IFOSUP

Institut de Formation Supérieure
Ville de Wavre

Mathématiques

appliquées à l'informatique

G. Barmarin

2023-2024

IFOSUP

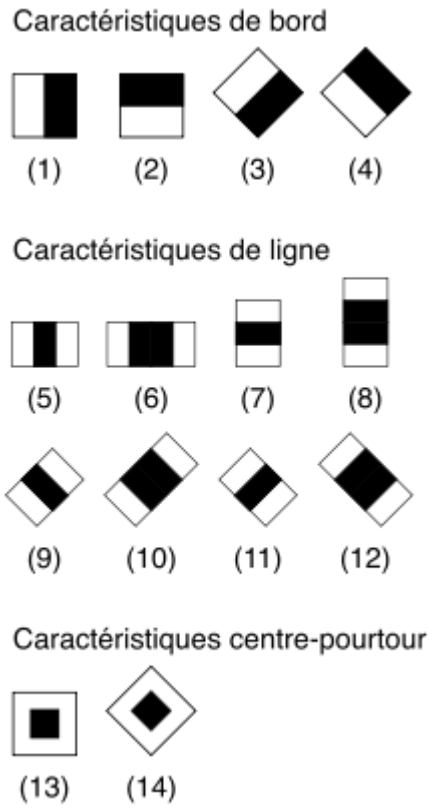
Institut de Formation Supérieure
Ville de Wavre

Reconnaissance de visages(*)

Avec OpenCV et les classificateurs de Haar

(*) ou de chats

1 Les classificateurs de Haar



Alfred Haar
(1885-1933)

Cascades de Haar

Les cascades de Haar, également appelées classificateurs de Viola Jones, sont une méthode de détection d'objets périmés proposée par Paul Viola et Michael Jones en 2001.

Il s'agit d'une approche basée sur l'apprentissage automatique dans laquelle une cascade est entraînée à partir d'un grand nombre d'images positives et négatives.

Elle est ensuite utilisée pour détecter des objets dans d'autres images.

Prenons l'exemple de la détection des visages.

Si nous devions utiliser les cascades de haar, nous devrions d'abord entraîner l'algorithme avec un grand nombre d'images positives (images de visages) et d'images négatives (images sans visages). Après l'entraînement, nous aurions un détecteur Haar entraîné qui serait capable de détecter les visages. Ce détecteur recherche les caractéristiques Haar et parcourt l'image en utilisant une fenêtre coulissante.

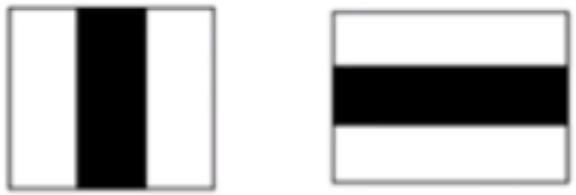
Ces caractéristiques de Haar sont une valeur unique obtenue en soustrayant la somme des pixels situés sous le rectangle blanc de la somme des pixels situés sous le rectangle noir.

→ it was proposed by **Alfréd Haar** in **1909**

→ they are like convolutional kernels

HAAR FEATURES ARE THE RELEVANT FEATURES FOR FACE DETECTION

edge features can detect edges quite effectively



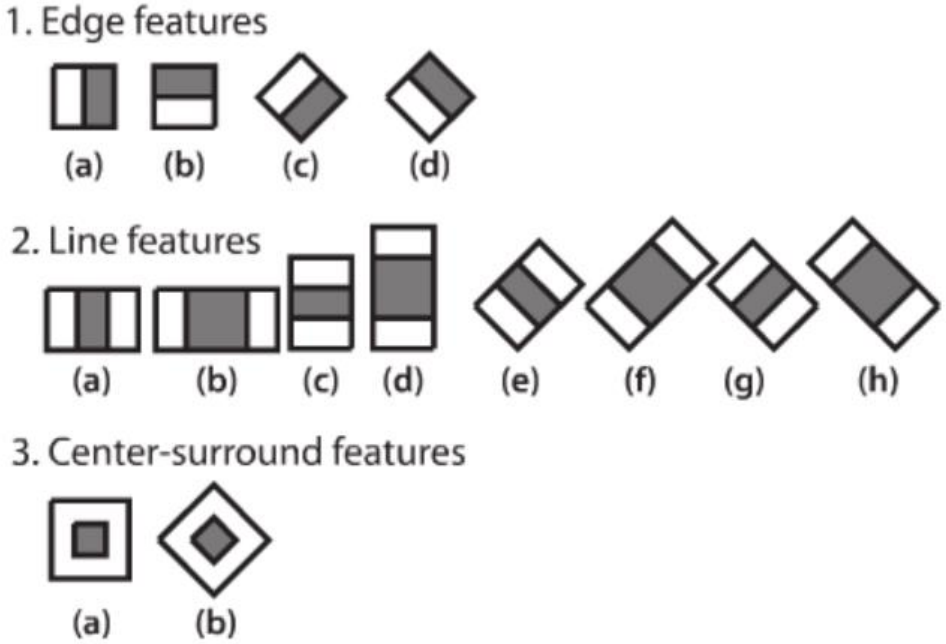
line features can detect lines quite effectively

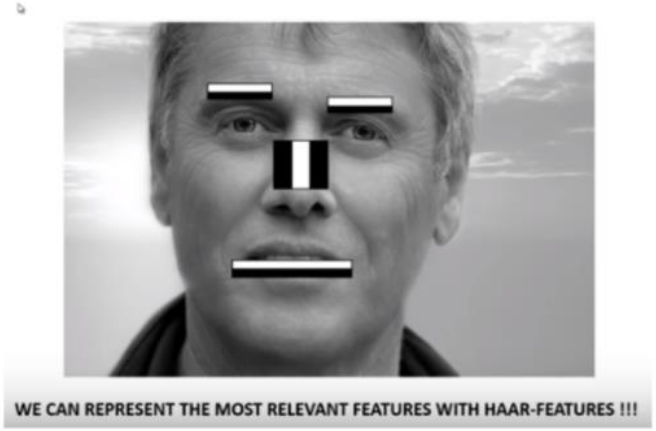
Comme vous pouvez le constater, il existe plusieurs types de caractéristiques. Certains recherchent les bords horizontaux/verticaux, d'autres les lignes, etc.

L'algorithme d'OpenCV utilise actuellement les caractéristiques de type Haar suivantes, qui constituent l'entrée des classificateurs de base

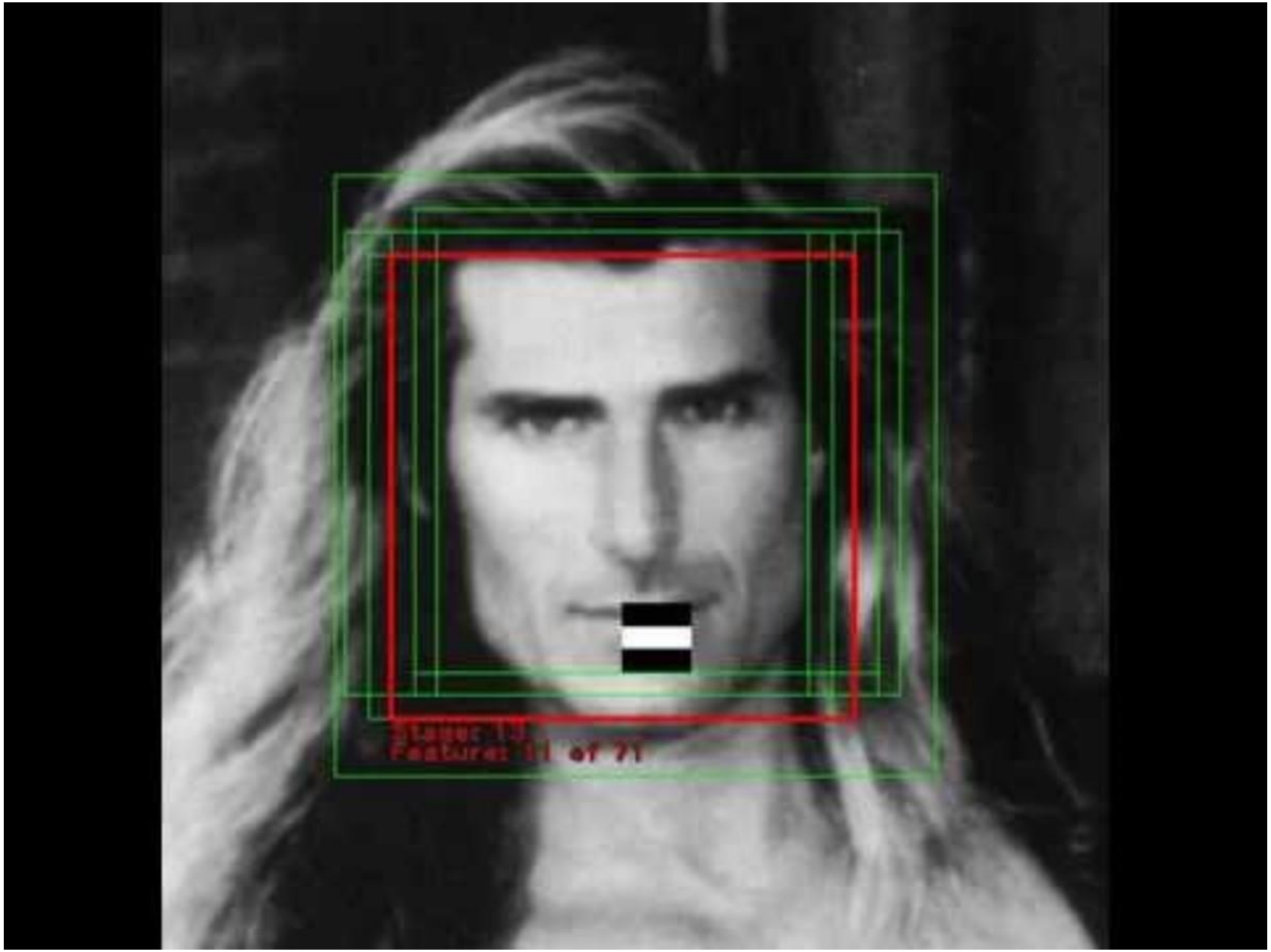
Vous pouvez voir ci-contre comment ces caractéristiques sont capables de détecter des parties du visage, par exemple un sourcil qui est comme un bord blanc-noir-blanc ou simplement une caractéristique blanc-noir, ou un nez qui est une caractéristique verticale noir-blanc-noir, ce qui signifie qu'un nez est généralement plus blanc que les côtés environnants, nous pouvons donc construire une caractéristique qui peut rechercher le nez en le décrivant comme étant blanc et noir de chaque côté

Bien sûr les caractéristiques Haar ne recherchent pas exactement des zones noires ou blanches mais des zones blanchâtres ou noirâtres.



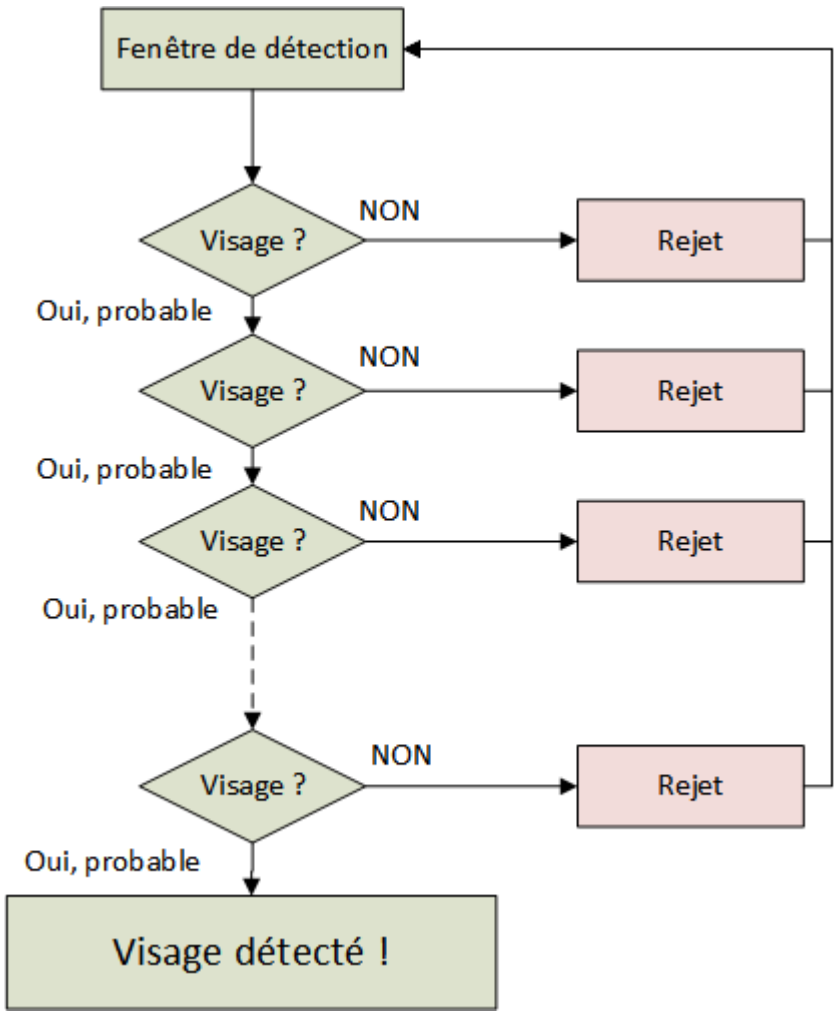


Les cascades de Haar à l’oeuvre



Steps Of Haar Cascades:

- The Cascade has a total of 32 layers and 6000 + feature classifiers
- 1st layer contains 2 feature classifiers
- 2nd layer has 5 features classifiers
- Next 3 layers have 20 feature classifiers
- Next 2 layers have 50 feature classifiers
- Next 5 layers have 100 feature classifiers
- Next 20 layers have 200 feature classifiers



Au lieu d'appliquer l'ensemble des 6 000 caractéristiques à une fenêtre, on regroupe les caractéristiques en différentes étapes de classificateurs et on les applique une par une.

Normalement, les premières étapes contiennent un très petit nombre de caractéristiques). Si une fenêtre échoue à la première étape, elle est rejetée.

On ne prend pas en compte les caractéristiques restantes.

Si une étape est positive, on applique la deuxième étape de caractéristiques et on continue le processus.

La fenêtre qui passe toutes les étapes est une région de visage

Étapes concrètes du programme

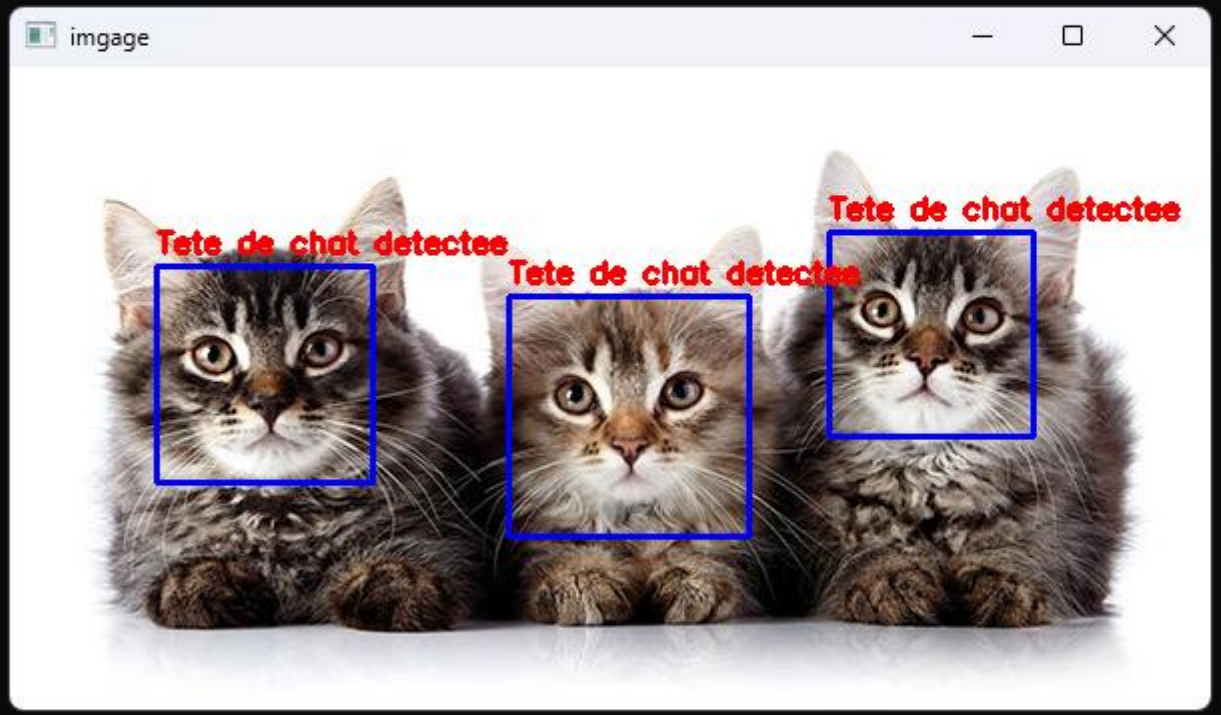
Pour détecter des visages de chats dans une image et dessiner des boîtes de délimitation autour d'eux, vous pouvez suivre les étapes ci-dessous

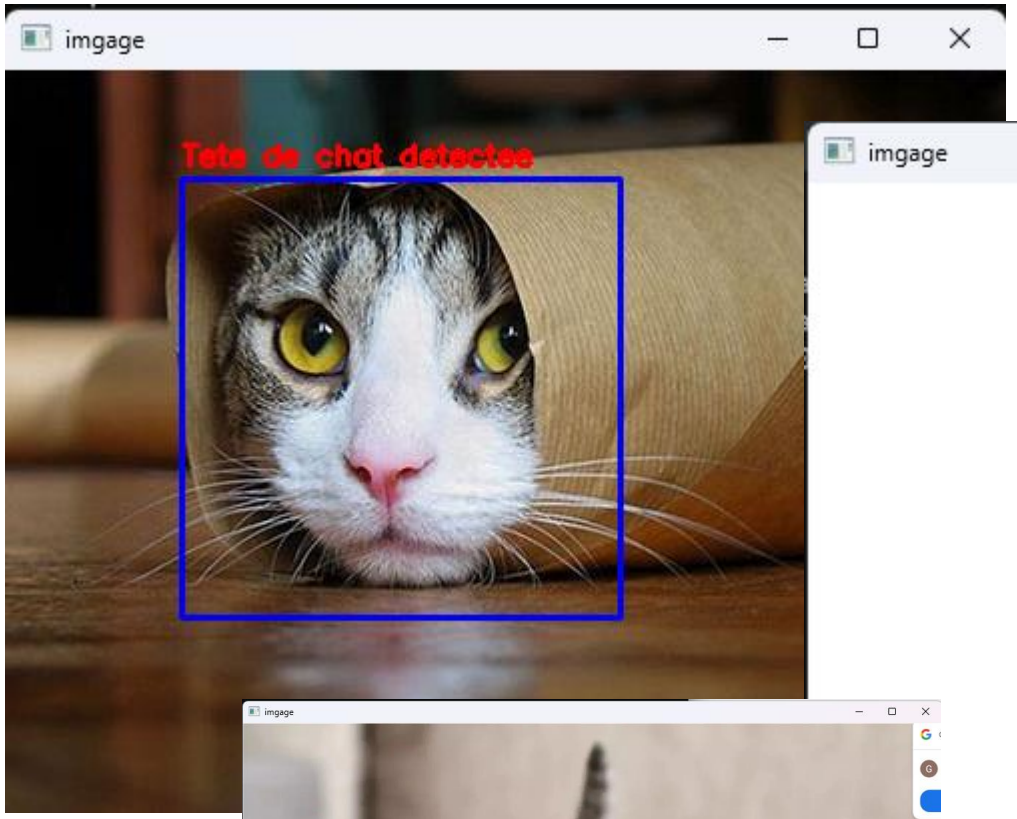
- Importez la bibliothèque OpenCV ou ssurez-vous de l'avoir déjà installée.
- Lisez l'image d'entrée en utilisant `cv2.imread()`. Spécifiez le chemin complet de l'image.
- Convertir l'image d'entrée en niveaux de gris:
  `image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)`
- Lancer un objet classificateur en cascade Haar: `cat_cascade = cv2.CascadeClassifier()` pour la détection des visages de chats. Transmettre le chemin complet du fichier xml de la cascade de Haar. Vous pouvez utiliser le fichier haar cascade `haarcascade_frontalcatface.xml` pour détecter les visages de chats dans l'image.
- Détectez les visages de chat dans l'image d'entrée à l'aide de `cat_cascade.detectMultiScale()`. Cette fonction renvoie les coordonnées des visages de chat détectés au format (x,y,w,h).
- Dessinez les rectangles de délimitation autour des visages de chats détectés dans l'image originale à l'aide de `cv2.rectangle()`.
- Affichez l'image avec les rectangles de délimitation et un titre dessiné autour des visages de chat: `cv2.imshow()`

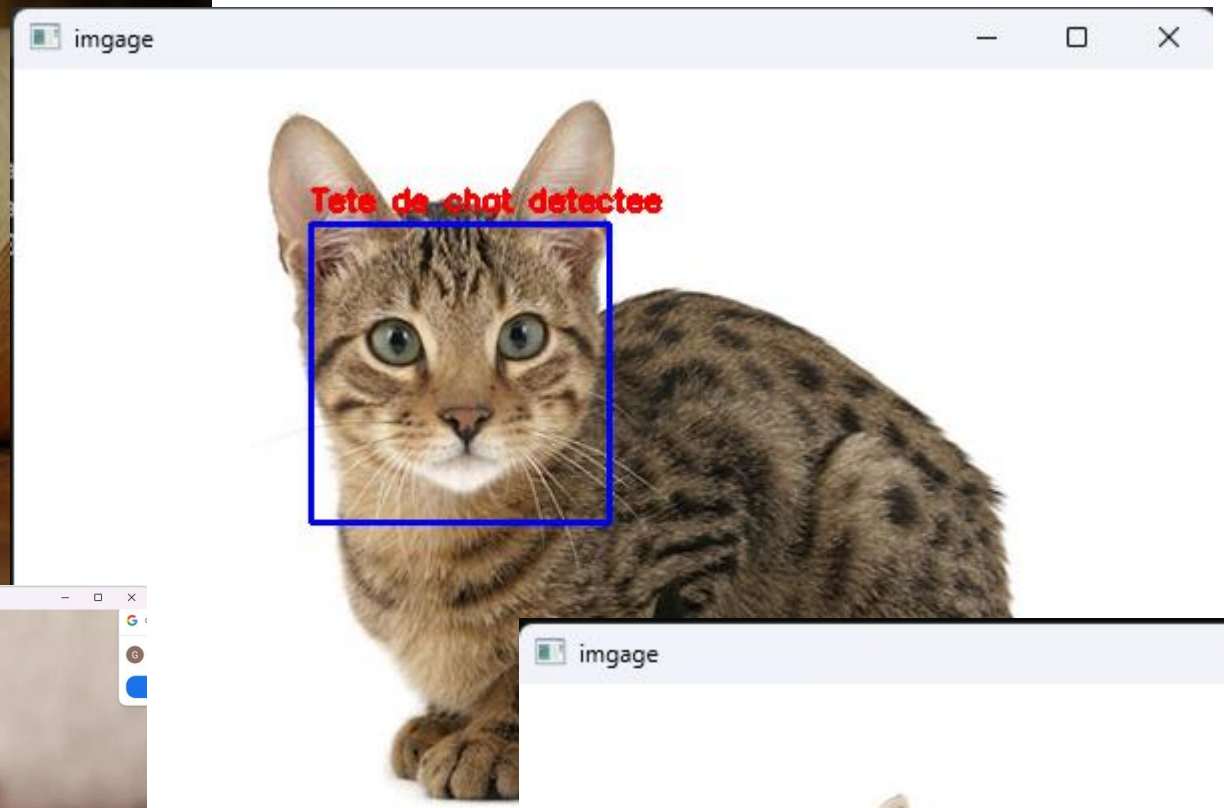
Fichier sur Teams: detection-chat.py

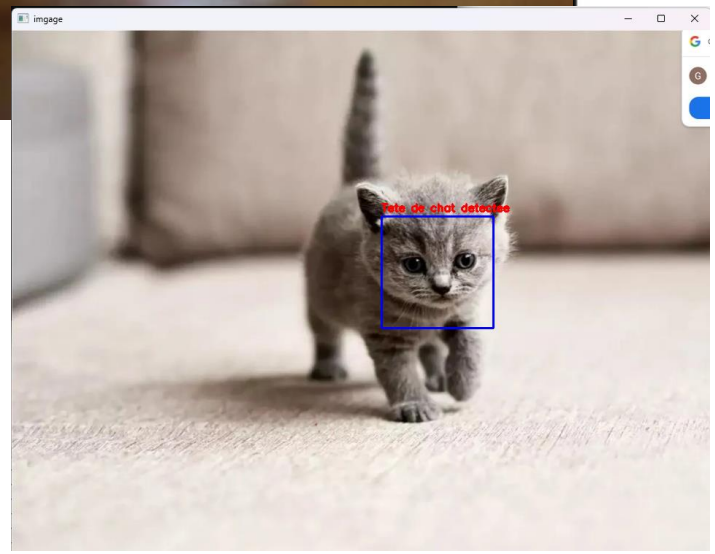
Résultat:

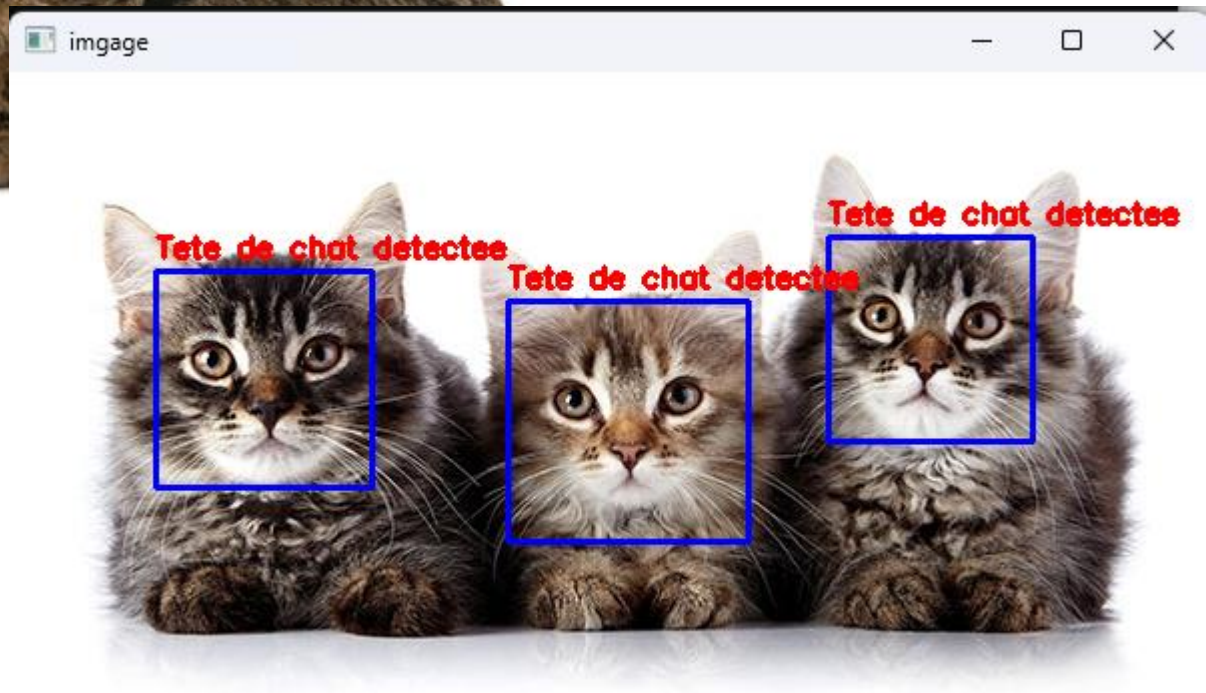
```
C:\WINDOWS\SYSTEM32\cmd
Dimensions de l'image cat_04.jpg :  (321, 600, 3)

C:\Users\gerar\AppData\Local\Programs\Python\Python311\Lib\site-packages\cv2\data\
C:\Users\gerar\AppData\Local\Programs\Python\Python311\Lib\site-packages\cv2\data\haarcascade_frontalcatface.xml
< cv2.CascadeClassifier 000002D97D52BB50>
Le fichier est chargé.

3 visages detectés dans l'image.

Coordonnées des faces détectées:

[[409  82 102 102]
 [ 73  99 108 108]
 [249 114 120 120]]

C'est tout!
```




imgage
Tete de chat detectee

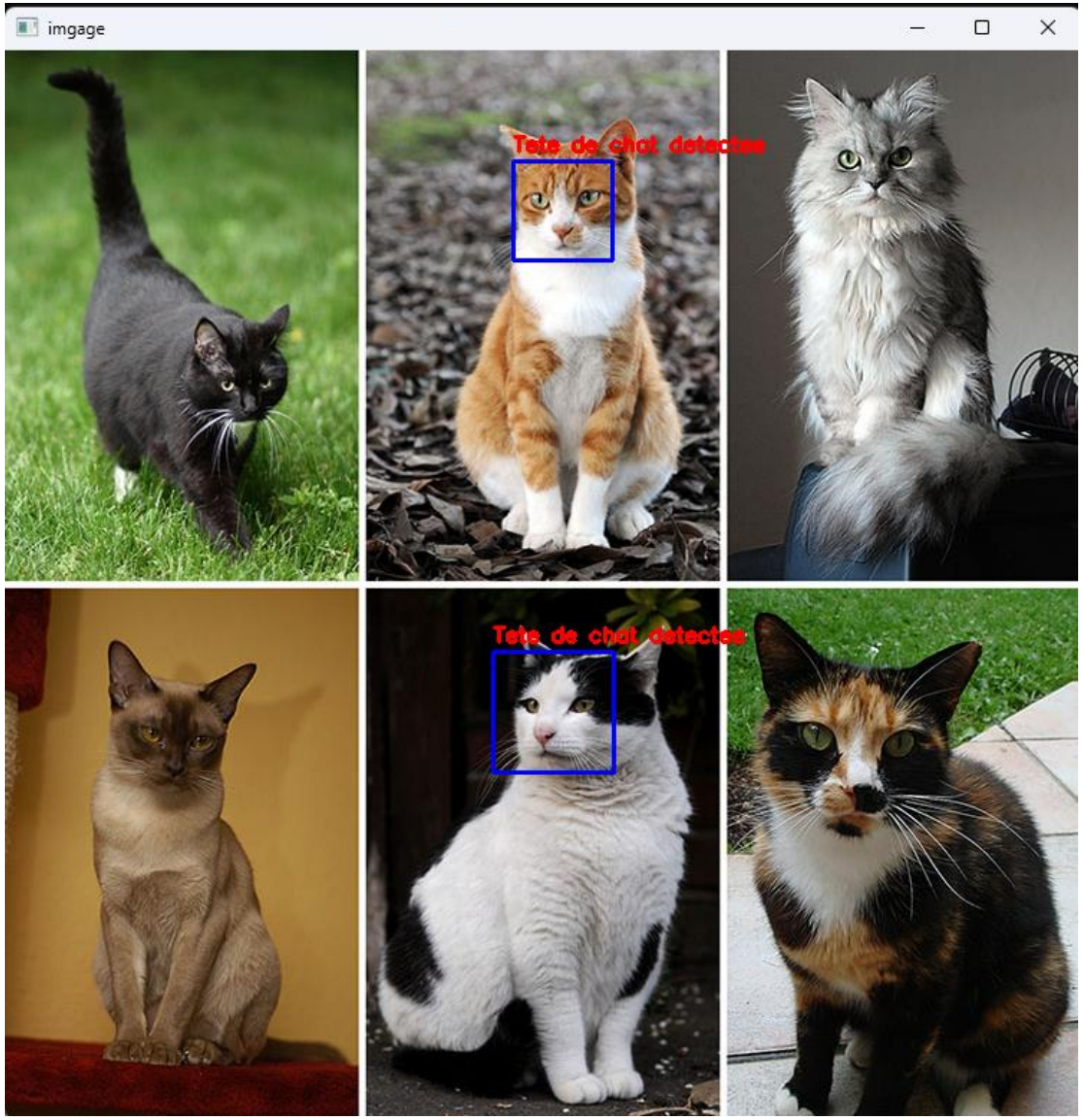

imgage
Tete de chat detectee


imgage
Tete de chat detectee


imgage
Tete de chat detectee
Tete de chat detectee
Tete de chat detectee

Mais...

Seulement 2 chats détectés...



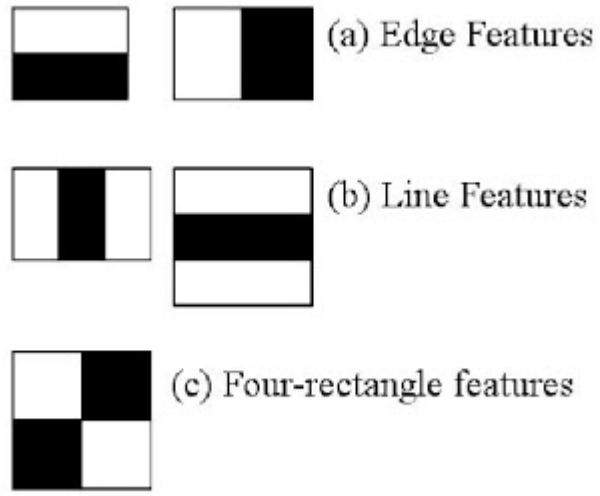
Et pas de détection

Créer votre fichier de détection:
https://www.framboise314.fr/i-a-creez-votre-propre-modele-de-reconnaissance-dobjets-1er-partie/

2 Les réseaux de neurones

Théorie

La détection d'objets à l'aide de classificateurs en cascade basés sur les caractéristiques Haar est une méthode efficace de détection d'objets proposée par Paul Viola et Michael Jones dans leur article intitulé "Rapid Object Detection using a Boosted Cascade of Simple Features" (Détection rapide d'objets à l'aide d'une cascade renforcée de caractéristiques simples) en 2001. Il s'agit d'une approche basée sur l'apprentissage automatique dans laquelle une fonction en cascade est entraînée à partir d'un grand nombre d'images positives et négatives. Elle est ensuite utilisée pour détecter des objets dans d'autres images.

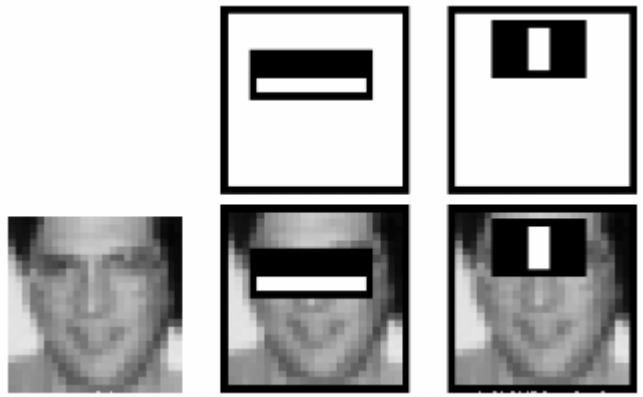


Nous travaillerons ici sur la détection des visages. Au départ, l'algorithme a besoin d'un grand nombre d'images positives (images de visages) et d'images négatives (images sans visages) pour entraîner le classificateur. Nous devons ensuite en extraire les caractéristiques. Pour ce faire, nous utilisons les caractéristiques Haar présentées dans l'image ci-dessous. Elles sont comparables à notre noyau convolutif. Chaque caractéristique est une valeur unique obtenue en soustrayant la somme des pixels situés sous le rectangle blanc de la somme des pixels situés sous le rectangle noir.

Maintenant, toutes les tailles et tous les emplacements possibles de chaque noyau sont utilisés pour calculer un grand nombre de caractéristiques. (Imaginez la quantité de calculs nécessaires ? Même une fenêtre 24x24 produit plus de 160000 caractéristiques). Pour chaque calcul de caractéristique, nous devons trouver la somme des pixels situés sous les rectangles blancs et noirs. Pour résoudre ce problème, ils ont introduit l'image intégrale. Quelle que soit la taille de votre image, elle réduit les calculs pour un pixel donné à une opération impliquant seulement quatre pixels. C'est bien, non ? Cela rend les choses très rapides.

Mais parmi toutes ces caractéristiques que nous avons calculées, la plupart ne sont pas pertinentes. Prenons l'exemple de l'image ci-dessous. La rangée du haut montre deux bonnes caractéristiques. La première caractéristique sélectionnée semble se concentrer sur la propriété selon laquelle la région des yeux est souvent plus foncée que la région du nez et des joues. La deuxième caractéristique sélectionnée s'appuie sur la propriété selon laquelle les yeux sont plus foncés que l'arête du nez. Mais les mêmes fenêtres appliquées aux joues ou à tout autre endroit ne sont pas pertinentes. Alors, comment sélectionner les meilleures caractéristiques parmi plus de 160000 ? C'est possible grâce à Adaboost.Traduit avec DeepL.com (version gratuite)

Pour ce faire, nous appliquons chaque caractéristique à toutes les images d'apprentissage. Pour chaque caractéristique, nous trouvons le meilleur seuil qui permettra de classer les visages comme positifs ou négatifs. Il est évident qu'il y aura des erreurs ou des mauvaises classifications. Nous sélectionnons les caractéristiques ayant un taux d'erreur minimal, ce qui signifie qu'il s'agit des caractéristiques qui classent le plus précisément les images de visages et de non-visages. (Le processus n'est pas aussi simple que cela. Au départ, chaque image se voit attribuer un poids égal. Après chaque classification, les poids des images mal classées sont augmentés. Le même processus est ensuite répété. De nouveaux taux d'erreur sont calculés. De nouveaux poids sont également calculés. Le processus est poursuivi jusqu'à ce que la précision ou le taux d'erreur requis soit atteint ou que le nombre requis de caractéristiques soit trouvé.)Le classificateur final est une somme pondérée de ces classificateurs faibles. Il est qualifié de faible parce qu'il ne peut pas, à lui seul, classer l'image, mais, combiné à d'autres, il forme un classificateur fort. L'article indique que même 200 caractéristiques permettent une détection avec une précision de 95 %. Leur configuration finale comportait environ 6 000 caractéristiques. (Imaginez une réduction de 160000+ caractéristiques à 6000 caractéristiques. C'est un gain considérable).Prenons maintenant une image. Prenez chaque fenêtre 24x24. Appliquez-y 6000 caractéristiques. Vérifiez s'il s'agit d'un visage ou non. N'est-ce pas un peu inefficace et fastidieux ? Oui, c'est vrai. Les auteurs ont trouvé une bonne solution.Traduit avec DeepL.com (version gratuite)



Dans une image, la majeure partie de l'image n'est pas une zone de visage. Il est donc préférable de disposer d'une méthode simple pour vérifier si une fenêtre n'est pas une zone de visage. Si ce n'est pas le cas, éliminez-la en une seule fois et ne la traitez plus. Au lieu de cela, nous nous concentrons sur les régions où il peut y avoir un visage. De cette manière, nous passons plus de temps à vérifier les régions où il peut y avoir un visage.Pour ce faire, ils ont introduit le concept de cascade de classificateurs. Au lieu d'appliquer l'ensemble des 6 000 caractéristiques à une fenêtre, les caractéristiques sont regroupées en différentes étapes de classificateurs et appliquées une par une. (Normalement, les premières étapes contiennent beaucoup moins de caractéristiques). Si une fenêtre échoue à la première étape, elle est rejetée. Nous ne prenons pas en compte les caractéristiques restantes. Si elle réussit, nous appliquons la deuxième étape de caractéristiques et continuons le processus. La fenêtre qui passe toutes les étapes est une région faciale. Quel plan !Le détecteur des auteurs comportait plus de 6 000 caractéristiques et 38 étapes avec 1, 10, 25, 25 et 50 caractéristiques dans les cinq premières étapes. (Les deux caractéristiques de l'image ci-dessus sont en fait les deux meilleures caractéristiques obtenues par Adaboost). Selon les auteurs, en moyenne, 10 caractéristiques sur plus de 6 000 sont évaluées par sous-fenêtre.Il s'agit donc d'une explication intuitive simple du fonctionnement de la détection de visages de Viola-Jones. Lisez l'article pour plus de détails ou consultez les références dans la section Ressources supplémentaires.

L'exemple de code suivant utilise des modèles de cascade de Haar pré-entraînés pour détecter les visages et les yeux dans une image.

Ressources

Paul Viola and Michael J. Jones. Robust real-time face detection. International Journal of Computer Vision, 57(2):137–154, 2004. **[227]**

Rainer Lienhart and Jochen Maydt. An extended set of haar-like features for rapid object detection. In Image Processing. 2002. Proceedings. 2002 International Conference on, volume 1, pages I–900. IEEE, 2002. **[135]**

Video : Face Detection and Tracking

Video Vimeo de Adam Harvey : OpenCV Face Detection: Visualized

An interesting interview regarding Face Detection by Adam Harvey