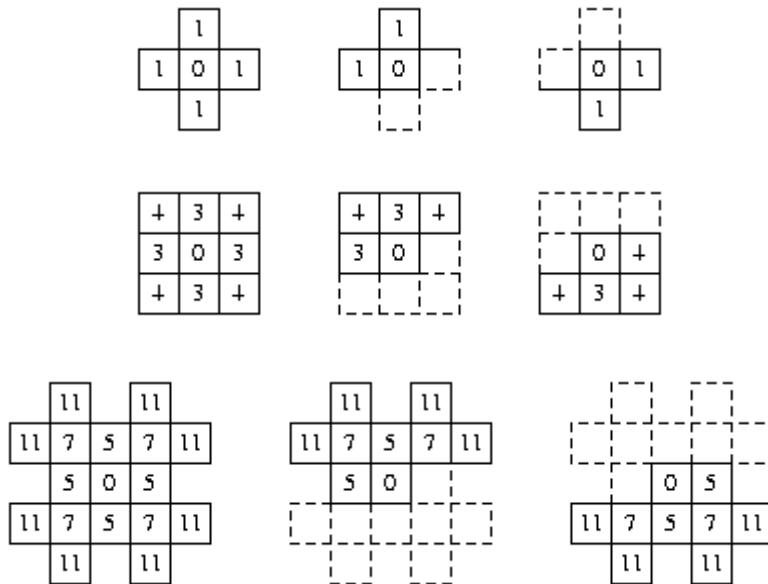
$$d(p,X) = \min \{d(p,x) \mid x \text{ dans } X\}.$$

Soient S une partie de Z_2 et R une partie de S . La transformée de distance de R dans S est la fonction TFD _{R,S} donnée par :

$$\text{TFD}_{R,S}(p) = d(p,R) \text{ pour tout pixel } p \text{ de } S.$$

L'ensemble R est appelé marqueur, tandis que l'ensemble S est appelé domaine.

Pour calculer la TFD, on décompose le masque de chanfrein en deux sous masques : le masque postérieur M_+ et le masque antérieur M_- (associé au voisinage postérieur V_+ ou antérieur V_-):



On applique ensuite l'algorithme suivant :

```
/* Initialisation */

Pour y = 1 à Nb_lignes
Pour x = 1 à Nb_colonnes
Si pixel(x,y) appartient à R, alors TFD[x,y] <- 0
Sinon TFD[x,y] <- + infini (une marque non numérique
```

```

ou supérieure à la plus grande des distances possibles =
Nb_lignes + Nb_colonnes + 1)

Fin pour

Fin pour

/* Balayage descendant */

Pour y = 1 à Nb_lignes

Pour x = 1 à Nb_colonnes

TFD[x,y] <- min(TFD[x',y']+M - pour tout (x',y') de Vde
(x,y))

Fin pour

Fin pour

/* Balayage ascendant */

Pour y = Nb_lignes à 1

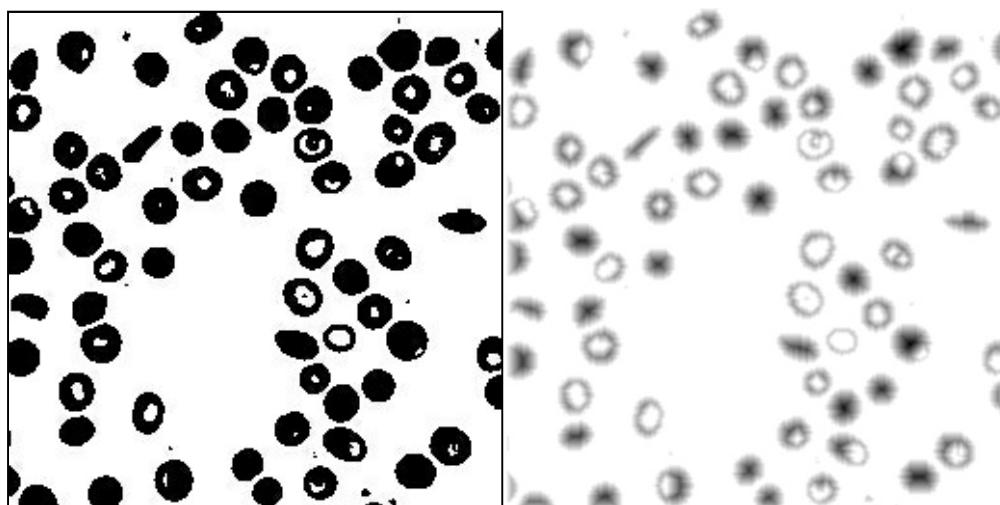
Pour x = Nb_colonnes à 1

TFD[x,y] <- min(TFD[x',y']+M + pour tout (x',y') de V+
de (x,y))

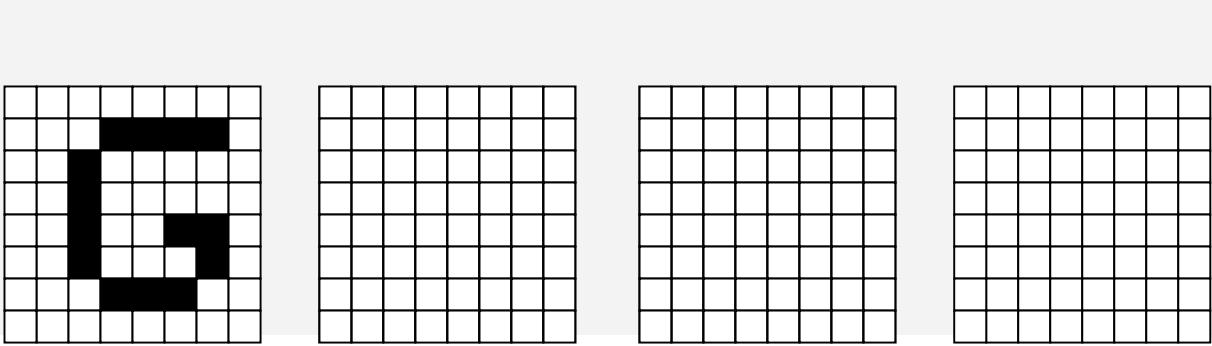
Fin pour

Fin pour

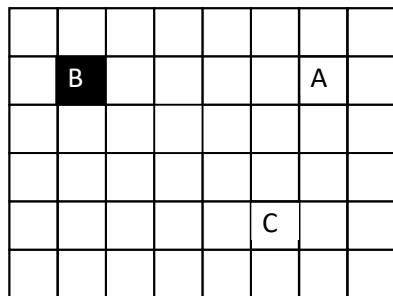
```



Exercice : Calculer le résultat de l'application de la TFD lorsque le marqueur est la région en noir dans l'image suivante, lorsqu'on utilise la $d_{3,4}$ de Borgefors :



Exercice : On souhaite analyser la précision de différentes transformées en distance. Pour cela, on considère sur une image binaire trois points A, B, et C.



Seul le pixel correspondant au point B est à 1, tous les autres (y compris les pixels correspondant à A et C) sont à 0.

Question 1 : Montrer que $AB = BC$ en distance euclidienne.

Question 2 : Calculer la transformée en distance de l'image en utilisant tour à tour :

D1 : la 8-distance ($d_{1,1}$)

D2 : la 4-distance ($d_{1,2}$)

D3 : la $d_{3,4}$ de Borgefors

Question 3 : en fonction des résultats de la question 1, indiquer laquelle de ces trois distances vérifie "le mieux" l'égalité de la question 1. Quelle distance vérifie "le moins" cette égalité ?

Question 4 : Que se passe-t-il si on compare ces trois distances à l'aide de l'exemple suivant ?

Exercice :

Soit l'image binaire I :

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

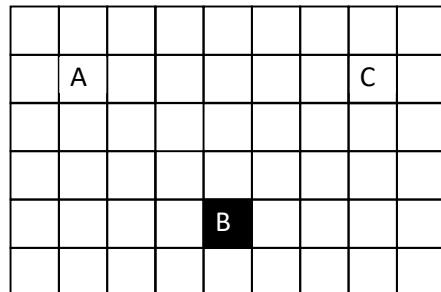
On définit un masque de chanfrein 3x3 par

$$\begin{bmatrix} b & a & b \\ a & 0 & a \\ b & a & b \end{bmatrix}$$

Question 1 : donnez le résultat de la transformée en distance à l'aide de ce masque lorsque $a=2$ et $b=4$. Identifiez le cercle de rayon 6 sur le résultat.

Question 2 : même question avec $a=4$ et $b=2$.

Question 3 : Montrez que si $b > 2a$ alors la solution d'une transformée en distance par masque de chanfrein est identique à celle obtenue lorsque $b=2a$.



Une fois la TFD appliquée, de nombreux traitements peuvent être appliqués. En particulier, il est possible de déterminer directement la distance à l'objet le plus proche à partir d'un pixel quelconque de l'image. On peut ainsi construire des relations de dépendance ou d'influence entre les objets.

f) Application à la reconnaissance d'objets

Exemple : OCR par appariement avec une TFD :

10	7	4	3	3	3	3	4	7	10
9	6	3	0	0	0	0	3	6	9
9	6	3	0	0	0	0	3	6	9
10	7	4	3	0	0	3	4	7	10
12	9	6	3	0	0	3	6	9	12
12	9	6	3	0	0	3	6	9	12
12	9	6	3	0	0	3	6	9	12
12	9	6	3	0	0	3	6	9	12
10	7	4	3	0	0	3	4	7	10
9	6	3	0	0	0	0	3	6	9
9	6	3	0	0	0	0	3	6	9
10	7	4	3	3	3	3	4	7	10

Score « I » \ I = 6

Taux = 26/28

4	3	3	3	4	4	3	3	3	4
3	0	0	0	3	3	0	0	0	3
3	0	0	0	3	3	0	0	0	3
4	3	0	0	3	3	0	0	3	4
6	3	0	0	3	3	0	0	3	6
6	3	0	0	0	0	0	0	3	6
6	3	0	0	0	0	0	0	3	6
6	3	0	0	3	3	0	0	3	6
4	3	0	0	3	3	0	0	3	4
3	0	0	0	3	3	0	0	0	3
3	0	0	0	3	3	0	0	0	3
4	3	3	3	4	4	3	3	3	

Score « I » \ H = 15

Taux = 23/52

G. Graphes et Méthodes s'appuyant sur la logique

Exemples : Logique propositionnelle, modale, floue

H. Support Vector Machine

I. Réseaux de neurone

1. Réseaux de Neurones Numériques

Principe du neurone artificiel

Un réseau de neurones artificiels peut être vu comme un graphe de décision orienté avec :

- Une couche d'entrée (les paramètres, les caractéristiques)
- Une couche de sortie (les valeurs de décision pour chaque classe concernée, ou la valeur de régression)
- Des couches intermédiaires – dites « cachées »

Un neurone est un sommet du graphe qui envoie sur tous ses arcs sortants la même valeur calculée de la manière suivante :

$$\varphi \left(\sum_{i=1}^N \omega_i \cdot f_i + b \right)$$

où :

- f_i est la valeur reçue sur le $i^{\text{ème}}$ arc entrant du neurone
- ω_i est le poids associé à cet arc
- φ est la fonction d'activation du neurone. Il s'agit généralement d'une fonction continu dont le comportement se rapproche de celui d'une fonction binaire (ie. en dessous d'un seuil, la valeur renvoyée est nulle ; au-dessus, elle est égale à 1)
- b est le biais. Il permet d'ajuster le comportement de la fonction d'activation. En pratique, il est considéré comme étant une valeur d'entrée fixée à 1, elle-même pondérée.

Exemples de fonctions d'activation

Nom	fonction	Dérivée
Rampe	$f(x) = x$	$f'(x) = 1$
Sigmoïde	$f(x) = 1/(1+\exp(-x))$	$f'(x) = f(x)(1-f(x))$
tangente hyperbolique	$f(x) = (2/(1+\exp(-2x))) - 1$	$f'(x) = 1-f(x)*f'(x)$
unité de rectification linéaire	$f(x) = 0 \text{ si } x < 0 \text{ et } x \text{ sinon}$	$f'(x) = 0 \text{ si } x < 0 \text{ et } 1 \text{ sinon}$

Apprentissage :

L'apprentissage va consister à calculer les valeurs des poids à attribuer à chaque arc de manière à optimiser les performances du réseau de neurones sur les données de la BA. On utilise pour cela une méthode de « rétropropagation du gradient ».

Soit y la valeur attendue à la sortie du $j^{\text{ème}}$ neurone de la couche de sortie K, on calcule l'erreur :

$$e_j^K = \varphi' \left(\sum_{i=1}^N \omega_i^K \cdot f_i^K \right) \left(y - \varphi \left(\sum_{i=0}^N \omega_i^K \cdot f_i^K \right) \right)$$

Pour les neurones des couches intermédiaires, l'erreur est calculée avec :

$$e_m^{k-1} = \varphi' \left(\sum_{i=1}^N \omega_i^{k-1} \cdot f_i^{k-1} \right) \left(\sum_{j=0}^N \omega_j^k \cdot e_j^k \right)$$

(Peut être différent en cas d'utilisation d'un optimiseur et d'une fonction de loss spécifique)

Les poids des arcs entrants du neurone sont modifiés par la formule suivante

$$\omega_{j,t+1}^k = \omega_{j,t}^k + \lambda \cdot e_{j,t}^k \cdot f_{j,t}^k$$

Où :

- λ est le taux d'apprentissage (ie. un pourcentage de modification accepté à chaque étape) à fixer (en général entre 0.01 et 0.001)

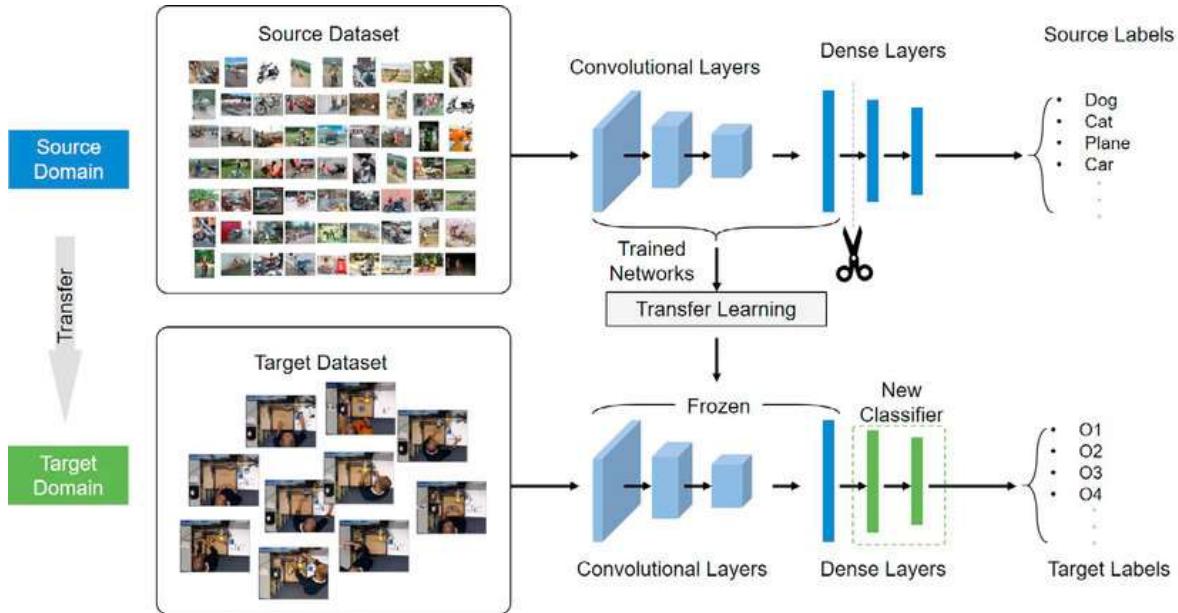
Exercice : On cherche à résoudre le problème suivant avec 1 neurone ayant pour fonction d'activation ReLU :

X	1	2
Y	0	1
F1	2	4
F2	0	8
F3	4	2

On fixe b et tous les poids à 1. On choisit $f = 0$ si $x < 0$ et $f = x$ si $x \geq 0$ et $\lambda=0.01$. Donnez la valeur des poids des entrées du neurone après 1 epoch.

Finetuning / Transfer learning

La dernière couche d'un réseau de neurones (avant la prise de décision) produit des données discriminantes pour la décision ces informations sont appelées « embeddings ». Elles peuvent être utiles pour détecter d'autres concepts que ceux figurant dans la base d'apprentissage. Pour qu'un réseau de neurones puisse détecter un nouveau concept, il suffit parfois de réentraîner uniquement la couche de décision (ou les quelques dernières couches d'un réseau). On parle alors de « transfer learning ». Le fait de ne modifier que les poids des dernières couches s'appelle « fine tuning ».



Overfitting / Drop out

Il est possible qu'un réseau soit surspécialisé sur le corpus d'apprentissage (ie les résultats sont très bons sur les données du corpus) et incapable de produire des résultats satisfaisants en phase d'exploitation. On parle alors de problème d'« overfitting ». Le réseau n'est pas capable de généraliser les concepts appris. Une solution pour limiter cet effet consiste à supprimer quelques neurones pendant la phase d'apprentissage pour obliger les autres neurones à compenser la perte brutale de représentation. Cette méthode est appelée « drop out ».

2. Architecture standard et mise en œuvre

Perceptron

Réseau « à une seule couche » produisant un résultat binaire (= 1 neurone à n entrée et 1 seule sortie avec la fonction de Heaviside en guise de fonction d'activation). Pb = la fonction de Heaviside est nulle partout sauf en 0 => Pb pour la correction des poids.

Perceptron multicouche (Multi Layer Perceptron)

C'est l'architecture classique en deep learning avec :

- Des couches comportant toutes le même nombre de neurones
- Complètement connectées
- Une fonction d'activation non-binaire

3. Choix des paramètres

Lorsqu'on conçoit complètement un réseau de neurones, il existe un nombre important de paramètres relatifs à son architecture (appelés hyperparamètres) pour lesquels il faut faire un « meilleur choix » :

Nombre de couches, nombre de neurones, connexions entre neurones

- Neurone « mort »
C'est un neurone qui ne produit que la valeur 0 en sortie (car toutes les entrées sont à 0 ou négatives ou biais très négatif dans le cas d'une RELU).
- Vanishing/exploding gradient
Un gradient proche de 0 conduira à des corrections de plus en plus faibles en remontant les couches jusqu'à ne pas modifier les premières couches, réduisant ainsi leur intérêt.
Un fort gradient (devant le taux d'apprentissage) va à l'inverse croître en remontant les couches en modifiant conséquemment les valeurs des premières, empêchant ainsi toute convergence

Choix de la fonction d'activation

Choix de la fonction de perte

La fonction de perte « Loss function » est celle qui calcule l'erreur. Il est possible de choisir une fonction spécialisée en fonction du type de résultat recherché :

- Erreur en valeur absolue
- Erreur quadratique
- Entropie croisée : $L(\hat{y}_{l,c}) = - \sum_{c=1}^N y_{i,c} \cdot \ln(\hat{y}_{l,c})$
 - Avec $\hat{y}_{l,c}$ = probabilité prédictive que x_i soit de classe c , et $y_{i,c}$ la vérité terrain sous forme booléenne (voir fonction softmax)
- ...

Choix du taux d'apprentissage

- Statique
- Dynamique

Choix du nombre d'époques

L'apprentissage est généralement reconduit sur la base complète plusieurs fois, éventuellement en changeant l'ordre de présentation des exemples. Chaque itération est appelée une « epoch ». Le nombre d'epochs peut être fixé a priori, ou dépendre du fait que les performances ne sont plus significativement meilleurs lors d'une itération par rapport à la précédente.

Choix de la taille des lots

Apprentissage à l'unité => erreur à évolution « chaotique » => lisser le comportement de l'erreur en ne rétropageant les poids qu'après plusieurs observations sur la base d'un cumul du calcul du gradient (= un batch)

Apprentissage avec l'ensemble des données d'un seul coup => problème de gestion de la mémoire => n'utiliser qu'une partie des données à la fois (= un batch). En général, les batches permettent de converger plus rapidement vers une solution optimale.

Choix de l'optimiseur

Choix de l'algorithme de descente de gradient : en général, plusieurs possibilités :

- Adam
- SGD
- Variantes d'Adam

4. Neurones spécialisés

Neurones de classification

Les neurones en charge d'une classification peuvent générer un vecteur de probabilité grâce à la fonction softmax :

$$\varphi_i(f_1, f_2, \dots) = \frac{e^{f_i}}{\sum_k e^{f_k}}$$

Il y a autant de neurones sur la couche de classification que de classes à détecter. La fonction de loss alors utilisée est généralement la cross entropie. Le gradient alors calculé pour la rétrorépropagation est tout simplement :

$$\varphi_i(f_1, f_2, \dots) - y_i$$

Neurones « Convolutionnels »

Les neurones dits « convolutionnels » visent à identifier des motifs récurrents pour un concept donné. Ils reçoivent entrée directement les intensités des pixels d'une zone rectangulaire ou les résultats d'autres neurones convolutionnels ou de pooling. La fonction d'activation calcule un produit scalaire entre les intensités et un vecteur de poids, d'une manière sensiblement similaire à une convolution.

On trouve autant de neurones sur la couche d'entrée qu'il y a de zones rectangulaires de la taille prédéfinie dans l'image. Les N neurones de la couche d'entrée déterminent les poids les plus adaptés pour la caractérisation d'un motif (= filtre). Tous les neurones partagent les mêmes poids (la mise à jour des poids d'un neurone met à jour les poids de tous les neurones de la même couche). Comme il est souvent nécessaire d'identifier plusieurs motifs caractéristiques, il est nécessaire d'ajouter N neurones pour chaque motif.

En phase d'apprentissage, le calcul est appliqué sur toutes les zones rectangulaires de même taille qui peuvent être trouvées dans l'image, sans connaissance de la position de la région qui illustre le concept. Cela signifie que le neurone est invité à réagir à toutes les positions, y compris celles qui sont négatives. La convergence est donc très lente.

La taille des zones rectangulaires et le nombre de motifs sont des hyperparamètres.

Neurones de pooling

Ils jouent principalement le rôle de traitement multi-échelle en renvoyant une valeur unique à partir des résultats des neurones traitant des zones rectangulaires voisines. Typiquement un neurone d'une couche de « max pooling » renvoie la valeur maximale de ses entrées sans pondération. La rétrorépropagation de l'erreur sur la couche de max pooling est directe (sans calcul de dérivée).

5. Architectures spécialisées

Autoencodeurs

Un autoencodeur est un réseau de neurone dont la taille de la couche d'entrée est la même que celle de sortie et présentant une couche intermédiaire de taille significativement plus réduite (appelée goulot d'étranglement). Un autoencodeur est entraîné pour produire en sortie les mêmes valeurs qu'en entrée.

R-CNN (Region CNN)

CNN classique, mais entraîné sur des régions rectangulaires issues d'un algorithme de segmentation (= sursegmentation, puis fusion, puis approximation du résultat par une fenêtre rectangulaire).

Avantages :

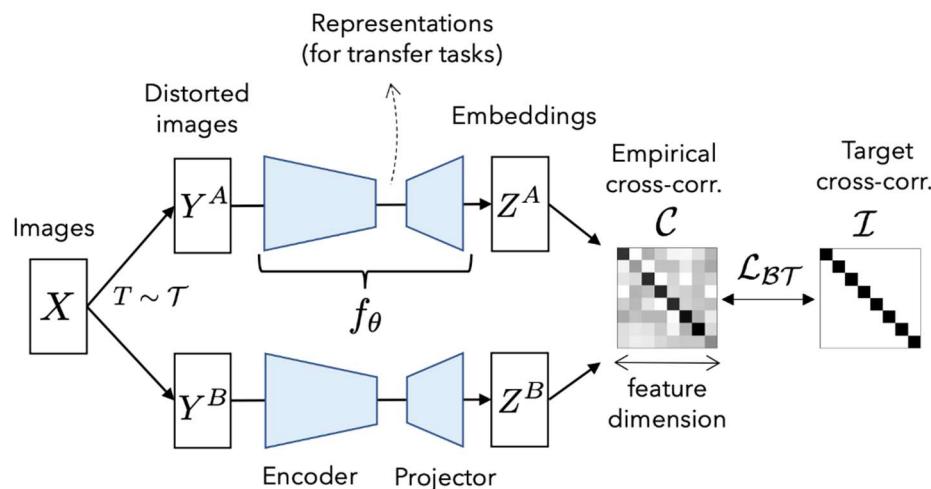
- les zones qui contiennent le moins probablement le concept ne sont pas utilisées dans l'apprentissage.
- couplé à un SVM qui prend en entrée les caractéristiques du CNN et les régions de l'algorithme de segmentation, il est possible de proposer une localisation du concept sous forme de boîte englobante.

(! différent des RNN – réseaux récurrents, plus utilisés pour l'analyse de séquences)

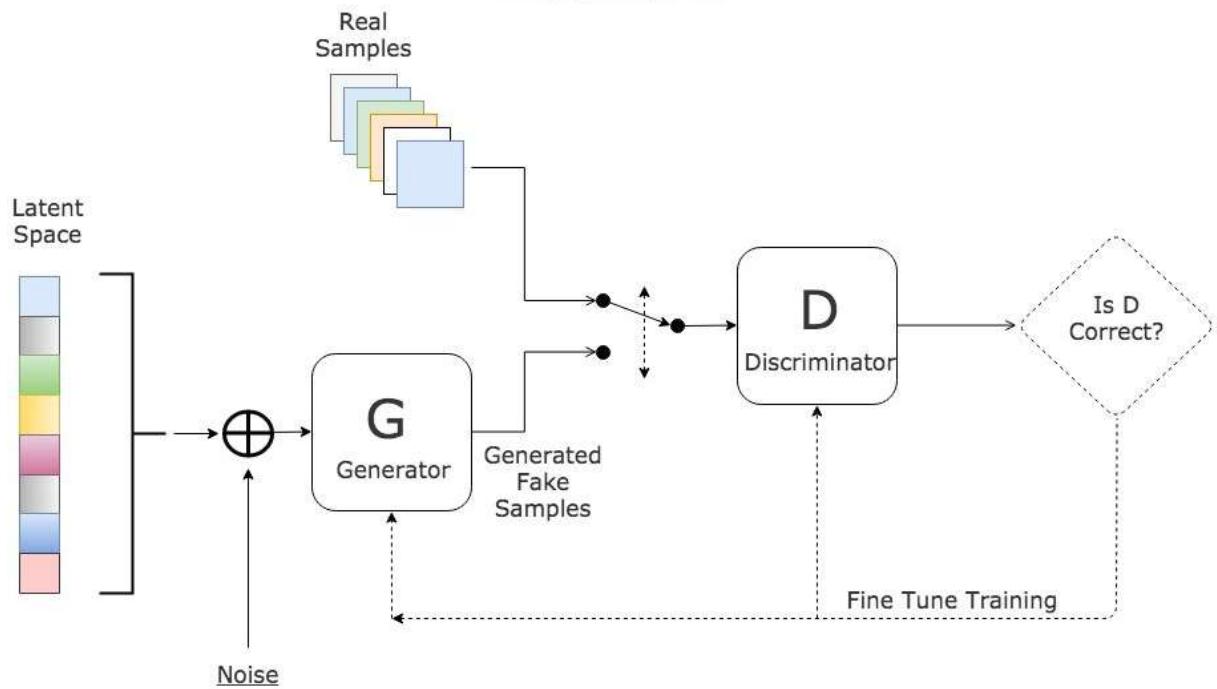
Réseaux siamois

- Autosupervision

Modèle de Barlow-Twins

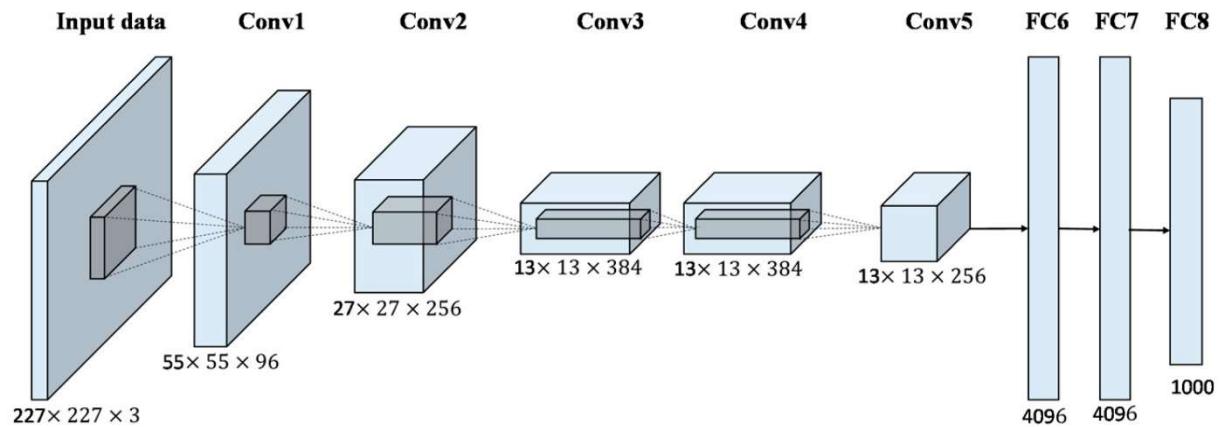


- GAN (Generative adversarial networks)

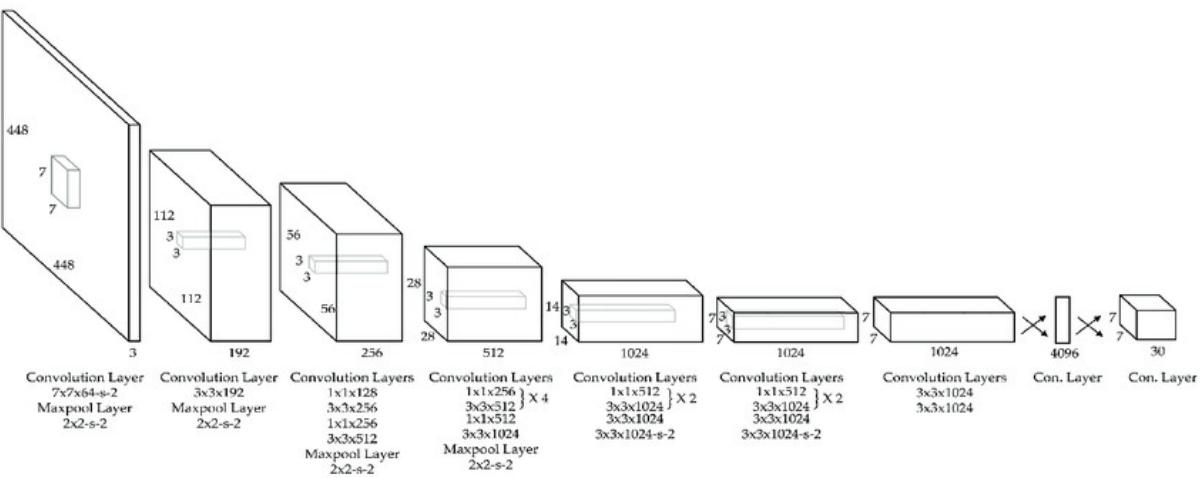


6. Architectures référencées en classification de concept dans les images

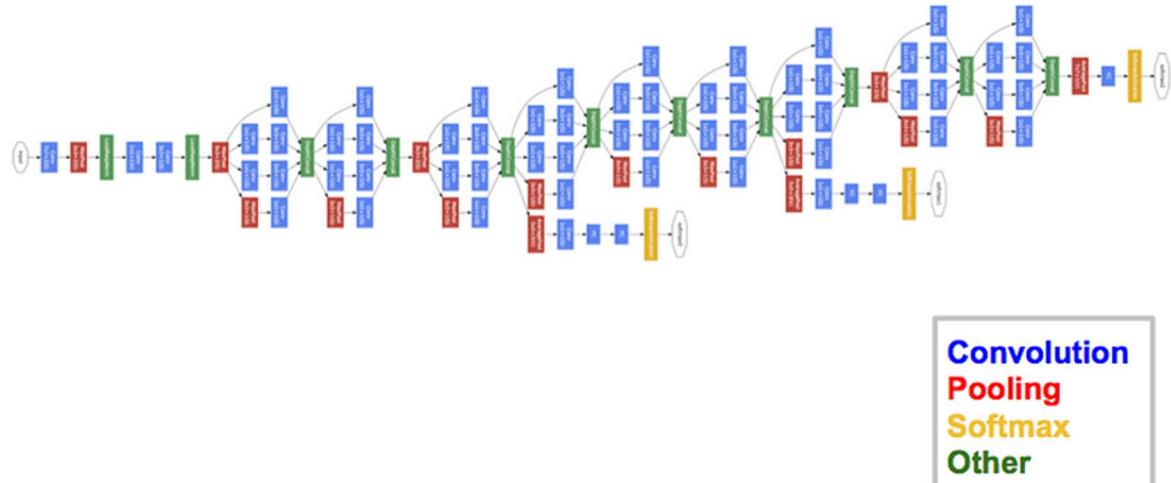
Alexnet



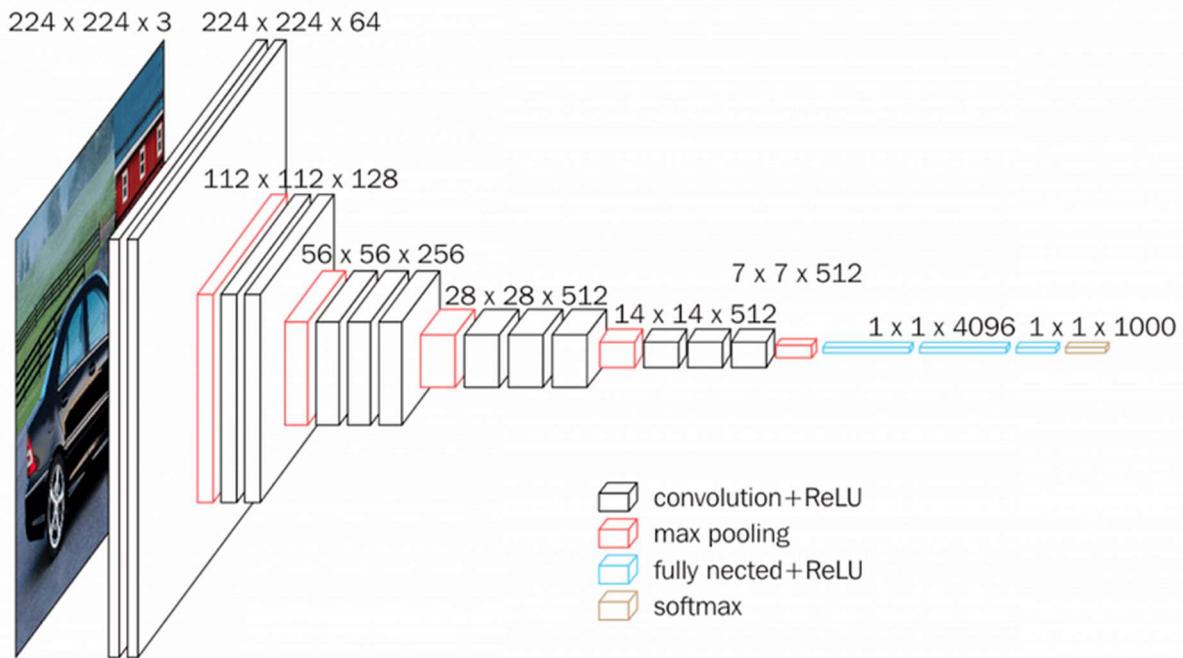
YOLO



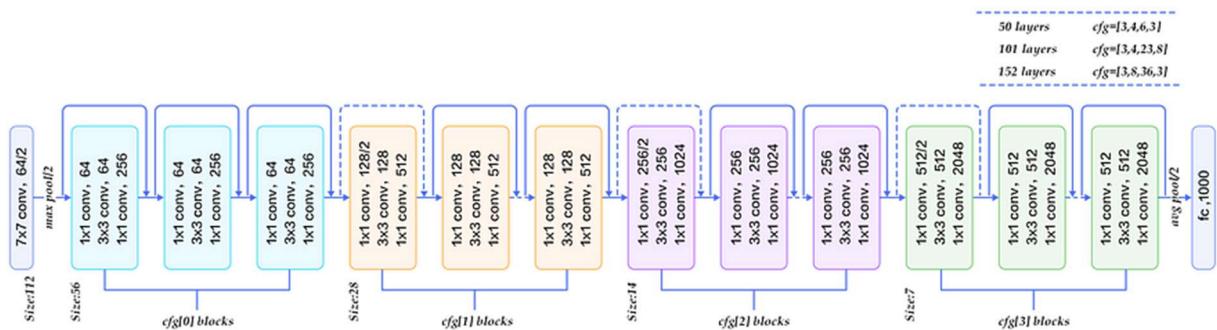
GoogLeNet / Inception



VGG



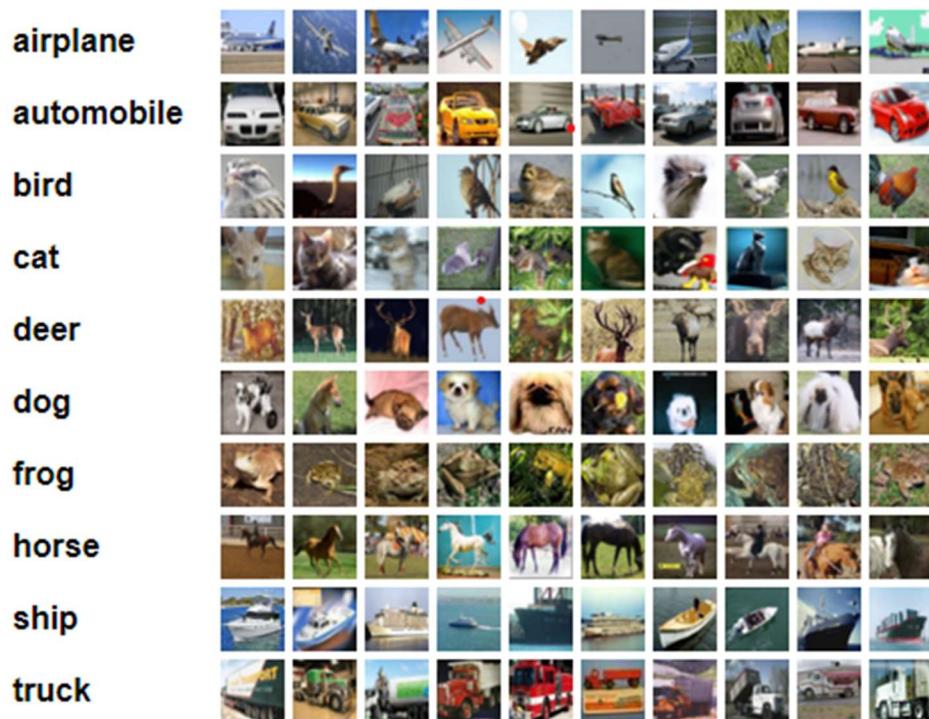
ResNet



7. Bases d'apprentissage

MNIST = 70.000 images de chiffres manuscrits de 28x28 pixels

CIFAR-10 / CIFAR-100 = 60.000 images de 32x32 pixels représentant 10/100 classes équilibrées différentes



ImageNet = 1.431.167 images de resolution variable (moyenne = 469x387) représentant 1000 classes

COCO (Common Objects in COntext) => 200.000 images labellisées avec 172 classes

8. Réseaux de Neurones à Spike Input spike trains (from presynaptic neurons)

