

Master spécialisés : Ingénierie de Données et Développement Logiciel  
Année universitaire 2019- 2020

**Module : Machine Learning**

**Rapport de projet de fin de module :**

# **Reconnaissance faciale avec OpenCV et Python**

**Réalisé par :**

**El Msayer Mohamed  
Wadie Hamdani**

**Encadre par :**

**M. Abdelhak Mahmoudi**

# Table des matières

## Introduction générale

## Chapitre I : A propos du projet

### I. Reconnaissance faciale

1. Introduction
2. Comment fonctionne la technologie ?
3. Le rôle de l'intelligence artificielle dans la technologie de reconnaissance faciale

## Chapitre II : Outils utilisés et réalisation du projet

### I. Environnement de travail

1. Langage de programmation
2. Outils utilisés
3. Bibliothèques utilisées
  - 3.1. NumPy
  - 3.2. OpenCV
  - 3.3. Pillow python
4. Modèles utilisés
  - 4.1. Cascade Classifier
  - 4.2. Techniques de reconnaissance de visages :
    - 4.2.1. Réseaux de neurones artificiels :

### II. Réalisation du projet :

1. Phase d'apprentissage
2. Phase de training
3. Phase de test

## Conclusion Générale et perspectives

## Webographie

## Introduction générale

Un système de reconnaissance faciale est une application logicielle visant à reconnaître une personne grâce à son visage de manière automatique. C'est un domaine de la vision par ordinateur consistant à reconnaître automatiquement une personne à partir d'une image de son visage. Il s'agit d'un sujet particulièrement étudié en vision par ordinateur. La reconnaissance de visage a de nombreuses applications en vidéosurveillance, biométrie, robotique, indexation d'images et de vidéos, recherche d'images par le contenu, etc.

Ces systèmes sont généralement utilisés à des fins de sécurité pour déverrouiller ordinateur/mobile/console, mais aussi en domotique. Ils sont appréciés car considérés comme peu invasifs, en comparaison avec les autres systèmes biométriques (empreintes digitales, reconnaissance de l'iris...). Le fonctionnement de ces systèmes se base sur une ou plusieurs caméras pour reconnaître l'utilisateur.

Ils peuvent également être utilisés afin de faciliter la vie de l'utilisateur, comme le font par exemple certains réseaux sociaux sur internet (Facebook, Google) ou certaines applications mobiles (NameTag, FaceRec) pour identifier des visages sur des images. Ces systèmes se basent alors sur des photos/vidéos d'une ou plusieurs personnes.

La reconnaissance faciale, sous-catégorie de l'Intelligence Artificielle, représente un ensemble de méthodes de détection et d'analyse d'images pour permettre l'automatisation d'une tâche spécifique. Il s'agit d'une technologie qui est capable d'identifier des personnes, au sein d'une image et d'en tirer des conclusions en l'analysant.

La reconnaissance faciale peut être réalisée à des degrés de précision différents, par rapport au type d'information ou de concept recherché. En effet, un modèle ou algorithme est capable de détecter un visage spécifique, tout comme il peut simplement attribuer une image à une grande catégorie.

En vue de rendre la structure de ce travail plus compréhensive et claire il fallait présenter son articulation.

D'abord, le 1er chapitre présente une vue plus détaillée sur le principe de reconnaissance d'image et le fonctionnement cette technologie. Le but de ce chapitre est de tracer le chemin vers les deux chapitres suivants.

Ensuite, le 2ème chapitre vise à présenter la réalisation de notre projet, l'environnement de travail et l'analyse du code et algorithmes.

Ce rapport s'achève sur une conclusion générale dans laquelle on a fait une comparaison des résultats obtenus.

# Chapitre I :

A propos du projet  
Reconnaissance Faciale

Dans ce chapitre nous allons présenter une vue plus détaillée sur le principe de reconnaissance d'image et le fonctionnement de cette technique. Le but de ce chapitre est de tracer le chemin vers les deux chapitres suivants.

## **I. Reconnaissance faciale**

### **4. Introduction**

La reconnaissance faciale, objet du présent rapport, est un procédé informatisé biométrique qui compare, à partir des traits du visage, deux ou plusieurs images prises à des lieux et/ou des moments différents. Cette technique permet notamment de satisfaire plusieurs objectifs :

- **Authentifier une personne**

Par ce biais consiste à comparer une de ses caractéristiques physiques (en l'occurrence son visage) avec celle enregistrée préalablement et censée la caractériser.

L'authentification permet de répondre à la question « Est-ce bien Monsieur X ? » et donc de confirmer, par exemple, que le porteur d'un titre d'accès est bien celui qu'il prétend être. On parle de système « 1 pour 1 ».

- **Identifier une personne**

Nécessite de comparer la même caractéristique avec une base de données de personnes connues. L'identification permet de répondre à la question « Qui est cette personne ? » et donc de confirmer, par exemple, que le porteur d'un titre d'accès est bien légitime à accéder au site, puisqu'il figure dans la base de données des « autorisés ». On parle de système « 1 pour N ».

- **Ré-identifier une personne**

visé à retrouver quelqu'un préalablement observé. La ré-identification permet de répondre à la question « Ai-je déjà vu cette personne ? » et de confirmer, par exemple, qu'elle a cheminé par différents points du site. Cette action est engagée à la suite d'une première identification formelle (1 pour N) ou à partir de caractéristiques non morphologiques (par ex. un sac ou un vêtement d'une couleur donnée).

- **Rechercher une personne**

Suppose une action de surveillance permanente d'une foule de personnes inconnues, afin de vérifier si y figure une personne dont on dispose des caractéristiques biométriques de son visage. La recherche permet de répondre à la question « Est-ce que Monsieur X est présent ? ». On parle de système « N pour M ».

Ces grandes catégories d'exploitation d'un dispositif de reconnaissance faciale sont mises en œuvre selon les cas sur des points de passage obligé ou en surveillance plus générale de zone, en milieu public ou privé, en intérieur ou extérieur, en temps réel ou différé et sur une période de temps limitée (à la suite d'un événement) ou en continu.

D'autres technologies, non étudiées dans le présent rapport mais pourtant souvent associées à la reconnaissance faciale, utilisent également la comparaison d'images à des fins de sécurité. Il s'agit principalement de dispositifs permettant de :

- + Rechercher des objets abandonnés (colis suspect)
- + Identifier des objets (recherche d'un objet dans des banques vidéo)
- + Analyser des changements de scènes (sécurisation de trajets ou de lieux)
- + Détecter des intrusions (définition de zone interdites, « vidéo sensor »)
- + Lire automatiquement des plaques d'immatriculation (LAPI)
- + Analyser des comportements (contre sens de circulation, mouvements de foules, ..)
- + Analyser des expressions ou des émotions exprimées par les visages.

## 5. Comment fonctionne la technologie ?

Tous les dispositifs de reconnaissance faciale reposent sur une action de comparaison avec un attribut présenté par une base de données ou un support physique détenu par la personne, et sont généralement construits autour de 5 modules :

### ▪ Un système d'acquisition :

Collecte l'échantillon d'une image à partir d'un dispositif matériel (caméra, appareil photo, ordinateur ...)

### ▪ Un procédé de traitement :

D'image convertit les données d'échantillon en canevas biométriques uniques, propres à chaque personne. Ce traitement peut être simple si l'image est prise de face par un appareil de bonne qualité, ou bien très complexe, s'il faut isoler une personne dans une foule, corriger les données d'environnement (éclairage ...) et « redresser » la photographie, ou bien « vieillir » une image ancienne, ou encore dé-pixelliser une image trop zoomée...

### ▪ Une unité de stockage

Des données caractéristiques biométriques sur un support informatique (base de données, puce électronique sur un passeport, code barre sur une carte d'enregistrement de vol, ...). Un mécanisme de codage / chiffrement peut être appliqué avant le stockage, rendant les données non lisibles. D'autres données numériques (identité de la personne, données administratives, métadonnées) peuvent être associées à ce stockage.

- **Un dispositif de traitement**

Par comparaison entre les canevas biométriques, recueillis à des endroits et des moments différents, selon le processus retenu Ce dispositif donne lieu à un score de correspondance (exprimé en pourcentage de certitude).

- **Un système de décision**

S'appuie sur les résultats de la comparaison et des critères de décision (seuil par exemple). Le résultat est de type « Oui/Non » pour une vérification et une liste de candidats possibles dans le cadre d'une identification par score.

**Le processus d'enrôlement** concerne les 3 premières phases précitées et permet ainsi d'enregistrer les caractéristiques biométriques faciales d'une personne ainsi que ses données d'identification.

Actuellement mises en œuvre sur des systèmes dédiés, les implémentations de reconnaissance faciale pourraient se déployer à l'avenir sur des environnements plus distribués et mutualisés, notamment dans le Cloud dans un souci de réduction de coût et d'amélioration des performances.

## 6. Le rôle de l'intelligence artificielle dans la technologie de reconnaissance faciale :

La reconnaissance faciale évolue considérablement depuis quelques années, portée par les innovations technologiques. La comparaison de deux images ne repose plus désormais sur leurs seuls points caractéristiques, mais sur la confrontation de tous les éléments de l'image (pixels) à partir d'algorithmes bâtis sur des réseaux de neurones artificiels. Cette technologie présente l'avantage de l'auto-apprentissage : la machine s'entraîne à détecter des visages en examinant des milliers d'images de visage et les résultats (succès ou échec) des comparaisons.

- **Des algorithmes perfectibles**

Les algorithmes étant codés ou entraînés sur des bases prescrites par des humains, ils sont intrinsèquement marqués par des biais de leurs producteurs. Jean-Michel Loubes, statisticien à l'université de Toulouse, considère qu'il faut donc s'interroger sur le fonctionnement de l'intelligence artificielle et la contrôler. La performance de la reconnaissance faciale nécessiterait une auscultation des entrailles d'algorithmes. Une série de test serait donc indispensable à mettre en œuvre pour vérifier les réponses d'algorithme, et s'assurer que les biais sont gérés.

Une stratégie pourrait consister à introduire des exemples « tordus » dans la phase d'apprentissage, pour entraîner le système à se méfier.

Afin de démontrer les vulnérabilités des algorithmes de reconnaissance d'image, Nicolas Papernot, chercheur scientifique, et ses collègues ont cherché à altérer des images de panneaux de signalisation pour transformer des « stops » en « céder le passage », sans que ce changement soit perceptible aux humains.

Enfin, il reste difficile d'expliquer comment la machine obtient ses résultats. Dans ce contexte, un fonctionnement totalement autonome ne semble pas envisageable : cela reviendrait à donner le pouvoir décisionnel à un système de type « boîte noire » sans aucune garantie d'objectivité. Ce point conduit le sociologue Jean-Gabriel Ganascia à affirmer que le pouvoir décisionnel doit rester à l'humain.

- **Principales contraintes en termes de besoins d'échantillons**

Pour développer les algorithmes de reconnaissance faciale, les laboratoires de recherche et développement (R&D) ont besoin de gigantesques bases de données « contextualisées » (des centaines de milliers de données), i.e. des images plaçant les personnes dans le contexte de la recherche à mener et les données d'entraînement associées pour réaliser les correspondances (« hits »). Ces données sont donc « l'or noir » des algorithmes d'apprentissage

Or, en Europe, ces données sont qualifiées de « données sensibles à caractère personnel ». Leur regroupement et leur exploitation relèvent de règles contraignantes selon les finalités. Les bases de données du NIST<sup>8</sup> ne sont pas disponibles pour la R&D (recherche et développement), mais uniquement dans le cadre de « benchmarks » collaboratifs. Les laboratoires utilisent donc des bases de données publiques - « data-set » académiques ou bibliothèques de visages de personnalités (acteurs, célébrités ...) - qu'ils trient ou enrichissent pour répondre au mieux aux cas d'usage des algorithmes développés.



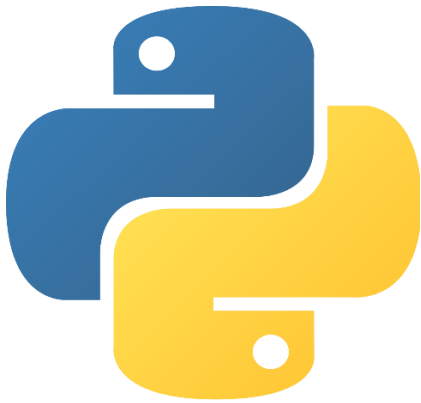
# Chapitre III

## Outils utilisés et Réalisation du projet

Ce chapitre correspond à la phase de construction du service de synchronisation et de l'application principale. Nous commencerons tout d'abord, par la présentation de l'environnement de travail utilisé pour développer notre projet. Puis nous donnerons quelques screenshots.

## I. Environnement de travail

### 1. Langage de programmation



Python est un langage de programmation multi-paradigme, polyvalent, interprété et de haut niveau conçu pour être facile à lire et simple à mettre en oeuvre. Python permet aux programmeurs d'utiliser différents styles de programmation pour créer des programmes simples ou complexes, obtenir des résultats plus rapides et écrire du code presque comme s'il parlait dans un langage humain. Parmi les systèmes et applications populaires qui ont utilisé Python lors du développement, citons les moteurs

de recherche Google, YouTube, Bit Torrent, Google App Engine, Eve Online...

### 2. Outils utilisés

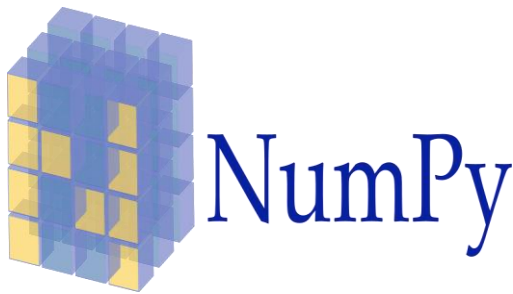


Pour rédiger et exécuter un programme en python, il existe de nombreux logiciels (appelés interfaces graphiques ou IDE : Environnement de Développement Intégré) possibles. Parmi les interfaces possibles (Notebook, IDLE, Eclipse, bloc-notes, Word ...), nous avons choisi d'utiliser le logiciel «Jupyter» Celui-ci présente de nombreux avantages d'aide à la programmation.

Jupyter est une application web utilisée pour programmer dans plus de 40 langages de programmation, dont Python, Julia, Ruby, R, ou encore Scala2. Jupyter est une évolution du projet IPython. Jupyter permet de réaliser des calepins ou notebooks, c'est-à-dire des programmes contenant à la fois du texte en markdown et du code en Julia, Python, R... Ces notebooks sont utilisés en science des données pour explorer et analyser des données.

### 3. Bibliothèques utilisées

#### 3.1. NumPy :

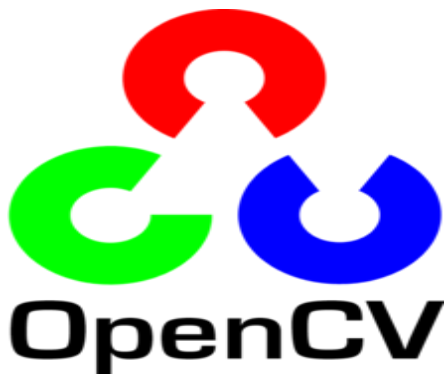


NumPy est une extension du langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux.

Plus précisément, cette bibliothèque logicielle libre et open source fournit de multiples fonctions permettant notamment de créer directement un tableau depuis un fichier ou au contraire de sauvegarder un tableau dans un fichier, et manipuler des vecteurs, matrices et polynômes.

NumPy est la base de SciPy, regroupement de bibliothèques Python autour du calcul scientifique

#### 3.2. OpenCV :



OpenCV (Open Source Computer Vision Library) est une bibliothèque de logiciels de vision et d'apprentissage automatique open-source. OpenCV a été construit pour fournir une infrastructure commune pour les applications de vision informatique et pour accélérer l'utilisation de la perception des machines dans les produits commerciaux. OpenCV étant un produit sous licence BSD, il est facile pour les entreprises d'utiliser et de modifier le code.

La bibliothèque compte plus de 2500 algorithmes optimisés, qui comprennent un ensemble complet d'algorithmes de vision et d'apprentissage machine classiques et de pointe. Ces algorithmes peuvent être utilisés pour détecter et reconnaître les visages, identifier les objets, classer les actions humaines dans les vidéos, suivre les mouvements des caméras, suivre les objets en mouvement, extraire des modèles 3D d'objets, produire des nuages de points 3D à partir de caméras stéréo, assembler des images pour produire une image haute résolution d'une scène entière, trouver des images similaires à partir d'une base de données d'images, supprimer les yeux rouges des images prises à l'aide de flash, suivre les mouvements des yeux, reconnaître le paysage et établir des marqueurs pour le superposer avec la réalité augmentée, etc. OpenCV a plus de 47 mille personnes de la communauté des utilisateurs et le nombre estimé de téléchargements de plus de 18

millions. La bibliothèque est largement utilisée par les entreprises, les groupes de recherche et les organismes gouvernementaux

### 3.3. Pillow python :



Python Imaging Library (ou PIL) est une bibliothèque de traitement d'images pour le langage de programmation Python. Elle permet d'ouvrir, de manipuler, et de sauvegarder différents formats de fichiers graphiques.

La bibliothèque est disponible librement selon les termes de la Python Imaging Library licence.

Plus maintenue depuis 2009, elle est remplacée à partir de 2010 par un dérivé proche, Pillow.

La bibliothèque supporte plusieurs formats de fichier, parmi lesquels PNG, JPEG, GIF, TIFF, et BMP. Il est aussi possible d'ajouter son propre décodeur de fichiers pour étendre le nombre de formats disponibles.

## 4. Modèles utilisés

### 4.1. Cascade Classifier

Travailler avec une cascade boostée de classificateurs faibles comprend deux étapes principales : la formation et la détection. L'étape de détection utilisant des modèles basés sur HAAR ou LBP est décrite dans le **didacticiel de détection d'objets**. Cette documentation donne un aperçu des fonctionnalités nécessaires pour former votre propre cascade renforcée de classificateurs faibles. Le guide actuel passera par toutes les différentes étapes : collecte des données de formation, préparation des données de formation et exécution de la formation réelle du modèle.

Pour prendre en charge ce didacticiel, plusieurs applications OpenCV officielles seront utilisées : `opencv_createsamples`, `opencv_annotation`, `opencv_traincascade` et `opencv_visualisation`.

#### Objectif :

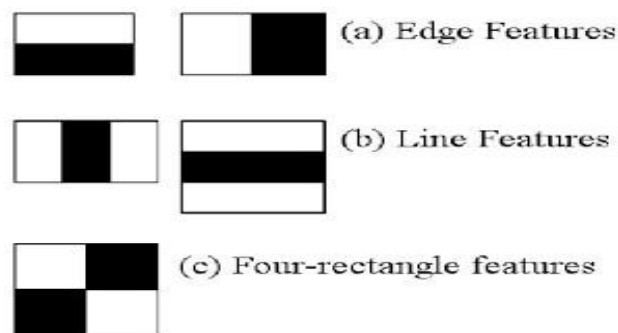
- Nous apprendrons comment fonctionne la détection d'objets en cascade Haar.
- Nous verrons les bases de la détection des visages et des yeux à l'aide des classificateurs en cascade basés sur les fonctionnalités de Haar
- Nous utiliserons la classe **cv : CascadeClassifier** pour détecter des objets dans un flux vidéo. En particulier, nous utiliserons les fonctions :

- **cv : CascadeClassifier : load** pour charger un fichier de classificateur .xml. Il peut s'agir d'un classificateur Haar ou LBP
- **cv : CascadeClassifier : detectMultiScale** pour effectuer la détection.

### **Théorie :**

La détection d'objets à l'aide des classificateurs en cascade basés sur les fonctionnalités de Haar est une méthode efficace de détection d'objets proposée par Paul Viola et Michael Jones dans leur article, "Rapid Object Detection using a Boosted Cascade of Simple Features" en 2001. Il s'agit d'une approche basée sur l'apprentissage automatique où la fonction en cascade est formée à partir d'un grand nombre d'images positives et négatives. Il est ensuite utilisé pour détecter des objets dans d'autres images.

Ici, nous allons travailler avec la détection des visages. Initialement, l'algorithme a besoin de beaucoup d'images positives (images de visages) et d'images négatives (images sans visages) pour former le classificateur. Ensuite, nous devons en extraire des fonctionnalités. Pour cela, les fonctionnalités Haar illustrées dans l'image ci-dessous sont utilisées. Ils sont exactement comme notre noyau convolutionnel. Chaque entité est une valeur unique obtenue en soustrayant la somme des pixels sous le rectangle blanc de la somme des pixels sous le rectangle noir.



Désormais, toutes les tailles et emplacements possibles de chaque noyau sont utilisés pour calculer de nombreuses fonctionnalités. (Imaginez juste combien de calcul il a besoin? Même une fenêtre 24x24 donne plus de 160000 fonctionnalités). Pour chaque calcul d'entité, nous devons trouver la somme des pixels sous des 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. Sympa, non? Cela rend les choses super rapides.

Mais parmi toutes ces caractéristiques que nous avons calculées, la plupart d'entre elles ne sont pas pertinentes. Par exemple, considérez l'image ci-dessous. La rangée du haut montre deux bonnes caractéristiques. La première caractéristique sélectionnée semble se concentrer sur la propriété que la région des yeux est souvent plus foncée que la région du nez et des joues. La deuxième caractéristique sélectionnée repose sur la propriété que

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 pouvons-nous sélectionner les meilleures fonctionnalités parmi 160000+ fonctionnalités? Il est réalisé par **Adaboost**.



Pour cela, nous appliquons chaque fonctionnalité sur toutes les images d'entraînement. Pour chaque entité, il trouve le meilleur seuil qui classera les faces positives et négatives. De toute évidence, il y aura des erreurs ou des erreurs de classification. Nous sélectionnons les fonctionnalités avec un taux d'erreur minimum, ce qui signifie que ce sont les fonctionnalités qui classent le plus précisément les images faciales et non faciales. (Le processus n'est pas aussi simple que cela. Chaque image reçoit un poids égal au début. Après chaque classification, les poids des images mal classées sont augmentés. Ensuite, le même processus est effectué. De nouveaux taux d'erreur sont calculés. De nouveaux poids également. le processus se poursuit jusqu'à ce que la précision ou le taux d'erreur requis soit atteint ou que le nombre requis de fonctions soit trouvé).

Le classificateur final est une somme pondérée de ces classificateurs faibles. Il est appelé faible parce qu'il ne peut à lui seul classer l'image, mais forme avec d'autres un classificateur fort. Le document indique que même 200 fonctionnalités offrent une détection avec une précision de 95%. Leur configuration finale avait environ 6000 fonctionnalités. (Imaginez une réduction de 160000+ fonctionnalités à 6000 fonctionnalités. C'est un gros gain).

Alors maintenant, vous prenez une image. Prenez chaque fenêtre 24x24. Appliquez-y 6000 fonctionnalités. Vérifiez si c'est face ou non. Wow. N'est-ce pas un peu inefficace et chronophage ? Oui, ça l'est. Les auteurs ont une bonne solution pour cela.

Dans une image, la majeure partie de l'image est une zone sans visage. Il est donc préférable d'avoir une méthode simple pour vérifier si une fenêtre n'est pas une zone de visage. Si ce n'est pas le cas, jetez-le en une seule fois et ne le traitez pas à nouveau. Concentrez-vous plutôt sur les régions où il peut y avoir un visage. De cette façon, nous passons plus de temps à vérifier les régions de visage possibles.

Pour cela, ils ont introduit le concept de **cascade de classificateurs**. Au lieu d'appliquer toutes les 6000 fonctionnalités sur une fenêtre, les fonctionnalités sont regroupées en différentes étapes des classificateurs et appliquées une par une. (Normalement, les premières étapes contiennent beaucoup moins de fonctionnalités). Si une fenêtre échoue à la première étape, jetez-la. Nous ne considérons pas les fonctionnalités restantes dessus. S'il réussit, appliquez la deuxième étape des fonctionnalités et poursuivez le processus. La fenêtre qui passe toutes les étapes est une région de visage. Comment est ce plan!

Le détecteur des auteurs avait plus de 6000 caractéristiques avec 38 étapes avec 1, 10, 25, 25 et 50 caractéristiques dans les cinq premières étapes. (Les deux fonctionnalités de l'image ci-dessus sont en fait obtenues comme les deux meilleures fonctionnalités d'Adaboost). Selon les auteurs, en moyenne 10 fonctionnalités sur 6000+ sont évaluées par sous-fenêtre.

Il s'agit donc d'une explication simple et intuitive du fonctionnement de la détection des visages Viola-Jones. Lisez l'article pour plus de détails ou consultez les références dans la section Ressources supplémentaires.

La cascade est un cas particulier d'apprentissage d'ensemble basé sur la concaténation de plusieurs classificateurs, en utilisant toutes les informations collectées à partir de la sortie d'un classificateur donné comme informations supplémentaires pour le classificateur suivant dans la cascade. Contrairement aux ensembles de vote ou d'empilement, qui sont des systèmes multi-experts, la mise en cascade est un système en plusieurs étapes.

Les classificateurs en cascade sont formés avec plusieurs centaines de vues d'échantillons "positives" d'un objet particulier et des images arbitraires "négatives" de la même taille. Une fois le classificateur formé, il peut être appliqué à une région d'une image et détecter l'objet en question. Pour rechercher l'objet dans le cadre entier, la fenêtre de recherche peut être déplacée sur l'image et vérifier chaque emplacement pour le classificateur. Ce processus est le plus couramment utilisé dans le traitement d'images pour la détection et le suivi d'objets, principalement la détection et la reconnaissance faciales.

Le premier classificateur en cascade a été le détecteur de visage de Viola et Jones (2001). L'exigence de ce classifieur devait être rapide afin d'être implémentée sur des processeurs de faible puissance, tels que des caméras et des téléphones.

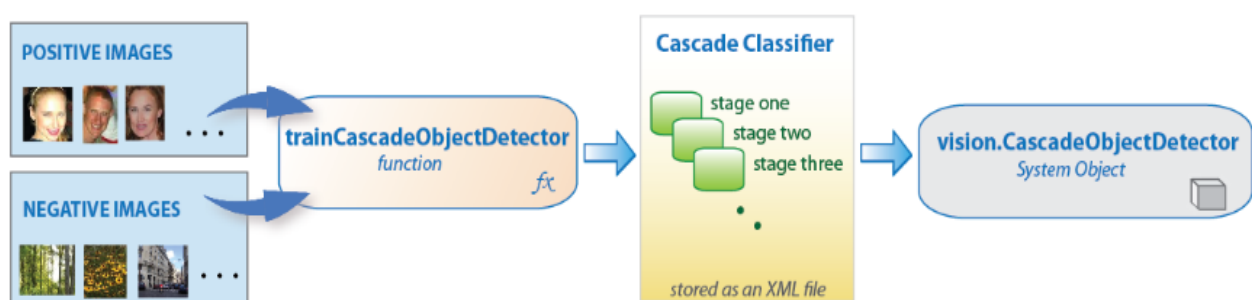


Figure : Cascade Classifier

## 4.2. Techniques de reconnaissance de visages :

### 4.2.1. Réseaux de neurones artificiels :

Les réseaux de neurones artificiels (RNA) : Les réseaux de neurones artificiels sont des réseaux fortement connectés de processeurs élémentaires fonctionnant en parallèle. Chaque processeur élémentaire calcule une sortie unique sur la base des informations qu'il reçoit. Toute structure hiérarchique de réseaux est évidemment un réseau.

Les RNA ont été initialement inspirés de la psychologie du système nerveux. Dans les années quarante le neurone formel introduit par J.McCulloch et W.Pitts.

Un neurone formel est une fonction algébrique non-linéaire de variables réelles appelées entrées, qui tente de reproduire ce mode de fonctionnement, il effectue une somme pondérée des signaux d'entrée qui lui parviennent. Cette somme pondérée sert de paramètre à une fonction, souvent non linéaire, qui la transforme en nouveau signal transmis à la sortie. En référence au fonctionnement du neurone biologique, la fonction intervenant après la sommation des entrées est appelée fonction d'activation.

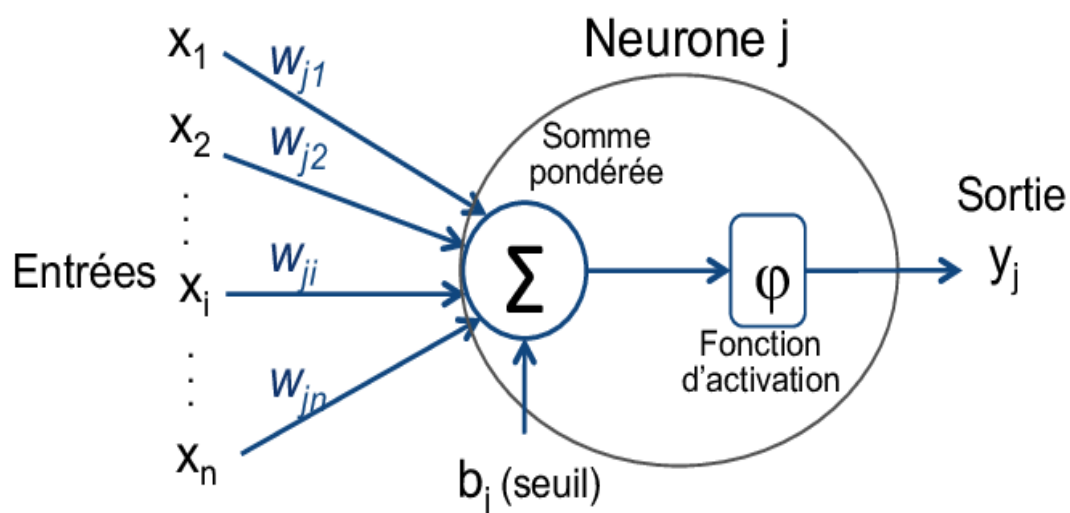


Figure : Neurone formel

Les RNA ont été utilisés dans plusieurs applications, particulièrement pour la classification de données, la modélisation de processus complexes et le traitement non linéaire des signaux. Dans le cas de la reconnaissance de visages, les réseaux de neurones sont utilisés pour la détection de visages, pour l'extraction de signatures et pour la classification.



## II. Réalisation du projet :

Mener à bien un projet de Machine Learning consiste à réaliser six étapes consécutives :

- 1 - Définition du problème à résoudre**
- 2 - Acquisition des données d'apprentissages et de tests**
- 3 - Préparer et nettoyer les données**
- 4 - Analyser, explorer les données**
- 5 - Choisir un modèle d'apprentissage**
- 6 - Visualiser les résultats, et ajuster ou modifier le modèle d'apprentissage**

La phase de préparation des données est la plus importante dans un projet de Machine Learning, car en tant qu'humains, nous devons essayer de trouver les données les plus intéressantes qui nous permettront de répondre au problème donné.

Bien plus qu'une simple analyse des données, il faut déterminer comment il nous est possible de résoudre manuellement le problème, à partir des informations dont nous disposons, avant de le confier à la machine.

Ainsi, un même jeu de données peut être exploité différemment en fonction du problème donné.

Dans cette partie on va aborder trois parties :

- **Phase d'apprentissage**
- **Phase de training**
- **Phase de test**

### 1. Phase d'apprentissage

Nous allons dans un premier temps initialiser le classifieur d'OpenCV qui prend en paramètre le fichier de configuration « haarcascade\_frontalface\_default.xml » ce modèle sert à la détection d'objets à l'aide de classificateurs en cascade basés sur les fonctionnalités de Haar est une méthode efficace de détection d'objets proposée par Paul Viola et Michael Jones. Il s'agit d'une approche basée sur l'apprentissage automatique où la fonction en cascade est formée à partir d'un grand nombre d'images positives et négatives. Il est ensuite utilisé pour détecter des objets dans d'autres images.

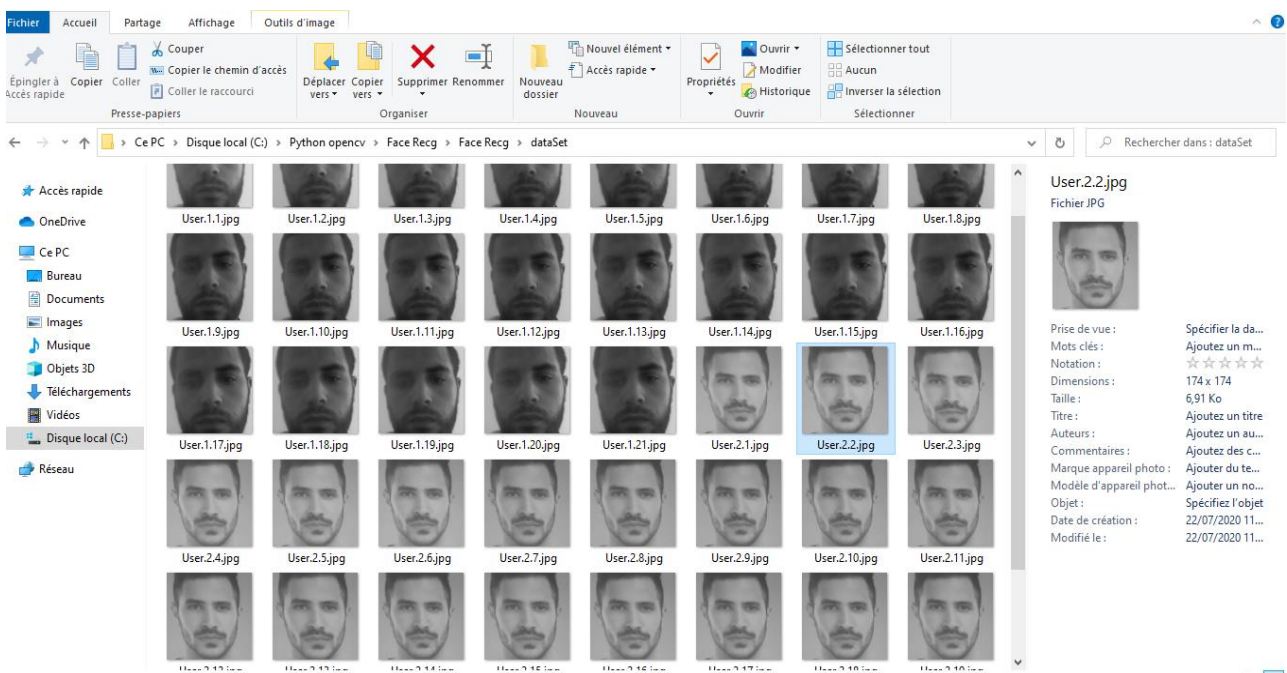
Ici, nous allons travailler avec ce modèle pour la détection des visages capturés par la caméra. Sur lesquelles il va se baser pour la création de sa propre base de données après les convertir en niveau de gris afin de pouvoir utiliser la fonction de reconnaissance faciale.

## Création de Dataset (Préparer et nettoyer les données)

```
File Edit Selection View Go Debug Terminal Help
trainer.py datasetCreator.py detector.py
C: > Users > ELMSAYER > Desktop > ML > Face Recg > Face Recg > datasetCreator.py
1 import cv2
2 cam = cv2.VideoCapture(0)
3 detector=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
4
5 Id=raw_input('enter your id')
6 sampleNum=0
7 while(True):
8     ret, img = cam.read()
9     gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
10    faces = detector.detectMultiScale(gray, 1.3, 5)
11    for (x,y,w,h) in faces:
12        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
13
14        #incrementing sample number
15        sampleNum=sampleNum+1
16        #saving the captured face in the dataset folder
17        cv2.imwrite("dataSet/User."+Id+'.'+ str(sampleNum) + ".jpg", gray[y:y+h,x:x+w])
18
19        cv2.imshow('frame',img)
20        #wait for 100 miliseconds
21        if cv2.waitKey(100) & 0xFF == ord('q'):
22            break
23        # break if the sample number is morethan 20
24        elif sampleNum>20:
25            break
26    cam.release()
27    cv2.destroyAllWindows()
28
```

Figure : Phase de préparation

## Capture de dataSet



## 2. Phase de training

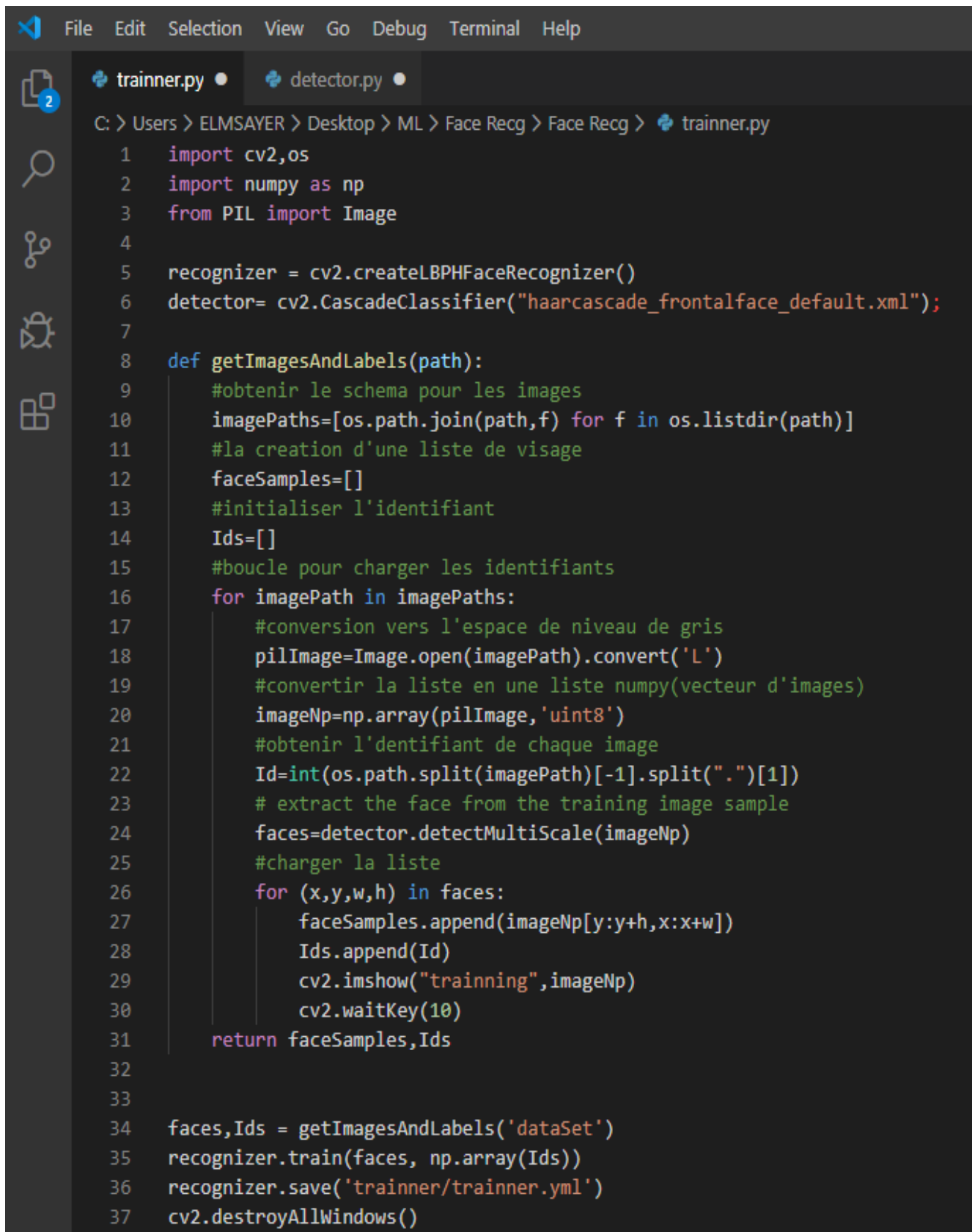
L'analyse de visage, c'est l'étape la plus importante de notre projet, celle où on va remplacer la totalité de l'image par « seulement » 128 nombres. Ces nombres, en rapport avec l'image, seront déterminés de manière à ce qu'ils différencient des individus facilement. Par exemple, on pourrait imaginer que les nombres suivants sont les mêmes sur toutes mes photos et sont vraiment spécifiques à moi-même :

- l'écart entre mes yeux
- la hauteur de mon front
- l'angle de ma mâchoire

Quelles sont les 128 valeurs à retenir dans une photo qui sont différentes pour chaque visage différent et extrêmement proches pour chaque visage d'une même personne !

Comment cela fonctionne-t-il ? un réseau de neurones convolutifs, nommé VGG, a appris à générer un vecteur de 128 nombres à partir d'une image en entrée. On lui donne donc la photo de nous recadrée et déformée, et il nous renvoie le vecteur qui est « spécifique à nous-même ». Dans notre projet ces vecteurs sont stockés dans un fichier « `trainer.yml` »

Donc dans notre base de données, on n'a pas stocké les photos mais directement les vecteurs (à 128 dimensions) représentant chacune des photos. Ainsi, lorsque le vecteur de notre caméra arrive, il suffit de chercher dans notre base de données lequel de ces vecteurs est le plus proche.



```
File Edit Selection View Go Debug Terminal Help

trainer.py • detector.py •

C: > Users > ELMSAYER > Desktop > ML > Face Recg > Face Recg > trainer.py

1  import cv2,os
2  import numpy as np
3  from PIL import Image
4
5  recognizer = cv2.createLBPHFaceRecognizer()
6  detector= cv2.CascadeClassifier("haarcascade_frontalface_default.xml");
7
8  def getImagesAndLabels(path):
9      #obtenir le schema pour les images
10     imagePaths=[os.path.join(path,f) for f in os.listdir(path)]
11     #la creation d'une liste de visage
12     faceSamples=[]
13     #initialiser l'identifiant
14     Ids=[]
15     #boucle pour charger les identifiants
16     for imagePath in imagePaths:
17         #conversion vers l'espace de niveau de gris
18         pilImage=Image.open(imagePath).convert('L')
19         #convertir la liste en une liste numpy(vecteur d'images)
20         imageNp=np.array(pilImage,'uint8')
21         #obtenir l'identifiant de chaque image
22         Id=int(os.path.split(imagePath)[-1].split(".")[1])
23         # extract the face from the training image sample
24         faces=detector.detectMultiScale(imageNp)
25         #charger la liste
26         for (x,y,w,h) in faces:
27             faceSamples.append(imageNp[y:y+h,x:x+w])
28             Ids.append(Id)
29             cv2.imshow("training",imageNp)
30             cv2.waitKey(10)
31     return faceSamples,Ids
32
33
34 faces,Ids = getImagesAndLabels('dataSet')
35 recognizer.train(faces, np.array(Ids))
36 recognizer.save('trainer/trainer.yml')
37 cv2.destroyAllWindows()
```

Figure : Phase de trainnig

```

File Edit Selection View Go Debug Terminal Help
trainer.py • datasetCreator.py Face-and-eyes-Detection.py X detector.py •
C: > Users > ELMSAYER > Desktop > ML > Face Recg > Face Recg > Face-and-eyes-Detection.py
1 import numpy as np
2 import cv2
3 face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
4 eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')
5 cap=cv2.VideoCapture(0)
6
7 while(True):
8     ret,img=cap.read();
9     gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY);
10    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
11    for (x,y,w,h) in faces:
12        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
13        roi_gray = gray[y:y+h, x:x+w]
14        roi_color = img[y:y+h, x:x+w]
15        eyes= eye_cascade.detectMultiScale(roi_gray)
16        for (ex,ey,ew,eh) in eyes:
17            cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
18    cv2.imshow('img',img);
19    if cv2.waitKey(1) & 0xFF == ord('q'):
20        break;
21    cam.release();
22    cv2.destroyAllWindows()
23

```

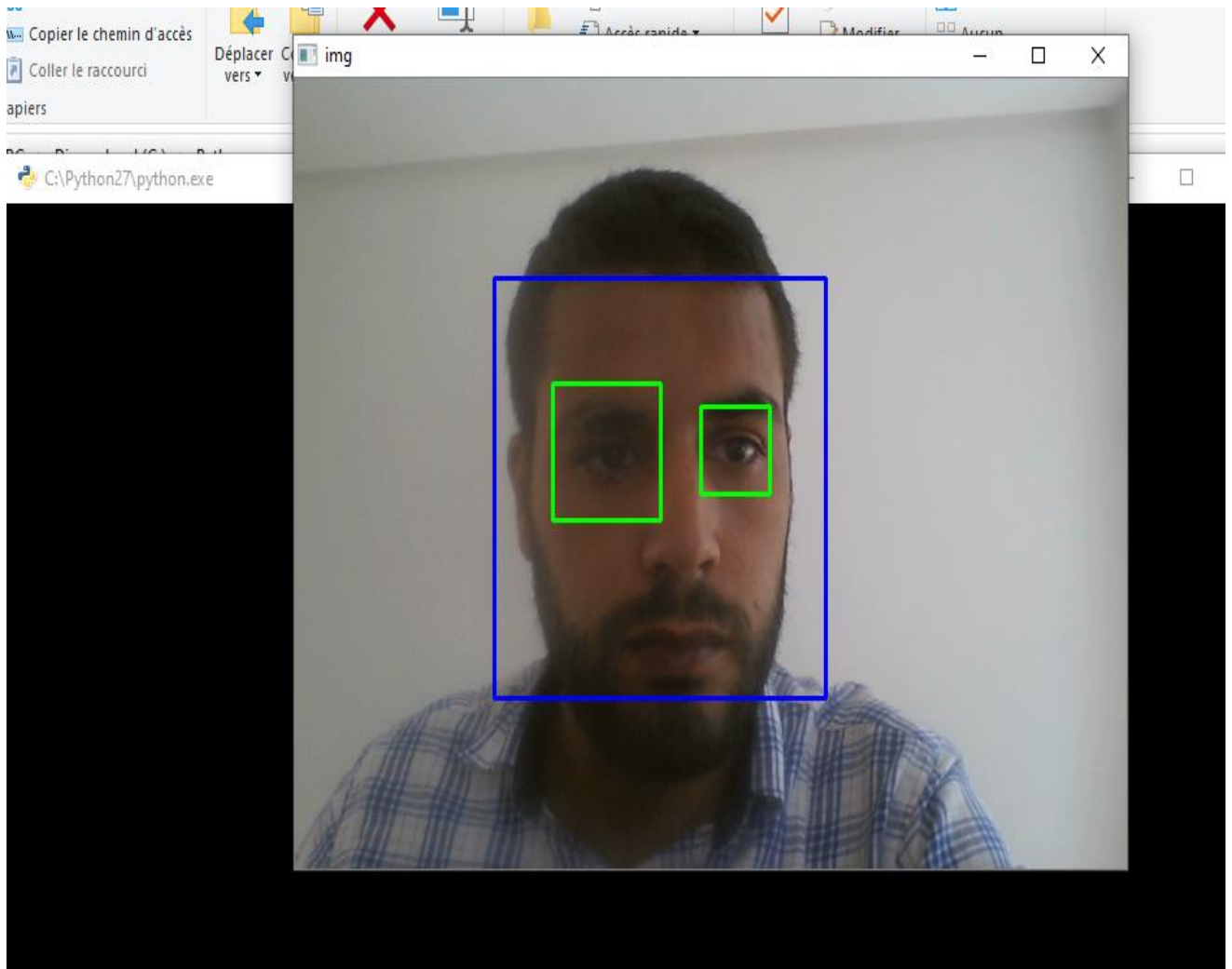
Figure : Phase de training détection du visage et yeux

```

File Edit Selection View Go Debug Terminal Help
trainer.py • datasetCreator.py Face-and-eyes-Detection.py FaceDetection.py X
C: > Users > ELMSAYER > Desktop > ML > Face Recg > Face Recg > FaceDetection.py
1 import numpy as np
2 import cv2
3 face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
4 eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')
5 cap=cv2.VideoCapture(0)
6
7 while(True):
8     ret,img=cap.read();
9     gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY);
10    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
11    for (x,y,w,h) in faces:
12        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
13        roi_gray = gray[y:y+h, x:x+w]
14        roi_color = img[y:y+h, x:x+w]
15        eyes= eye_cascade.detectMultiScale(roi_gray)
16        for (ex,ey,ew,eh) in eyes:
17            cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
18    cv2.imshow('img',img);
19    if cv2.waitKey(1) & 0xFF == ord('q'):
20        break;
21    cam.release();
22    cv2.destroyAllWindows()
23

```

Figure : Phase de training détection de visage

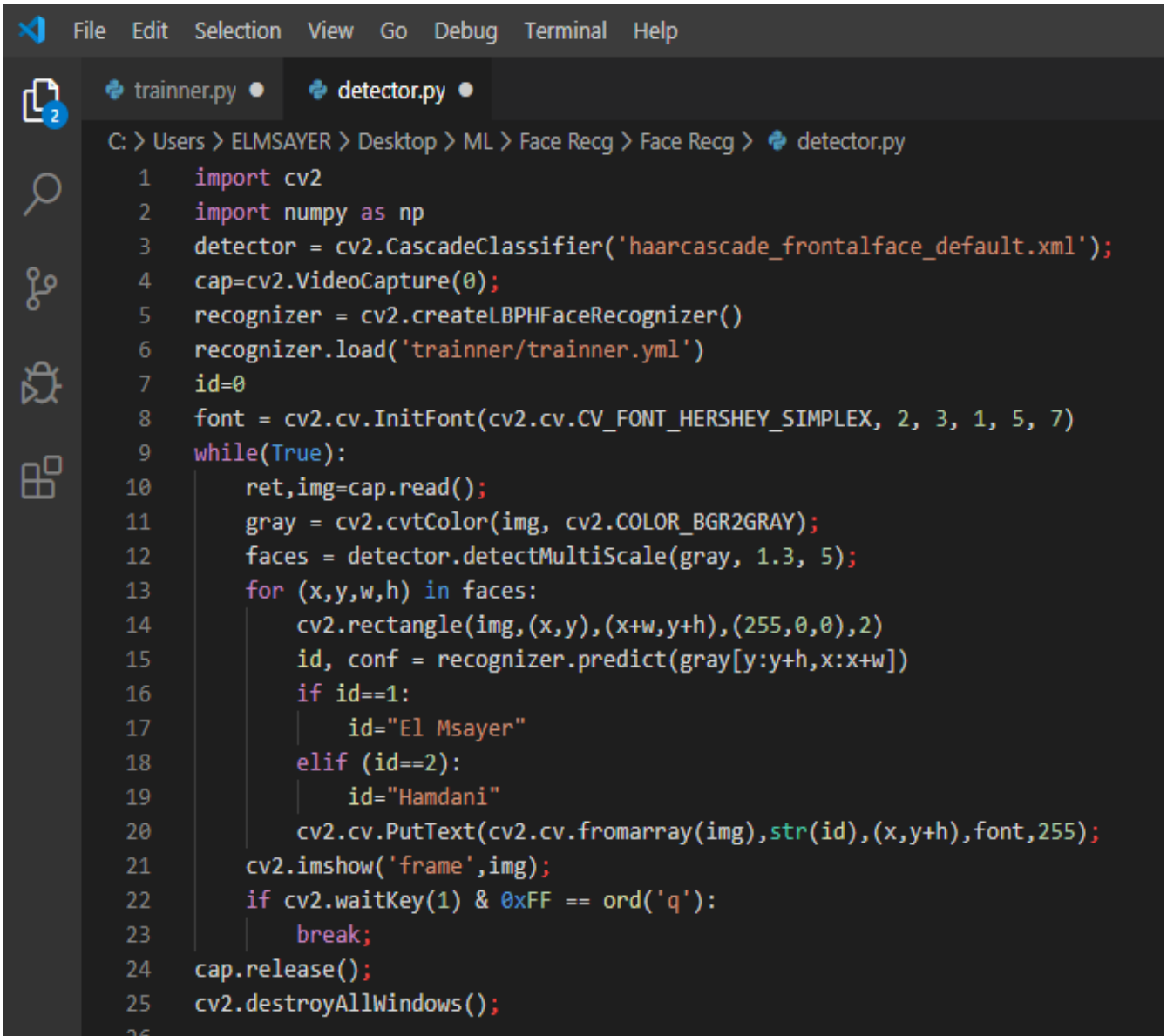


**Figure : Phase de training détection de visage et yeux**

### 3. Phase de test

Dans cette phase on va vérifier s'il a capturé une image ou non si oui faire le traitement suivant :

- Prendre chaque image du flux vidéo la convertir en gris
- Comparer avec chaque vecteur de la base de données : dans notre base chaque vecteur est associé à un identificateur (code1)



```
File Edit Selection View Go Debug Terminal Help
trainer.py • detector.py •
C: > Users > ELMSAYER > Desktop > ML > Face Recg > Face Recg > detector.py
1 import cv2
2 import numpy as np
3 detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml');
4 cap=cv2.VideoCapture(0);
5 recognizer = cv2.createLBPHFaceRecognizer()
6 recognizer.load('trainer/trainer.yml')
7 id=0
8 font = cv2.cv.InitFont(cv2.cv.CV_FONT_HERSHEY_SIMPLEX, 2, 3, 1, 5, 7)
9 while(True):
10     ret,img=cap.read();
11     gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY);
12     faces = detector.detectMultiScale(gray, 1.3, 5);
13     for (x,y,w,h) in faces:
14         cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
15         id, conf = recognizer.predict(gray[y:y+h,x:x+w])
16         if id==1:
17             id="El Msayer"
18         elif (id==2):
19             id="Hamdani"
20         cv2.cv.PutText(cv2.cv.fromarray(img),str(id),(x,y+h),font,255);
21     cv2.imshow('frame',img);
22     if cv2.waitKey(1) & 0xFF == ord('q'):
23         break;
24 cap.release();
25 cv2.destroyAllWindows();
26
```

Figure 5 : Phase de test

Si le visage détecté existe déjà dans la base de données il va retourner l'id associé à ce dernier sinon **unknown**



## **Conclusion Générale et perspectives**

L'objectif de ce projet est de concevoir et d'implémenter une application de reconnaissance faciale capable, en temps réel, de reconnaître les visages. Vu la quantité de logiciels potentiels (sécurité, réseaux sociaux,...) pouvant se baser sur cette application, celle-ci doit répondre à des exigences de rapidité et de robustesse des résultats.

Il nous a fallu résoudre plusieurs problèmes algorithmiques ayant plus ou moins de rapport avec les mathématiques, discipline importante dans le traitement d'images en général.

Il existe une variété de techniques consacrées à la détection de visage. Les systèmes de suivi de visage se sont beaucoup développés ces dernières années grâce à l'amélioration du matériel et à la forte demande industrielle.

Parmi les perspectives ouvertes à ce projet, l'utilisation d'autres méthodes de reconnaissance, et la combinaison avec d'autres technologies biométriques comme l'empreinte digitale ou l'iris pour finaliser l'application multimodale et fiabiliser le système en diminuant la sensibilité aux conditions d'éclairage par de nouvelles normalisations.



## Webographie

[https://docs.opencv.org/master/d9/df8/tutorial\\_root.html](https://docs.opencv.org/master/d9/df8/tutorial_root.html)

<https://docs.opencv.org/>

<https://www.pyimagesearch.com/2018/09/24/opencv-face-recognition/>

<https://www.geeksforgeeks.org/python-opencv-cv2-puttext-method/>

<https://towardsdatascience.com/face-detection-in-2-minutes-using-opencv-python-90f89d7c0f81>

<https://www.programcreek.com/python/example/83399/cv2.putText>

<https://realpython.com/face-recognition-with-python/>

<http://www.windowscpp.com/PDF/227.pdf>