UNIVERSSITE BADJI MOKHTAR- ANNABA

جامعة باجي مختار- عنابـة

Faculté : Sciences de L'ingéniorat

Année : 2021

Département : Electronique

MEMOIRE

Présenté pour l'obtention du diplôme de : MASTER

Titre :

Reconnaissance d'état des yeux par les méthodes d'apprentissage approfondie

Filière : Télécommunications.

Spécialité : Réseaux et Télécommunications.

Par :

Aouadi Noussaiba

Berrahil Chahira

Superviseur : Dr. Seif Allah El Mesloul Nasri — Université d'Annaba

Co-Superviseur : Dr. Djamel Eddine Benrachou — Université de Technologie Queensland, Australie

Devant les JURY

Présidente : Prof. Amira Yahi — Université d'Annaba

Examinatrice : Prof. Leila Sahraoui — Université d'Annaba

Remercîment

Nous remercions tout d'abord Dieu le tout puissant de nous avoir donné la volonté et la persévérance pour réaliser ce travail.

Nous tenons à saisir cette occasion et adresser nos profonds remerciements et nos profondes reconnaissances à Dr. Seif Allah El Mesloul Nasri, l'encadrant de mémoire, ainsi qu'à Dr. Djamel Eddine Benrachou, le Co-encadreur, pour leurs précieux conseils et leurs orientations ficelées tout au long de notre travail.

Nos vifs remerciements vont également aux membres du jury pour l'intérêt qu'ils ont porté à notre recherche en acceptant d'examiner notre travail et de l'enrichir par leurs propositions.

Nous souhaitons adresser nos remerciements les plus sincères aux personnes qui nous ont apporté leur aide et qui ont contribué à l'élaboration de ce mémoire ainsi qu'à la réussite de cette année universitaire. Nous remercions tous nos enseignants pour leur dévouement, leur patience et leur contribution à notre formation.

Enfin, nous adressons nos plus sincères remerciements à tous nos proches et amis, qui nous ont toujours encouragés au cours de la réalisation de ce mémoire.

Merci à tous et à toutes.

Dédicace

Du profond de mon cœur, je dédie ce travail

A la mémoire de ma mère, décédée trop tôt, qui sera contente d'apprendre que sa fille a enfin terminé ses études. J'espère que, du monde qui est sien maintenant, elle apprécie cet humble geste comme preuve de reconnaissance de la part d'une fille qui a toujours priée pour le salut de son âme.

Puisse Dieu, le tout puissant, l'avoir en sa sainte miséricorde.

A mon père, aucune dédicace ne saurait exprimer mon respect, et ma considération pour les sacrifices que vous avez consentis pour mon instruction et mon bien être. Je vous remercie pour le soutien que vous me portez depuis mon enfance et j'espère que votre bénédiction m'accompagne toujours. Puisse Dieu, le Très Haut, vous accorder santé, bonheur et longue vie.

A mes frères, Chakib et Badis qui m'ont encouragé sans cesse et cru en moi, les mots ne suffisent guère pour exprimer l'attachement, l'amour et l'affection que je porte pour vous.

A mes chers cousines, Sara, Maria et Nesrine, qui n'ont jamais cessée de me soutenir, merci d'être toujours à mes côtés par votre amour, bon humeur et joie. Que ce travail vous témoigne de ma sincère affection.

Merci pour leurs amours et encouragement.

Sans oublier mon binôme et ma très chère amie Noussaiba, pour son soutien moral, sa patience et sa compréhension tout au long de ce projet.

Ainsi que tous mes amis et collègues, en souvenir des moments agréables que nous avons passés ensemble.

Chahira.

Dédicace

A ma très chère mère

Maman qui m'a soutenu et encouragée durant ces années d'études. Ton affection me couvre, ta bienveillance me guide et ta présence à mes côtés a toujours été ma source de force pour affronter les différents obstacles.

A mon très cher père

Tu as toujours été à mes côtés pour me soutenir et m'encourager, je ne pouvais espérer avoir un père plus formidable que toi J'ai eu beaucoup de chance que le bon Dieu t'a choisi pour que tu sois mon papa.

Que ce travail traduit ma gratitude et mon affection.

Quoi que je fasse ou que je dise, je ne saurai point vous remercier comme il se doit.

A mes chers frères Abdel Bari, Lokmane, ma sœur Oulfa et mes chères cousines qui ont partagé avec moi tous les moments d'émotion lors de la réalisation de ce travail. Ils m'ont chaleureusement supporté et encouragé tous au long de mon parcours.

Puisse Dieu vous donne la santé, le bonheur, le courage et surtout la réussite.

A mes très chers amis, mes proches et à ceux qui me donnent de l'amour et de la vivacité.

A tous ceux qui ont une place dans mon cœur.

Sans oublier mon binôme et ma chère amie Chahira pour son soutien moral, sa patience et sa compréhension tout au long de ce projet, Merci pour tous ma meilleure.

À tous, je présente mes sincères remerciements et ma profonde gratitude.

Noussaiba.

Résumé

Les applications de détection des yeux et la reconnaissance de leurs différents états ont reçu une attention particulière et assez croissante ces dernières années et dans plusieurs domaines, telles que celles basées sur les techniques de vision par ordinateur et bien d'autres domaines ; tel que l'interaction homme-machine (IHM), la détection de la fatigue des conducteurs des véhicules, l'analyse et la reconnaissance des expressions faciales, etc. Cependant, malgré les progrès réalisés dans ces domaines, et les dispositifs existants sur le marché, pouvoir fournir une réponse correcte par ces systèmes est une tâche fastidieuse, en raison de plusieurs contraintes du monde réel, à savoir le changement de l'éclairage, les images bruitées, l'occlusion, la qualité des images, les variations des textures non-rigides de la peau humaine, etc. Pour surmonter toutes ces difficultés, ce travail utilise une combinaison des techniques de vision par ordinateur (CV) et l'apprentissage profond pour assurer une détection précise des yeux pour ensuite reconnaître leurs états d'ouverture et de fermeture. Plus particulièrement, l'algorithme conçu combine des méthodes conventionnelles et des récentes pour la détection de la région du visage, pour ensuite délimiter la région oculaire à l'aide des HaarCascade, et l'apprentissage profond à base de réseaux de neurones convolutifs (CNN), dans le but de mesurer et reconnaître les états ouverts et fermés des yeux. Notre algorithme peut être exécuté en ligne avec une simple caméra Web, et peut fournir une réponse précise et rapide des différentes étapes successives, à savoir la détection du visage, la détection des yeux et enfin la reconnaissance des états : des yeux ouverts et des yeux fermés.

Mots clés :

Vision par ordinateur, l’apprentissage approfondie, La détection de visage, La détection des yeux, La localisation des yeux, La reconnaissance d’état des yeux.

Abstract

The detection of the eyes and their binary states (open / closed) recognition have received particular and increasing attention this recent years, in several fields of research and application in computer vision and in many other domains, such as the human-machine interaction (HMI), fatigue detection, automatic facial expression, etc. Despite the progress made in this domains, and existing devices on the market, recognising the correct eye states remains a real tedious task, due to several constraints encountered in the real world, such as change of lighting, image noise, occlusion (individuals wearing glasses / hats / beards), image signal quality, variations in the non-rigid textures of human skin, etc. To overcome these issues, this manuscript uses a combination of computer vision (CV) and deep neural network (DNN) based on methods to locate the eyes and recognize their binary states (open and closed). More precisely, the designed algorithm combines conventional methods, such as HaarCascade face and eyes detectors, and advanced machine learning methods, such as Convolutional Neural Networks (CNN) for measuring and recognising the open and closed states of the eyes. Our algorithm can be executed online and provides an accurate and fast response of the two subsequent steps ; detecting the eyes and distinguishing between their two states (open or closed)

Keywords :

Computer Vision, Deep learning, Eye detection, Eye localisation, Eye state recognition, Face detection.

ملخص

حظيت تطبيقات كشف العين والتعرف على حالاتها المختلفة باهتمام خاص ومتزايد في السنوات الأخيرة وفي العديد من المجالات، مثل تلك القائمة على تقنيات المراقبة بالكمبيوتر والعديد من المجالات الأخرى؛ منها التفاعل بين الإنسان والآلة (HMI)، والكشف عن حالة السائق، وتحليل تعابير الوجه والتعرف عليها، إلخ. مع ذلك، وعلى الرغم من التقدم المحقق في هذه المجالات، والأجهزة الموجودة في السوق، فإن القدرة على توفير اجابة صحيحة من خلال هذه الأنظمة مهمة شاقة، بسبب العديد من الصعوبات عند القيام بهذه العملية في العالم الحقيقي، منها: تغير الإضاءة، الصور الصاخبة، الانسداد وجودة الصورة واختلاف لون البشرة عند الاشخاص، إلخ. للتغلب على كل هذه الصعوبات، يستخدم هذا العمل مجموعة من تقنيات المراقبة عن طريق الكمبيوتر (CV) والتعلم العميق لضمان الكشف الدقيق عن العيون للتعرف على حالاتها المفتوحة والمغلقة. ولأكثر دقة نجمع بين الخوارزمية القائمة الأساليب التقليدية والحديثة لاكتشاف منطقة الوجه، ثم تحديد منطقة العين باستخدامHaarCascade، والتعلم العميق بناءً على الشبكات العصبية المتكرر(CNN) ، من أجل التعرف على حالات العيون (مفتوحة او مغلقة). يمكن تنفيذ الخوارزمية الخاصة بنا عبر الإنترنت باستخدام كاميرا ويب بسيطة، مع توفير اجابة دقيقة وسريعة للخطوات المتتالية المختلفة، وهي تحديد الوجه، تحديد العين، وأخيرًا التعرف على حالات العين (مفتوحة او مغلقة).

الكلمات المفتاحية:

المتابعة عن طريق الكمبيوتر، التعلم العميق، تحديد العين، تحديد موقع العين، التعرف على حالة العين، تحديد الوجه.

Liste des figures

Listes des tableaux

Listes des équations

Liste d'abréviations

VA	Véhicule Autonome.
IHM	Interaction Homme Machine.
CNN	Convolutional Central Network.
PERCLOSE	Percentage of Eye Closure.
EAR	Eyes Aspect Ratio.
IA	Intelligence Artificial.
ROI	Region Of Interest.
ReLU	Rectified Linear Unit.
RNA	Réseaux de Neurones Artificiel.
PMC	Perceptron Multicouches.
DL	Deep Learning.
FC	Fully Connected.
Dlib	Digital Library.
IDE	Integrated Drive Electronics.
OpenCV	Open Source Computer Vision library.
NumPy	Numeric Python.
SciPy	Scientific Python.
XML	Extensible Markup Languange.
MIT	Institut de Technologie du Massachusetts.
LBP	Local Binary Patterns.
PS	Structure Picturale.

SVM	Support Vector Machine.

Table de Matière

Introduction générale

La reconnaissance de l'état des yeux est l'une des principales étapes de nombreux systèmes de traitement d'images en vision par ordinateur. Au cours des deux dernières décennies, les domaines du traitement d'images, de la reconnaissance des formes, de la détection de la fatigue ont connu une croissance spectaculaire, ce qui a conduit à l'introduction des techniques innovantes de détection de l'état des yeux. En effet, ces techniques ont de vastes applications dans le monde réel, notamment la phase de détection de l'état du conducteur pour les véhicules autonomes (AV), l'amélioration de l'interaction homme-machine (IHM), l'analyse et la reconnaissance visuelle des émotions faciales, etc. Il existe trois différentes méthodes pour la reconnaissance visuelle des états des yeux ; a) - les méthodes basées sur l'apparence. b) - les méthodes basées sur les caractéristiques .c) - les méthodes basées sur la structure géométrique de l'œil. d)- Les méthodes basées sur l'hybridation ou la combinaison des précédentes méthodes. Dans le travail fournis, nous avons choisi de travailler avec les méthodes basées sur l'apparence visuelle des yeux par leurs flexibilité et efficacité démontrée dans la littérature [1]. La réalisation d'un système de reconnaissance d'état des yeux en temps réel rencontre plusieurs difficultés parmi elles : le changement de l'éclairage, les images bruitées, l'occlusion, la qualité des images, les variations des textures non-rigides de la peau humaine, etc. Cependant, il est nécessaire d'utiliser une méthode de détection d'état des yeux pouvant surmonter ces contraintes en assurant : la fiabilité, le temps du traitement et la précision des résultats. Pour cela, diverses méthodes sont proposées notamment celles utilisant des méthodes d'IA avec des techniques classiques pour obtenir des résultats efficaces.

Les techniques de détection d'état des yeux peuvent être réaliser : en apprentissage approfondie par les CNN, avec le calcul du PERCLOSE et avec le calcul des valeurs du EAR, etc.
Nous allons présenter notre manuscrit en trois chapitres :

- Chapitre 1 : est un état de l'art qui donne un aperçu général sur deux étapes de détection : une étape de détection du visage accompagné de ses différentes méthodes basées sur les techniques de la comparaison des modèles de visage, les caractéristiques invariantes et les apparences, au niveau de cette dernière on a illustré la méthode de Viola et Jones. Et une étape de détection des yeux avec les méthodes basées sur l'apprentissage du modèle statistique d'apparence afin d'analyser leurs états.
- Chapitre 2 : est consacré pour les réseaux de neurones artificiels où on a défini le neurone artificiel, son fonctionnement et les types d'apprentissage. De plus, la méthode d'apprentissage profond et le concept des CNN pour reconnaitre l'état des yeux.
- Chapitre 3 : se présente sur deux parties : une partie qui décrit la conception de notre système en expliquant le concept des Haar Cascade, des CNN et les techniques utilisées pour analyser l'état des yeux et l'autre partie comporte les librairies adoptées dans notre système ainsi que des détails sur l'implémentation suivie d'une discussion sur les résultats obtenus.

Chapitre 1 : Etat de l’art

1 Introduction

Les yeux sont des éléments interactifs du visage pouvant refléter des signaux de communication interpersonnelle qui permettent de déduire l’état émotionnel de la personne ou son degré de conscience. Possédant des caractéristiques physiques, photométriques et des mouvements uniques, cela offre la possibilité de différencier les yeux facilement des autres organes du visage tels que : la bouche, le nez, les oreilles. Analyser l’état des yeux peut être bénéfique dans d’autres champs d’intérêts comme dans la médecine, la sécurité, l’éducation. Dans le contexte de la médecine, l’œil peut prévenir certaines maladies et identifier quelques processus mentaux. En sécurité, les yeux révèlent le degré de crédibilité de la personne. En éducation, les pupilles des auditeurs informent le professeur de la nécessité d’ajuster le message d’instruction.

De nombreuses applications de la reconnaissance de l’état des yeux basées sur la vision par ordinateur adoptant le domaine du traitement des images et des vidéos, nécessitent une tâche de détection de visage afin d’extraire à partir de celle-ci les yeux dont l’étape de la localisation de ces dernières est essentielle pour connaître leurs états. A partir de là, on peut déterminer la détection des yeux comme une opération de trouver approximativement l'œil dans une image capturer en temps réel par exemple, alors que la localisation des yeux estime avec précision la position des yeux [2, 3]. Le principal défi de la détection de visage sur laquelle on détecte et localise les yeux et leurs états (ouverts ou fermés) est de trouver une méthode immunisée le plus possible contre les changements d'illumination, le bruit de l'image, les échelles et les rotations, l’occlusion, etc. Pour cela, nous allons mettre en œuvre à travers ce chapitre quelques méthodes proposées par des chercheurs qui visent à remédier les problèmes susmentionnés.

2 Détection de visage

Pour l’être humain la détection de visage est une tâche triviale, mais elle parait difficile pour les machines qui nécessite une étape d’apprentissage sur un ensemble d’images pour qu'elles puissent extraire les éléments souhaités depuis un environnement complexe. En revanche, plusieurs contraintes s’imposent lors de la réalisation de cette étape par la machine, parmi elles : les variations d'échelle, les orientations, les conditions d'éclairage, les occlusions, etc. La détection des visages joue un rôle primordial dans un large domaine d’applications telles que : la vidéosurveillance, la reconnaissance faciale et la détection de la somnolence du conducteur et d’autres. Pour parvenir à un résultat précis plusieurs méthodes ont été proposées afin d’assurer la robustesse de détection malgré les changements environnementaux, la rapidité pour

effectuer le traitement en temps réel et la simplicité où le code doit être compris avec moins d'instructions.

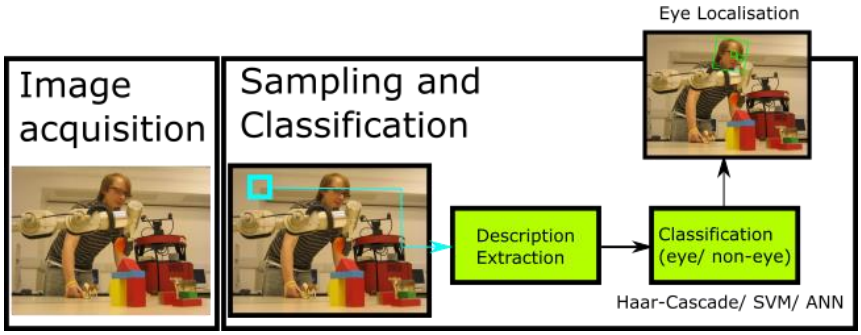

Figure 1: Intéraction Homme-Machine: application des méthodes basées sur l'apparences pour détecter le visage et localiser les yeux.

2.1 Les méthodes de détection de visage

En raison des difficultés rencontrées lors de la détection du visage l'évaluation de la recherche a stagné, ce qui a mené les chercheurs Yang et al. [4] à regrouper des méthodes en quatre catégories :

2.1.1 La méthode basée sur la comparaison des modèles :

Cette méthode permet d'effectuer une comparaison entre des modèles de visage entier ou une partie du visage (bouche, nez, œil) crée, avec le candidat de visage à l'aide d'une fonction de corrélation. Liang et al. [5] proposent une méthode qui utilise de plus que le modèle de visage le modèle des yeux. En raison de l'importance des yeux dans la forme du visage. L'inconvénient de cette méthode et la précision de la détection qui peut être affectée par des différents modèles de visages. De surcroit, cette méthode peut être coûteuse en termes de calcul.

2.1.2 La méthode basée sur les caractéristiques invariantes :

Cette méthode développe des algorithmes qui utilisent les caractéristiques invariantes d'un visage quel que soit la pose, l'orientation du visage, l'angle de prise de vue, la condition d'éclairage et les employées pour localiser les visages. On trouve ainsi des techniques basées sur la texture [6] et la signature de couleur de la peau [7]. Et d'autres basées sur les caractéristiques de visage qui consiste à localiser les cinq caractéristiques (deux yeux, deux narines, et la jonction nez/lèvre) pour décrire un visage typique développé par De Silva et al. [8]

2.1.3 La méthode basée sur les connaissances :

Cette méthode se basent sur la définition des règles pour encoder la connaissance humaine de ce qui constitue un visage typiquement humain. Chiang et al. [9] mesure les positions relatives de ces caractéristiques afin qu'elles servent à la classification de visage et non-visage. Cependant, construire une règle exacte qui contient seulement une classe de visage est difficile, dans le cas où la définition est trop détaillée certains visages ne vérifient pas toutes les règles donc ils ne seront pas détectés, tandis que lorsque les règles sont trop générales, beaucoup de de faux positives seront générer.

2.1.4 La méthode basée sur l'apparence :

Cette méthode applique généralement des techniques d'apprentissage automatique, ou des modèles sont appris à partir d'un ensemble d'images représentant des différents visages. Ces méthodes ont montré de bons résultats vu leurs rapidités d'exécution malgré qu'elles prennent un long temps d'entrainement par rapport aux méthodes présenter dans [10], parmi elles : la méthode basée sur les réseaux de neurones de Rowley et al. [11], et l'algorithme connu de Viola et Jones [12] utilisant les caractéristiques de Haar [13] fonctionnant en temps réel qui sera détaillé ci-dessous.

Algorithme de Viola et Jones :

Cette méthode basée sur l'apparence a été réalisée par les chercheurs Paul Viola et Michael Jones en 2001 [12]. C'est une méthode connue et plus utilisée en particulier pour détecter les visages en temps réel dans une image numérique, et développé aussi pour détecter d'autres objets comme des voitures, des avions, etc. La méthode de Viola et Jones [12] adopte l'apprentissage supervisé qui nécessite pour son entrainement un ensemble d'objets que l'on souhaite détecter afin d'obtenir un classifieur servant à détecter la présence de l'objet dans une image.

-Apprentissage du classifieur :

Le classifieur est entrainer pour le sensibiliser à détecter ce que l'on veut, pour notre cas des visages et des yeux. Pour ce faire, un ensemble d'images contenant des visages et non visages (des yeux et non yeux) de personnes est introduit afin d'avoir un classifieur sous forme d'un fichier XML pourront l'utiliser par la suite pour la détection. Viola et Jones ont entrainé leurs classifieur par l'intermédiaire des ensembles d'images MIT.

-Les caractéristiques :

Les caractéristiques sont des représentations informatives calculée à partie des pixels. Les caractéristiques adopter par Viola et Jones [12] sont les caractéristiques de Haar [13] aussi connues sur le nom de Haar-Like, ce sont des simples opérateurs, peu couteuse en temps de calcule, les fonctionnalités des 4 rectangles sont employés dans cette tâche afin de détecter les éléments souhaités.

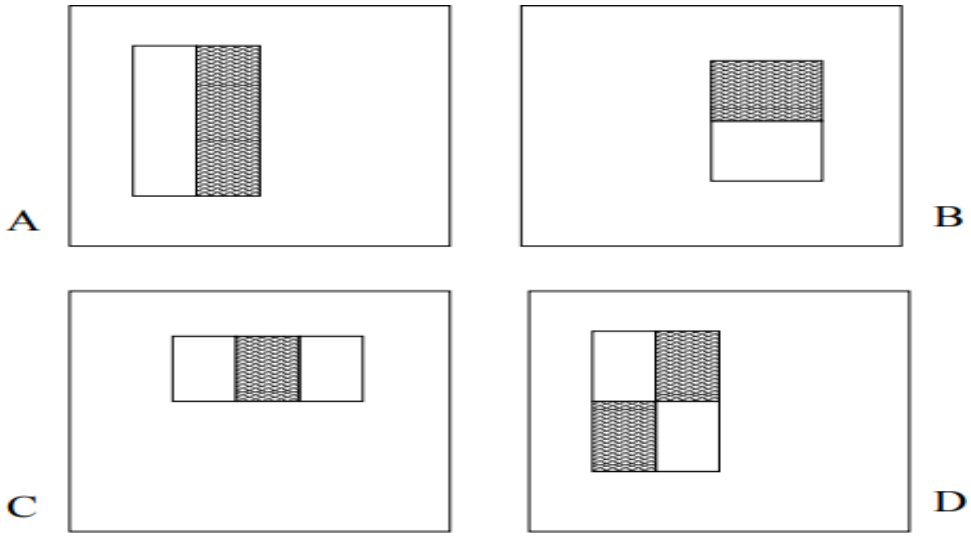

Figure 2: Illustration des caractéristiques de Haar [13] :(a, b) deux rectangles, (c) trois rectangles, (d) quatre rectangles.

-Le calcul des caractéristiques de Haar :

Le calcul des caractéristiques se fait en soustrayant la somme de tous les pixels dans le rectangle blanc de la somme de tous les pixels dans le rectangle noir. Le nombre des caractéristiques rectangulaires de Haar doivent être évaluées à chaque fois, ce qui à mener Viola et Jones [12] à mettre une technique pour réduire le calcul au lieu d'additionner toutes les valeurs des pixels sous les rectangles noirs et blancs à chaque fois. Ils ont introduit le concept d'image intégrale pour trouver la somme de tous les pixels sous un rectangle avec seulement 4 valeurs de coin de l'image intégral.

-Le calcul de l'image intégrale :

Une image intégrale est construite à partir de l'image d'origine de même taille qu'elle, elle contient en chacun de ses points la somme des pixels situés au-dessus et à gauche du pixel courant comme il est montré dans la figure ci-dessous ou la somme des pixels dans la région D est 1+4-(2+3).

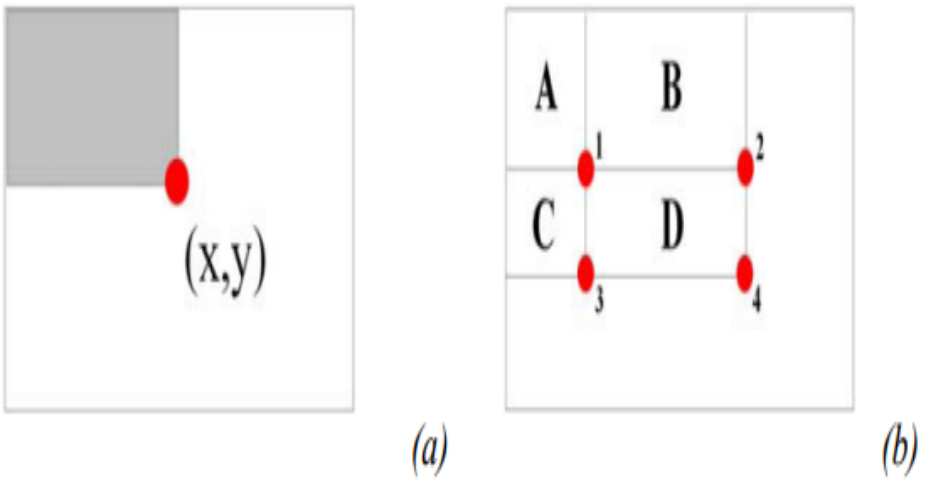

Figure 3 : (a) Image Intégrale (b) la somme des pixels dans la région D est 1+4-(2+3).

La méthode de Viola et Jones est basée sur une approche par la recherche exhaustive sur l'ensemble de l'image, qui teste la présence de l'objet dans une fenêtre à toutes les positions et à plusieurs échelles. Cette approche est cependant extrêmement coûteuse en calcul. L'une des idées-clés de la méthode pour réduire ce coût réside dans l'organisation de l'algorithme de détection en une cascade de classifieurs.

-Les classifieurs en cascades :

Ils sont appliqués séquentiellement, ces classifieurs prennent une décision d'acceptation : la fenêtre contient l'objet, l'opération est achevée et s'oriente vers le classifieur suivant, ou bien le rejet : la fenêtre ne contient pas l'objet et dans ce cas l'opération est rejetée. Selon Viola et Jones [12], la majorité des fenêtres testées négatives (non-visage/ non yeux) peuvent être rejeter rapidement en utilisant des classifieurs plus simples situés au début de la cascade tout en consacrant plus de temps aux entrées positives (visage/ yeux).

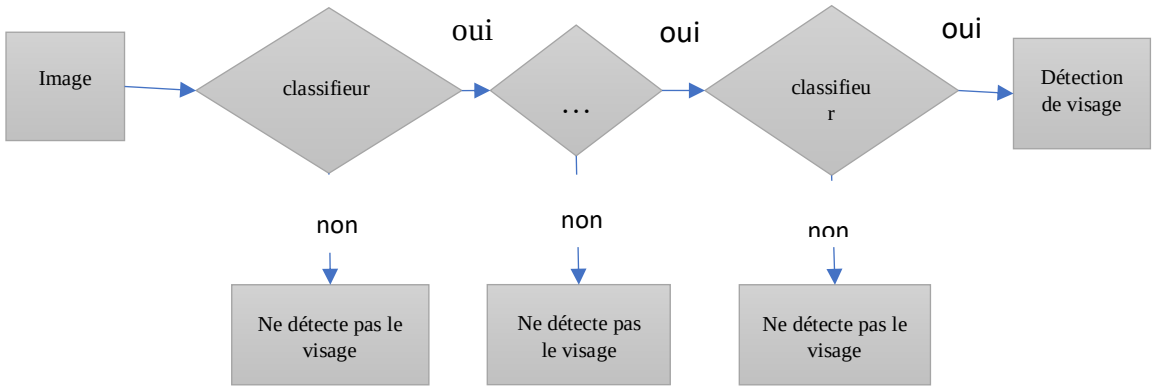

Figure 4: Illustration du classifieur en cascade.

3 Les facteurs de qualité d'une méthode de détection

Des différentes méthodes de détection de visages ont été proposées. Le problème était comment évaluer la performance de ces méthodes. Plusieurs métriques ont été adoptées pour évaluer les algorithmes, tels que le temps d'apprentissage, le temps d'exécution, le nombre d'échantillons nécessaires à l'apprentissage et le rapport entre le taux de détection et les fausses alarmes. Le taux de détection est défini comme le rapport entre le nombre de visages correctement détectés par l'ordinateur et le nombre de visages détectés par l'être humain. Les fausses alarmes dans une image identifiée comme étant un visage est considérée correcte dans la mesure où elle couvre un certain pourcentage de visages présents dans l'image. Généralement, les détecteurs de visage peuvent engendrer deux types d'erreur : les faux négatifs où le visage n'est pas détecté et les faux positifs où une partie de l'image est déclarée comme visage mais elle ne l'est pas. Une évaluation juste doit prendre en considération ces facteurs.

4 Principales difficultés de détection de visage

Le problème de ces méthodes est le manque d’efficacité, si les images en entrée sont de faible luminosité ou si certaines parties d’un visage sont masquées. De plus, le fait qu’un visage soit ombragé rend l’extraction des caractéristiques faciales très difficiles. Cependant, cette technique fait face à beaucoup de difficultés en raison de la très grande variabilité de la forme à détecter (image d’un visage quelconque, d’orientation et de taille quelconque avec un éclairement quelconque), les états d'expression faciales, d'occlusion et d'éclairage changent également l'aspect global des visages, faible luminosité, mais aussi le port de lunettes ou autre modification visible du visage (opération, blessure, pilosité, maquillage, etc.).

5 Détection des yeux :

La détection automatique des yeux occupe une place importante dans plusieurs domaines, ou elle est considérée comme une tâche polyvalente dans l’analyse faciale : la reconnaissance des visages [14], l'estimation du regard [15] et l'analyse du comportement des conducteurs [16]. Lors de la réalisation de cette opération, plusieurs facteurs interviennent, notamment le bruit externe et la diversité d’apparence des yeux. La structure de la diversité d'apparence comprend la forme, la taille, la variation de couleur et le mouvement de l'iris et de la paupière, tandis que le bruit externe comprend les lunettes, les cheveux, l'ombre, la qualité et les conditions d'imagerie, ainsi que les conditions environnementales.

Au cours des dernières décennies, diverses méthodes ont été proposé par des chercheur afin de résoudre les problèmes susmentionnés malgré la grande difficulté de cette tâche.

5.1 La méthode de détection des yeux proposé :

Selon les travaux présenter dans [17] [18], les méthodes de détection des yeux peuvent être classées comme suit :

5.1.1 La méthode basée sur les caractéristiques de l’œil :

C’est une méthode simple et rapide en raison de l’exploitation des caractéristiques intuitives des yeux telles que : la taille, la forme et la variation d'intensité de l'iris et de la paupière [19] le point de réflexion présent dans les images infrarouges de l'œil [20], etc. Dans [19] la fenêtre oculaire est détectée en combinant la variance de l'intensité et l'analyse en composantes indépendantes, tandis que dans [20] la détection des yeux est réalisée en utilisant une lumière infrarouge. Malgré les avantages que présente cette méthode pour la détection des yeux, le taux d’échec est élevé à son niveau en particulier dans les environnements complexes.

5.1.2 La méthode basée sur l’apparence de l’œil :

Cette méthode tente de créer un modèle d'apparence photométrique des yeux qui englobe des informations sur les caractéristiques intuitives (la forme et la taille) et les caractéristiques d'apparence (caractéristiques visuelles). La construction de ce modèle se déroule en trois étapes:

-Prétraitement qui consiste en une réduction du bruit, une correction de l'illumination et une amélioration de la texture.

-L'étape d'extraction et de normalisation des caractéristiques

-La classification les caractéristiques en établissant un modèle d'apprentissage.

Le classifieur en cascade de Viola et Jones [12] est largement utilisé pour la détection des yeux, Kroon et al. [21] ont utilisé la fonction LBP multi-échelle pour localiser l'œil dans des images faciales. Dans [22], les yeux sont localisés en utilisant la transformée de Log-Gabor et l'image intégrale. Cette méthode est capable de fournir des résultats acceptables pour la détection et la localisation de l'œil dans des conditions restreintes en temps réduit.

5.1.3 La méthode basée sur l’information spatiale :

L'approche basée sur l'exploitation de l'information spatiale combine à la fois l'apparence et les caractéristiques topologiques spatiales qui caractérisent les yeux de différentes manières. Ce type de modèle basé sur les caractéristiques structurelles est moins affecté par les changements environnementaux car le modèle est construit sur le contexte du visage. Liang et al. [23] ont proposé une méthode de localisation précise des points caractéristiques du visage en utilisant la recherche discriminante basée sur les composants. Dans [24], les auteurs proposent un modèle de structure picturale (PS) amélioré pour la localisation précise des yeux.

6 L’état des yeux :

L’identification d’état des yeux (ouverts/fermés) est employée dans divers domaines d’application, notamment dans la détection de somnolence chez le conducteur et la reconnaissance des émotions. Plusieurs méthodes sont conçues pour réaliser cette opération, parmi elles celles qu’on va présenter ci-dessous basant sur les apparences et les caractéristiques.

6.1 Les méthodes de la reconnaissance d’état des yeux :

6.1.1 Les méthodes basées sur les caractéristiques :

L'extraction des caractéristiques géométriques de l'œil, telles que l'iris visible et la forme elliptique des paupières présente la base de cette méthode afin de distinguer si l’œil est ouvert ou fermé [25]. De surcroît, les variations de la distribution d’intensité [26] permettent aussi d’établir cette opération par la présence ou l’absence de l’iris et de la région blanche de l’œil dans une image, quel que soit la localisation où le changement environnemental, des courbes représentant ainsi des formes différentes entre les yeux ouverts et fermés étant produites par la projection horizontale et verticale appliquées sur la région des yeux pour déterminer le degré de fermeture des yeux et aligner la région détectée.

6.1.2 Les méthodes basées sur les apparences :

Ces méthodes sont capables à gérer les différentes variations que peut subir les yeux, ainsi que le traitement des images à faible qualité. Ceci la rend plus fiable que d’autres méthodes, elle comprend deux étapes : l’extraction des caractéristiques visuelles à partir de l’aspect photométrique des yeux et leurs classifications.

7 Reconnaissance d’état des yeux :

Dans ce contexte, plusieurs travaux ont été menés pour déterminer l’état des yeux, on cite :

- Gonzalez-Ortega et al. [27] ont développé une approche basée sur la vision en temps réel dans le domaine de la sécurité des conducteurs, en appliquant : une approche hybride pour la détection de l’état des yeux qui combine l’apparence et la forme des caractéristiques des yeux, une fonction de projection pour reconnaitre l’état de l’œil (ouvert/ fermé) et les classifieurs : SVM [28] ainsi que PMC [29] interconnectés entre eux pour interpréter l’état. Cet algorithme a obtenu des bons résultats mais il peut échouer dans le cas de faible résolution, flou et lumière inégale.

- Cui Xu et al. [30] ont proposé une méthode de détection d’état des yeux, en la considérant comme un problème de classification binaire où les yeux sont classés dans l’une des catégories : ouverts/ fermés. Cela est établit en utilisant les histogrammes (LBP) ainsi qu’un classifieur basé sur AdaBoost pour localiser les yeux et reconnaître leurs états.

- Hashem Kalbkhani1 et al. [31], ont proposé un algorithme qui estime l'état ouvert ou fermé des yeux dans les images colorées. Leur cadre consiste à recadrer la région du visage d'abord, puis à ne retenir que 60 % de sa zone supérieure, qui sera ensuite prétraitée. Les yeux sont détectés dans cette région prédéfinie, en utilisant une version améliorée de l'algorithme EyeMap [32]. Ensuite, une estimation fine du centre de la pupille (iris) est définie comme centre de masse des régions oculaires. Dans la dernière étape, l'œil est considéré comme ouvert si le nombre de pixels blancs à l'intérieur du cercle de l'iris dans un sous-espace binaire est supérieur au nombre de pixels noirs. Sinon, les yeux sont fermés. Leur algorithme atteint un taux de reconnaissance élevé et ne nécessite pas des données d'entraînement.

8 Conclusion :

Dans ce chapitre nous avons présenté la détection automatique du visage et des yeux avec leurs différentes méthodes qui ont été proposées, Ces opérations sont importantes pour la reconnaissance de l’état des yeux où l’image contenant le visage simplifie leur localisation, afin de les analyser et identifier leurs états dans des conditions compliquées, il est souhaitable que la méthode utilisée soit simple, robuste et rapide. Dans notre travail nous proposons le détecteur de Viola et Jones [12] basé les HaarCascade [13] pour détecter le visage et les yeux. De plus, l’analyse de l’état de l’œil est établie en utilisant EAR et CNN que nous allons les traiter dans un autre chapitre.

Chapitre 4 : Les réseaux de neurones artificiels

1 Introduction

Les ordinateurs aujourd'hui ont évolué et sont devenus de plus en plus puissants. Cependant, cette puissance ne permet pas de résoudre certains problèmes logiques ou algorithmiques complexes. C'est là où les algorithmes basés sur l'intelligence artificielle (IA) interviennent afin de régler ces dilemmes qui ne peuvent pas être résolus avec des algorithmes classiques ou qui sont très difficiles à programmer. IA est une technique informatique qui permet à la machine de reproduire des fonctions que seul l'être humain peut effectuer du fait de leur complexité, ceci est établit en introduisant des exemples dans le but de l'entrainer à l'aide des neurones artificiels pour apprendre, afin que la machine se permet de trouver un modèle de prédiction, et cela dit qu'elle peut en quelque sorte s'adapter aux différentes situations. C'est ce qu'on appelle l'apprentissage profond, en anglais « Deep Learning ». Elles sont largement utilisées dans la reconnaissance de formes (images ou signaux), le diagnostic, le contrôle qualité, la traduction automatique, de la compréhension du langage, etc.

Dans ce chapitre nous présentons une étude sur les réseaux de neurones, ainsi qu'une introduction à l'apprentissage profond, suivi d'une description des réseaux de neurones convolutifs (CNN).

2 Définition d'un neurone

Un neurone artificiel [33] est une unité de traitement qui reçoit un signal et lors de la phase d'activation, émet un nouveau signal modifié dépendant du précèdent. Il existe deux types de neurones. Les neurones intelligents sont des neurones contenant un poids qui paramètre les opérations effectuées lors de l'activation. Les neurones de transition sont ceux effectuant un calcul ne dépendant d'aucun poids, il s'agit alors de neurones modifiant un signal suivant une loi constante. Le premier modèle du neurone artificiel a été proposé par le neuropsychiatre MCCULLOCH et l'informaticien PITTS [34].

3 Fonction d'activation

Les fonctions d'activation sont des éléments importants dans les réseaux de neurone, s'occupant de décider si l'information reçu est pertinente pour l'information entrée ou pas (sera ignorée).

Un neurone artificiel lors de la phase d'activation se présente comme ceci :

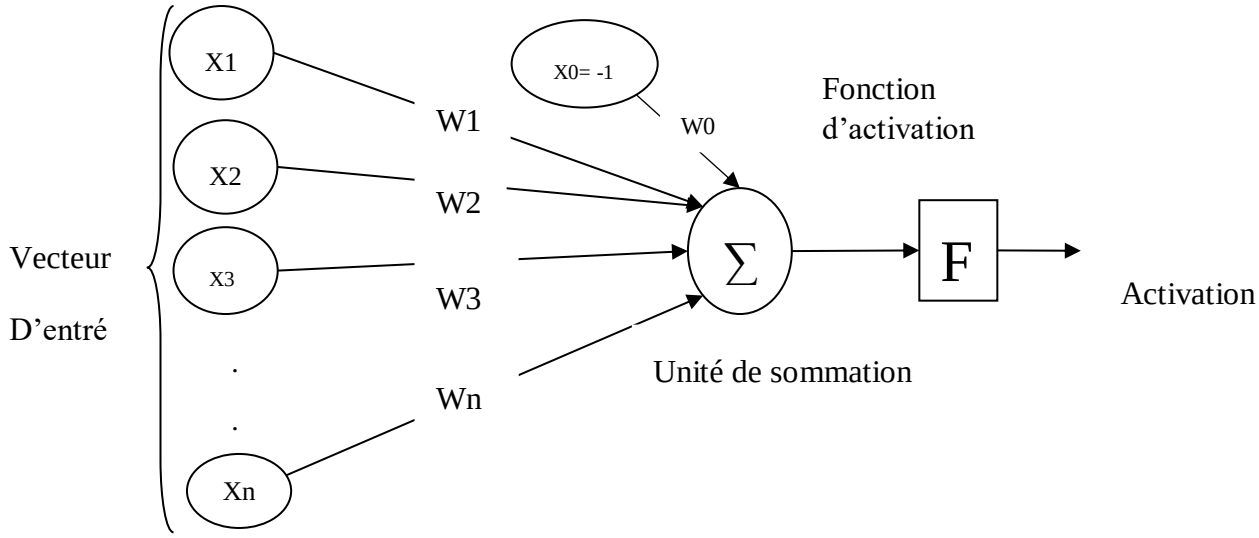

Figure 5: Neurone artificiel de MCCULLOCH et PITTS [34]

(X1, X2, X3…Xn) sont les entrées du neurone d'entrée (les signaux qui lui parvient).

(W1, W2, W3…Wn) sont les poids associés à chaque connexion.

X0 : est le seuil d'activation appelé le biais.

F : est une fonction d'activation binaire :

Si la valeur d'activation est supérieure à un seuil fixe b, alors le neurone s'activera, et passera l'information au neurone qui suit, sinon ce ne sera pas le cas.

A partir de ce modèle divers modèles ont été définit avec d'autres fonctions d'activation, plusieurs On site quelques-unes : la sigmoïde et la fonction ReLU.

4.1 La fonction sigmoïde :

La fonction sigmoïde S est une fonction continue, sa sortie est définie dans l'intervalle [0,1] (Figure 5). L'avantage principale est que sa dérivée existe en tout point. Elle est employée en général dans le perceptron multicouche.

Équation 1 : Fonction d'activation sigmoide.

$$S = \frac{1}{(1 + e^{w^T x})} \quad (1)$$

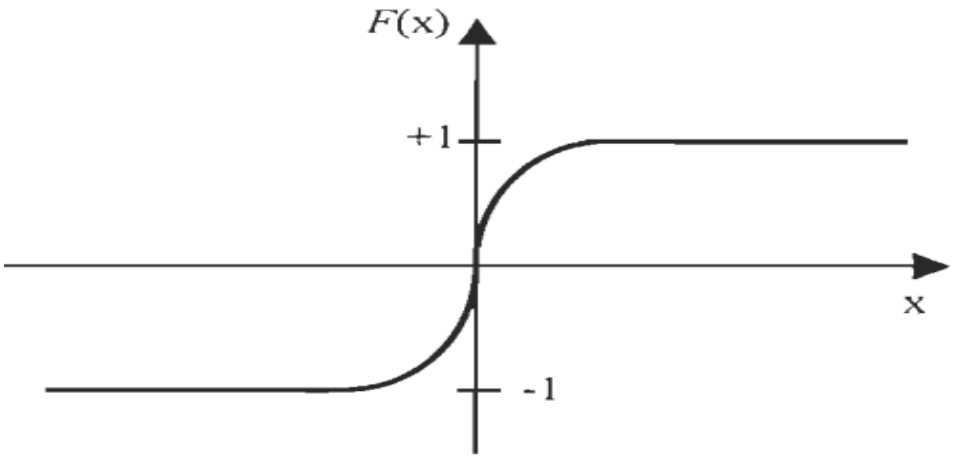

Figure 6 : Allure de la fonction d'activation sigmoïde.

3.2 La fonction ReLU

La fonction d'activation ReLU remplace les valeurs négatives reçues en entrées par des zéros.

ReLU(x)=max (0, x)

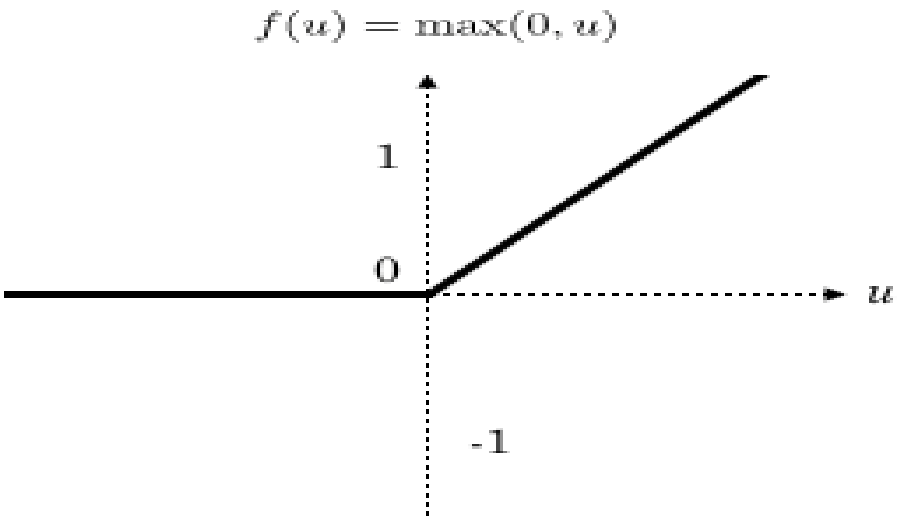

Figure 7 : Allure de la fonction ReLU.

4 Modèle formel (artificiel)

Un réseau de neurone est un ensemble de neurones inspiré du fonctionnement des neurones biologiques, interconnectés entre eux virtuellement, et se représente dans l'ordinateur sous forme d'un algorithme. Dans le cas des méthodes traditionnelles, il est nécessaire de passer par plusieurs étapes jusqu'à l'obtention d'un résultat suffisamment précis. Cependant, les réseaux de neurone artificiels sont entraînés sur des données grâce à un mécanisme d'apprentissage ce qui permet d'améliorer progressivement les performances du réseau sans programmation spécifique, le traitement effectuer est stocké sous forme d'un modèle pour qu'il puisse être

utilisé par la suite. Les RNA sont utilisés dans divers applications, en particulier lorsqu'il s'agit d'une programmation difficile et complexe nécessitant des règles avec un algorithme traditionnel. Par exemple, dans la reconnaissance des caractères faciaux tel que les yeux dans une certaine image, les réseaux de neurones artificiels pourraient facilement s'entrainer à les identifier en analysant un ensemble d'images qui ont été étiquetées manuellement comme des images de yeux. En outre, les résultats obtenus vont être utilisés afin d'identifier l'élément concerné dans d'autres images.

4.1 Architecture des réseaux de neurones artificiels

Généralement, les neurones sont organisés en couches fonctionnant en parallèle, effectuant des calculs dont les résultats seront transmis aux neurones avals. L'information numérique entrante au réseau se propage de couche en couche de l'entrée initiale à la dernière couche appelée sortie, associée d'un poids représentant la force de la connexion. Les réseaux de neurones artificiels tentent de résoudre les différents problèmes de la même façon que système nerveux. Le modèle le plus utiliser ces dernières années est le perceptron multicouches (PMC) sur lequel les neurones sont arrangés en couches successives et ayant des connexions avec les neurones des couches qui suit seulement (pas de connexion entre les neurones d'une même couche), donc un neurone peut avoir différentes entrées, ce qui exige d'avoir une fonction d'activation qui détermine l'état interne du neurone.

Les couches composant ce type de réseau sont la couche d'entrée contenant des neurones dites observables qui se contente de captés les données extérieur, la couche de sortie qui donne le résultat obtenu après compilation par le réseau des données entrées, séparée par une ou plusieurs couches intermédiaires (couches cachées) permettant de modéliser des relations non linéaires entre les entrées et la sortie. Dans ce réseau particulier, les informations des neurones envoyées sous forme de signaux à travers des liaisons (en biologie appelées synapses) se déplacent que dans une seule direction, de l'entrée vers la sortie.

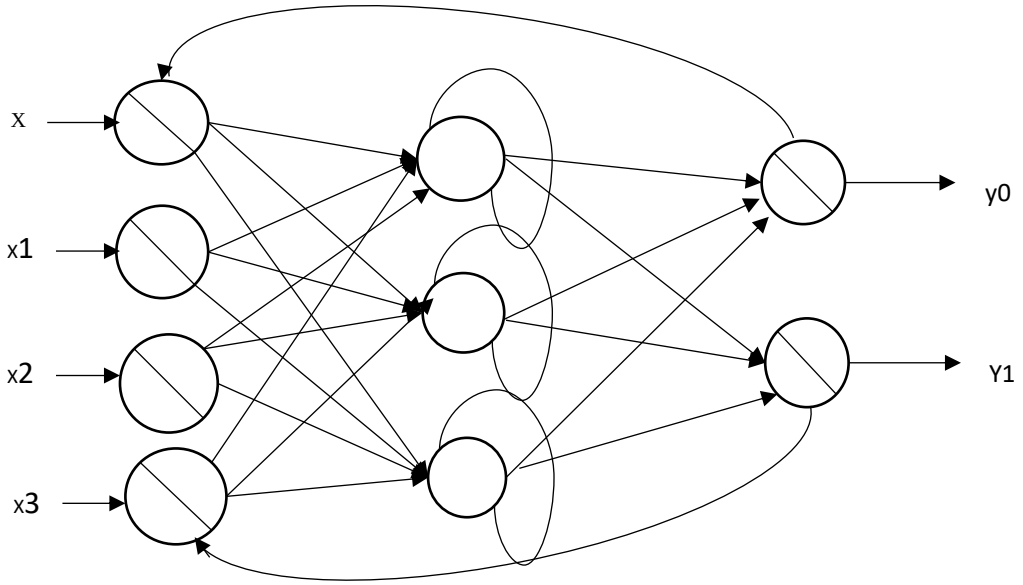

Figure 8: Définition des couches d'un réseau multicouche

4.2 Apprentissage de réseaux de neurones

L'apprentissage est un processus d'adaptation des paramètres d'un système pour remplir au mieux la tâche pour laquelle le réseau est destiné. Ces paramètres sont adaptés avec des différentes méthodes et algorithme [35] afin de déterminer le type d'apprentissage. Il existe plusieurs types d'apprentissage, les plus adoptées sont l'apprentissage supervisé et l'apprentissage non supervisé.

4.2.1 L'apprentissage supervisé

Un superviseur, ou professeur, fournit au réseau des couples d'entrées/sorties, il fait apprendre au réseau l'ensemble de ces couples. Le réseau doit ajuster ses poids de façon à réduire l'écart entre la réponse du réseau et la sortie désirée, L'algorithme le plus utilisé est celui de la rétro propagation de l'erreur en comparant pour chacun d'entre eux la sortie effective du réseau et la sortie désirée. L'apprentissage est terminé lorsque tous les couples entrées-sorties sont reconnus par le réseau.

Ce type d'apprentissage se retrouve dans le perceptron multicouche.

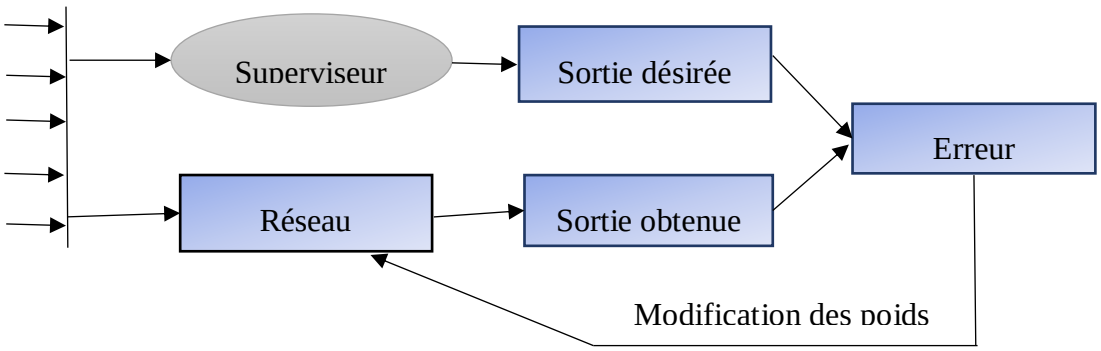

Figure 9: Apprentissage supervisé

4.2.2 Apprentissage non supervisé

L'apprentissage est qualifié non-supervisé lorsque seules les valeurs d'entrée sont disponibles Cet apprentissage consiste à détecter automatiquement des régularités au niveau des valeurs présentées au réseau et à modifier les poids de connexions pour que ces valeurs aient les mêmes caractéristiques de régularité afin de produire des valeurs de sortie proches des valeurs d'entrée. Autoadaptation du réseau. Ce comportement est connu sous le nom "auto organisation" [36].

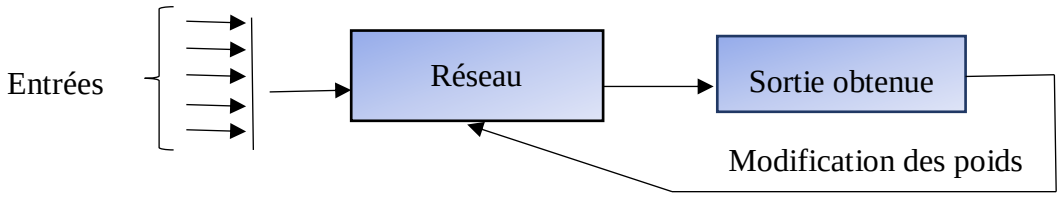

Figure 10: Apprentissage non supervisé

4.3 Apprentissage profond

L'apprentissage profond en anglais « Deep Learning » est un sous domaine de l'intelligence artificielle, la machine est capable d'apprendre par elle-même en s'entraînant sur des ensembles de données labellisées, obtenant ainsi un modèle informatique qui effectue des tâches de classification directement à partir d'images, de textes ou d'audio. Les méthodes d'apprentissage profond réalisent ces opérations avec des algorithmes qui utilisent des concepts mathématiques se basant sur des transformations non linéaires. L'apprentissage profond s'appuie sur le concept des réseaux de neurones artificiels expliquer précédemment, dans lequel les résultats d'une première couche de neurones servent d'entrée aux calculs d'une deuxième couche et ainsi de suite. Les progrès de l'apprentissage profond ont été possibles notamment grâce à l'augmentation de la puissance des ordinateurs et au développement de grandes bases de données. Diverses structures d'apprentissage profond, y compris les CNN sont largement utilisées dans différents domaines, y compris la reconnaissance de la parole, la traduction automatique, le filtrage des réseaux sociaux et autres.

4.3.1 Les réseaux de neurones convolutionnels

Les réseaux de neurones convolutionnels CNN inspirées du perceptron multicouche, sont efficaces dans plusieurs domaines parmi eux la reconnaissance d'état des yeux, la classification des images et l'identification des objets. Les réseaux CNN diffèrent des autres types de réseaux neuronaux artificiels qui se base sur l'intégralité du domaine problématique, ils sont utilisés dans l'apprentissage en profondeur non supervisé, alors que les CNN met en place une certaine architecture qui convient à traiter un type de données spécifiques en exploitant l'ensembles des images d'entrées, ce qui les rends plus performant.

Les réseaux de neurones convolutionnels sont à ce jour les modèles les plus performants pour classer des images. Ils comportent deux parties bien distinctes :

-En entrée, une image au niveau de gris est fournie sous la forme d'une matrice de pixels à deux dimensions. La couleur peut se représentée par une troisième dimension, de profondeur 3 pour représenter les couleurs fondamentales [Rouge, Vert, Bleu].

-La première partie d'un CNN est la partie convolutive fonctionnant comme un extracteur de caractéristiques des images. L'image est passée à travers une succession de filtres (appelé noyaux de convolution), formant de nouvelles images appelées cartes de convolutions. Certains filtres intermédiaires réduisent la résolution de l'image par une opération de maximum local. En fin, les cartes de convolutions sont mises à plat et concaténées en un vecteur de caractéristiques, appelé code CNN [37].

-Ce code CNN est branché par la suite avec l'entrée d'une deuxième partie, constituée de couches entièrement connectées PMC. Le rôle de cette partie est de combiner les caractéristiques du code CNN pour classer l'image.

-La sortie est une dernière couche comportant un neurone par catégorie. Les valeurs numériques obtenues sont généralement normalisées entre 0 et 1, afin de produire une distribution de probabilité sur les catégories.

L'architecture CNN est formée d'un empilement de couches de traitement indépendantes (figure 10) :

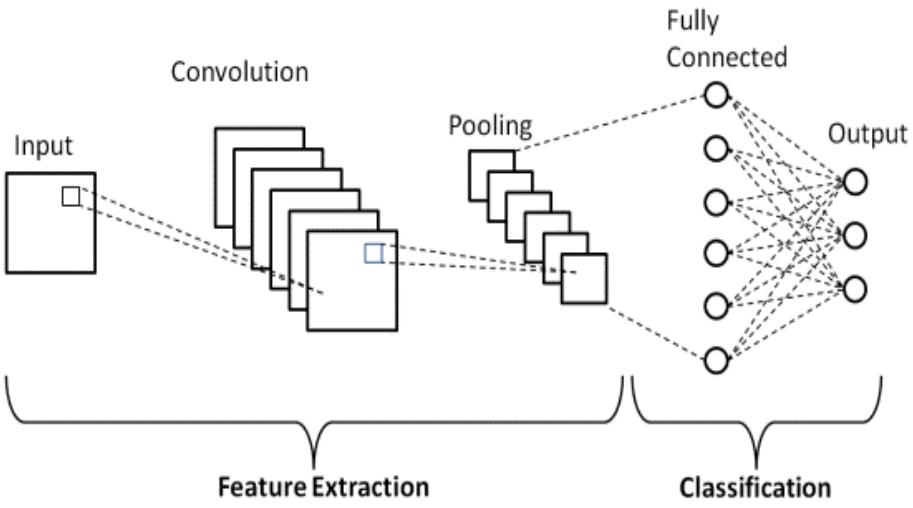

Figure 11: Schéma représentant l'architecture d'un CNN.[37]

•La couche de convolution (CONV), traite les données entrées.

• La couche de Pooling (POOL), compresse l'information en réduisant la taille de l'image intermédiaire (souvent par sous-échantillonnage).

• Fonction d'activation non-linéaire ReLU.

• La couche entièrement connectée (FC), est une couche de type perceptron.

• La couche de perte (LOSS).

4.3.1.1 Couche convolutif :

Les réseaux de neurones convolutionnels sont constitués d'une couche convolutif en tant que première couche du réseau. Le filtre appliqué sur l'image d'entrée sous forme de fenêtre (matrice), glisse sur cette dernière de gauche à droite et de haut en bas tout en effectuant une multiplication de chaque pixel par la valeur de la matrice, Les paramètres pris en compte lors de cette partie sont la profondeur, le pas et la marge.

Profondeur de la couche : désigne le nombre de neurones associées aux entrées du réseau ou appelé nombre de noyaux de convolution.

Le pas : se charge de contrôler le chevauchement des valeurs entrées, ce chevauchement est obtenu quand le pas est petit. Par conséquent, le volume de sortie sera grand.

La marge (à 0) ou (zero padding) : permet de contrôler la dimension spatiale du volume de sortie, il est parfois recommandé de mettre la frontière du volume de l'entrée à zéro. La taille de ce 'zero-padding' est le troisième hyper paramètre.

4.3.1.2 La fonction d'activation non linéaire ReLU :

ReLU est une fonction mathématique utilisée entre les couches des réseaux de neurones à convolution afin d'améliorer l'efficacité du traitement vu sa rapidité de calcule et sa meilleure performance. Comme mentionne auparavant, cette fonction force les neurones ayant des valeurs négatives à retourner des valeurs nulles.

4.3.1.3 La couche de Pooling :

Un autre concept important des CNN est le Pooling, ce qui est une forme de sous échantillonnage de l'image. L'image d'entrée est découpée en une série de rectangles de n pixels. Chaque rectangle peut être vu comme une tuile. Le signal en sortie de tuile est défini en fonction des valeurs prises par les différents pixels de la tuile. Le Pooling réduit la taille spatiale d'une image intermédiaire, réduisant ainsi la quantité de paramètres et de calcul dans le réseau. Il est donc fréquent d'insérer périodiquement une couche de Pooling entre deux couches convolutives successives d'une architecture CNN pour contrôler le sur-apprentissage (overfitting). La couche de Pooling fonctionne indépendamment sur chaque tranche de profondeur de l'entrée et la redimensionne uniquement au niveau de la surface. La forme la plus courante est une couche de mise en commun avec des tuiles de taille 2x2 (largeur/hauteur) et comme valeur de sortie la valeur maximale en entrée. On parle dans ce cas de « Max-Pool 2x2 ».

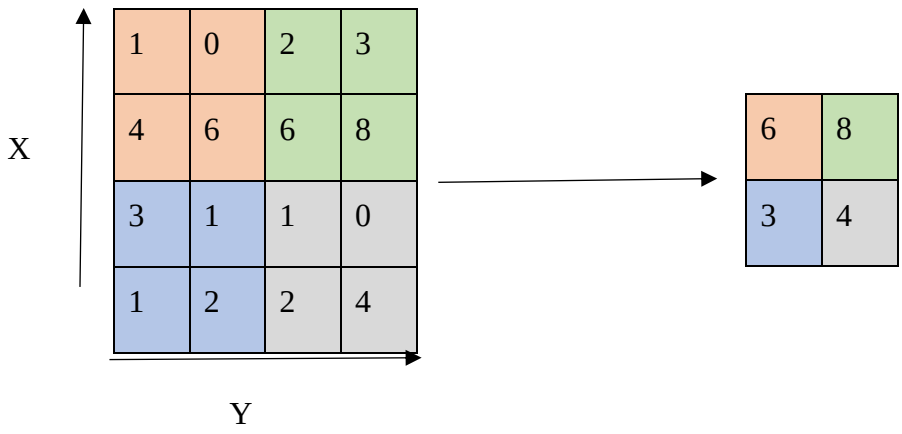

Figure 12: Pooling avec un filtre 2x2 et un pas de 2 [37]

Il existe plusieurs types de Pooling :

- Le « max Pooling » qui revient à prendre la valeur maximale de la sélection. C'est le type le plus utilisé car il est rapide à calculer (immédiat), et permet de simplifier efficacement l'image.

- Le « mean Pooling » (ou average Pooling), soit la moyenne des pixels de la sélection : on calcule la somme de toutes les valeurs et on divise par le nombre de valeurs. On obtient ainsi une valeur intermédiaire pour représenter ce lot de pixels.

- Le « sum Pooling » c'est la moyenne sans avoir divisé par le nombre de valeurs (on ne calcule que leur somme) [37].

Il est possible d'utiliser d'autres fonctions de Pooling que le maximum. On peut utiliser le « average Pooling » ou la sortie est la moyenne des valeurs du patch d'entrée. Cette fonction était souvent utilisée mais il s'est avéré que le max-Pooling était plus efficace car celui-ci augmente plus significativement l'importance des activations fortes.

Ce type de couche est souvent placé entre deux couches de convolution : elle reçoit en entrée plusieurs cartes d'activations, et applique à chacune d'entre elles l'opération de Pooling. Cette dernière consiste à réduire la taille des images, tout en préservant leurs caractéristiques essentielles, pour cela, l'image est découpée en cellules régulières en gardant au sein de chaque cellule la valeur maximale. En sortie le même nombre de carte d'activation qu'en entrée est obtenu, mais celles-ci sont bien plus petites. Cette opération diminue le nombre de paramètres et de calculs dans le réseau et améliore ainsi l'efficacité du réseau. Les valeurs maximales sont repérées de manière moins exacte dans les cartes d'activations obtenues après Pooling que dans celles reçues en entrée. Après la couche de Pooling la position des caractéristiques n'est plus importante dans le réseau : le changement de la position des caractéristiques ne devrait pas provoquer un problème dans la classification de l'image.

4.3.1.4 La couche entièrement connectée :

Cette couche est placée en fin d'architecture des CNN. Elle reçoit en entrée à partir d'un tableau (matrice) un vecteur obtenu par l'application d'une combinaison linéaire puis une fonction d'activation aux valeurs reçues, produisant ainsi un nouveau vecteur en sortie de taille N, où N est le nombre de classes, ces classes sont connectée à tous les neurones en aval. Chaque élément du vecteur présente une probabilité acquise en multipliant les éléments du tableau entrée par un poids, ce qui rend l'image dépendante d'une classe. Pour le cas des CNN, ces poids sont appris lors de la phase d'entraînement, par la méthode de rétro-propagation du gradient. Cela permet d'établir la phase de la classification des images et aussi détermine le lien entre la position des caractéristiques dans cette dernière et une classe.

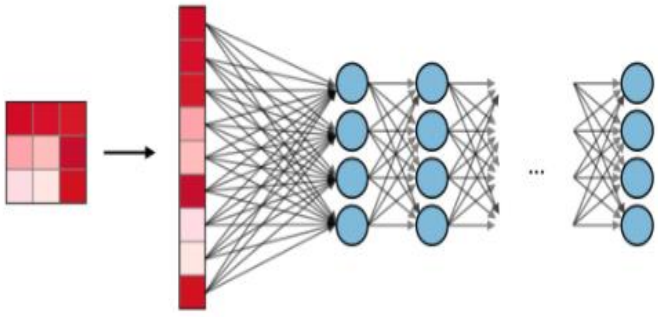

Figure 13: Architecture d'un réseau neuronal a la phase 'fully-connected'.[37]

4.3.1.5 Fonction de perte (LOSS) :

La couche de perte spécifie comment l'entrainement du réseau pénalise l'écart entre le signal prévu et réel. C'est la dernière couche du réseau. Diverses fonctions de perte adaptées à différentes tâches peuvent y être utilisées.

5 Conclusion

Dans ce chapitre nous avons commencé par définir le neurone et citer deux fonctions qui l'active, ensuite on a expliqué le réseau de neurone artificiel ainsi que son fonctionnement ,en passant par son architecture et une illustration sur l'un de ses modèles PMC, suivi de l'apprentissage qui l'adopte qui consiste aux deux types : supervisé et non supervisé, Aussi qu'une vision générale sur l'apprentissage profond, toute en présentant la méthode choisie dans notre travail de recherche qui est le CNN. Le prochain chapitre, traite les détails de la conception, ainsi que la méthode et les outils utilisés pour notre application.

Chapitre 3 : Conception et implémentation générale du système

1 Introduction

Après avoir présenté dans les chapitres précédents une étude théorique sur les différentes phases de détection d'état des yeux d'un individu, nous abordons dans la première partie de ce chapitre l'implémentation de notre système, le concept de la méthode de Viola et Jones [12] et celle de réseau de neurones convolutionnels, ainsi que les techniques utilisés et les étapes de la réalisation d'un système permettant d'analyser l'état des yeux. Dans la deuxième partie, nous allons mettre en évidence les différentes étapes de notre système proposé pour la détection d'état des yeux d'une personne. Dans un premier temps, nous commençons par présenter le langage de programmation adopté dans le développement de notre application, ensuite nous détaillons les structures de données utilisées, et enfin nous présentons nos implémentations réaliser, en expliquant le principe de détection de visage en temps réel avec la méthode de Viola et Jones[12], ainsi la détection des points caractéristiques du visage avec Dlib [38] pour détecter l'état des yeux par le calcul du EAR, et l'intégration d'un modèle entrainer avec les CNN pour plus de précision. La dernière partie concerne les résultats obtenus, et une discussion des résultats est donnée à la fin de ce chapitre.

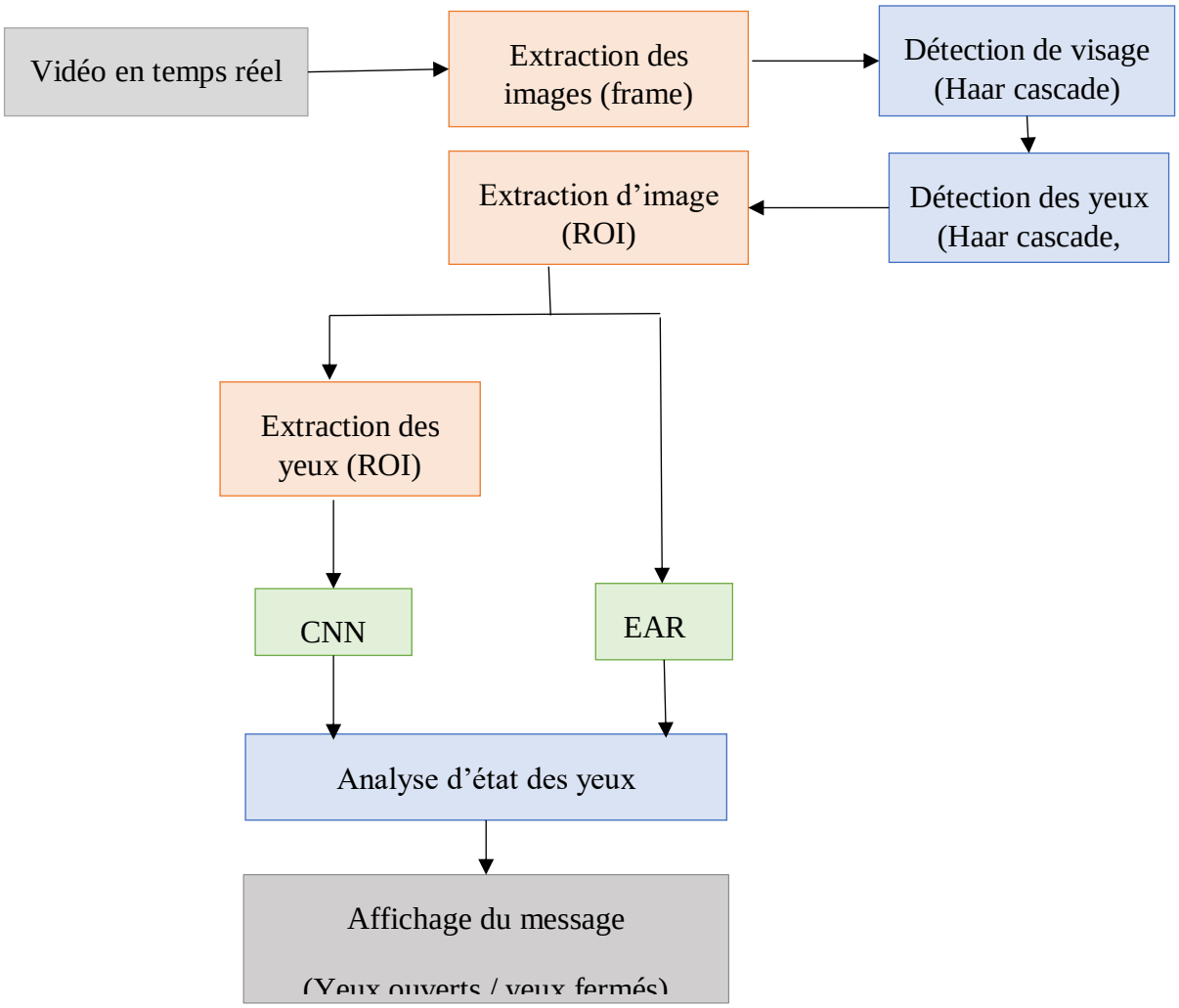

Figure 14 : Conception générale de notre système proposé.

2 Implémentation de notre système

2.1 Outils utilisés

2.1.1 Dispositifs logiciels

Notre configuration matérielle inclut les dispositifs suivants :

CPU	Intel (R) Core (TM) i5-5300U CPU @ 2.30 GHz.
Mémoire	8,00 Go
Système d'exploitation	Windows 10 OS, X64
Caméra	Caméra-Web

Tableau 1 : Dispositifs matériels.

2.1.2 Outils logiciels

Python [40] :

C'est un langage de programmation interprété, multiplateforme et orienté objet, développé par Guido van Rossum avec d'autres contributeurs bénévoles, offrant des outils de haut niveau et une syntaxe simple à utiliser ainsi qu'une très large bibliothèque standard. Ce langage permet aux programmeurs d'exprimer leurs idées en moins de lignes de code sans réduire la lisibilité. Comparant à d'autres langages tels que C / C ++, Python paraît plus lent. Pour cela, Il est possible de l'étendre avec C / C ++, ce qui nous permet d'écrire du code exigeant en calculs intensifs en C / C ++ et de créer un environnement Python utilisable en tant que modules Python. Cela nous donne deux avantages : premièrement, le code est aussi rapide que le code original C / C ++ (puisqu'il s'agit du code C ++ réel travaillant en arrière-plan) et deuxièmement, il est plus facile de coder en Python que C / C ++. OpenCV-Python est une enveloppe Python pour l'implémentation d'origine OpenCV C ++ et nous allons utiliser cette bibliothèque pour réaliser notre travail.

Pycharm [48] :

C'est l'un des environnements de développement intégrés (IDE) les plus célèbres pour Python, développé par l'entreprise tchèque JetBrains, c'est un logiciel multiplateforme fonctionnant sous Windows, Mac OS X et Linux. L'environnement intégré est livré avec la plate-forme pour l'analyse de code, le débogueur graphique, la prise en charge de la programmation Web et des frameworks, etc. il existe deux versions de PyCharm : la version Communautaire (version gratuite, c'est cette version que nous avons utilisé) et la version Professionnelle (version payante et soumise à licence commerciale).

OpenCV [39] :

C'est une bibliothèque graphique libre, initialement développée par Intel, spécialisée dans le traitement d'images en temps réel et prend désormais en charge une multitude d'algorithmes liés à la vision par ordinateur et à l'apprentissage automatique. Cette bibliothèque open source est utilisable avec une grande variété de langages de programmation tels que C++, Python, Java, etc, et peut être exécuté sous différentes platesformes, notamment Windows, Linux, OS X, Android et iOS.OpenCV-Python utilise NumPy [44] « Numeric Python », qu'est une bibliothèque hautement optimisée pour les opérations numériques avec une syntaxe proche de celle MATLAB. Toutes les structures de tableau OpenCV [38] sont converties vers et à partir de tableaux NumPy. Cela facilite également l'intégration avec d'autres bibliothèques utilisant NumPy [44], telles que SciPy [45] « Scientific library ».

Dlib [38] :

C'est une boîte à outils moderne du C++ afin de pouvoir créer des logiciels complexes sur ce dernier et résoudre des problèmes du monde réel, cette librairie contient des algorithmes d'apprentissage automatique. Elle est utilisée dans l'industrie, les universités et dans un large éventail de domaines, notamment la robotique, les appareils embarqués, les téléphones mobiles et les grands environnements informatiques haute performance.

Tensorflow [49] :

C'est une bibliothèque open source permettant de développer et exécuter des applications de l'apprentissage automatique et l'apprentissage profond crée par Google Brain en 2011 sous forme d'un système dédié aux réseaux de neurones qui font partie de l'apprentissage en profondeur. Cette bibliothèque s'agit d'une boite à outils permettant de résoudre des problèmes mathématique complexes en développant des architectures d'apprentissage expérimentales et de les transformer en logiciels. Il regroupe un grand nombre de modèle d'apprentissage profond et automatique. Ces applications sont-elles même des application python, les bibliothèques disponibles sur tensorflow sont écrites en c++.

Keras [43] :

C'est une bibliothèque open source pour la constitution rapide de réseaux de neurones. Ecrit en python avec Tensorflow. Keras fonctionne comme une API (interface de programmation applicative) pour l'accès et la programmation des différents frameworks de l'apprentissage automatique pris en charge par Keras citant par exemple tensorflow. Sa structure est un modèle qui organise les couches, le modèle séquentiel est le plus simple constitué d'un empilement linéaire de couches

Version des logiciels et librairies utilisés dans notre système :

Logiciel/ librairie	Version utilisé
Pycharm	3.8.8
OpenCV	4.5.1
Dlib	19.19.0
Tensorflow	2.5.0
Keras	2.4.3

Tableau 2 : Version des logiciels et librairies utilisés dans notre système

3 Conception générale de notre système proposé

Les expressions faciales diffèrent selon les situations : joie, tristesse, fatigue, etc. Chaque individu exprime son état de plusieurs manières, par exemple la fatigue peut s'exprimer en fermant les yeux pendant quelques secondes ou involontairement. L'objectif de notre travail est d'utiliser deux méthodes : une méthode classique basée sur les classifieurs HaarCascade [13] de Viola et Jones [12] expliquer précédemment, dont on extraira les yeux à partir d'un visage détecté afin d'identifier l'état de l'œil (ouvert ou fermé) en utilisant les CNN, et une autre méthode basée sur la détection des points caractéristiques du visage précisément des yeux par

l'intermédiaire de la librairie Dlib [38] en mesurant EAR [41] pour plus de précision afin de prédire le résultat. Nous allons intégrer ces méthodes à une vidéo en temps réel.

3.1 Méthode de Viola et Jones

La méthode Viola et Jones [12] a été proposée dans divers applications pour détecter des objets comme les voitures et autres, puis elle a été utilisée pour détecter des visages dans une image numérique ou une séquence vidéo. Ensuite cette méthode a eu une implémentation de la bibliothèque OpenCV [39] nommée « détecteur de HaarCascade [13] ». Le point fort de cette méthode est la rapidité de détection, ce qui la rend capable de s'exécuter en temps réel et de répondre aux exigences du traitement vidéo. En revanche, elle présente quelques limites telles que la difficulté de détection simultanée de plusieurs vues du même objet ainsi que les fausses détections.

3.1.1 La conception de la méthode de Viola et Jones

La phase de détection

La détection du visage et des yeux est une phase primordiale pour l'être humain, par contre pour la machine il est nécessaire de passer par différentes étapes afin qu'elle puisse établir ce processus, cela implique une recherche de la présence d'un visage dans une succession d'images capturées par l'intermédiaire d'une caméra et de trouver l'emplacement exact du visage et des yeux de la personne, puis extraire les yeux à partir de ce dernier.

Dans notre système, nous avons besoin d'obtenir ce résultat à partir du modèle de détection de visage. Le modèle que nous souhaitons réaliser, doit recevoir une succession d'image en entrée et donner les sous-images du visage de l'utilisateur en sortie avec une technique assurant la détection du visage en temps réel, afin d'obtenir un résultat satisfaisant. Pour cette raison, nous utilisons les classifieurs de HaarCascade pour détecter le visage, en traitant les séquences vidéo, image par image. Cette étape se décompose en deux tâches à savoir, la détection du visage et des yeux à l'aide de la méthode Viola et Jones [12], l'extraction des traits faciaux souhaités c-à-d. les yeux et le suivi des déplacements du visage dans la scène.

3.1.2 La détection des points caractéristiques dans un visage

L'analyse du visage afin de déterminer l'état d'un individu consiste à extraire des points caractéristiques spécifiques dans ce dernier, et d'éliminer les informations redondantes. Dans notre cas, la détection de ces points est réalisée par l'implémentation de la librairie Dlib [38] utilisé sous le langage Python [40]. Elle est utilisée pour estimer l'emplacement de 68 coordonnées (x, y) qui cartographient les points faciaux sur le visage d'une personne, ces points sont stockés dans un tableau indexé. L'image ci-dessous représente ces derniers qui sont identifiés à partir d'un modèle pré-entraîné.

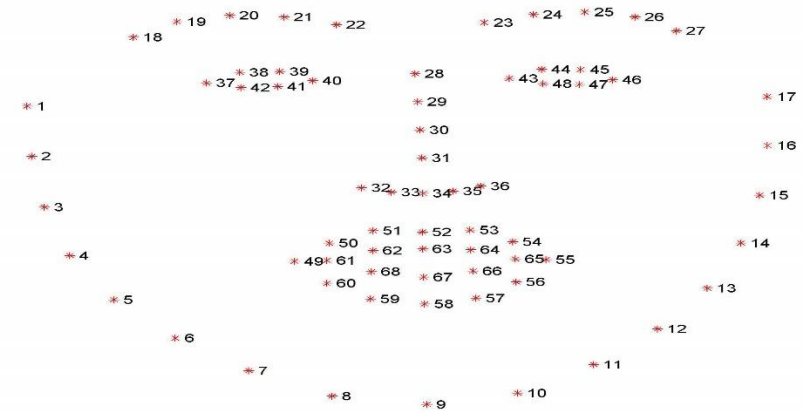

Figure 15 : Les points de repère faciaux selon la bibliothèque Dlib [38]

L'implémentation de la librairie Dlib [38] a pour objectif de trouver les points caractéristiques des yeux afin de calculer la distance euclidienne, pour trouver la distance entre deux points. Pour ce faire, les formules ci-dessous vont être utilisé dans le but de les exploiter par la suite afin de déterminer l'EAR [41].

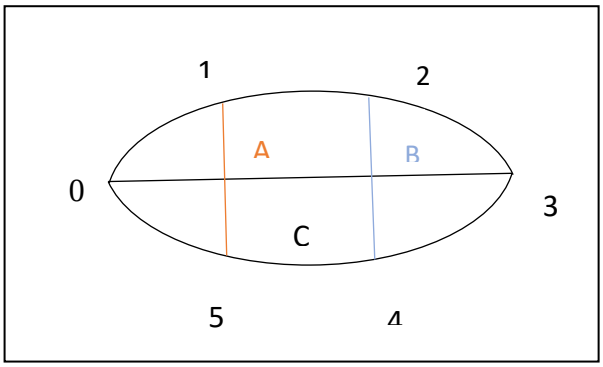

Figure 16 : Région d'un œil représenté par les points caractéristiques

A = d (eye [1]; eye [5])

B = d (eye [2]; eye [4])

C = d (eye [0]; eye [3])

EAR = (A+B)/(2*C)

Tel que «A» et «B» mesurent respectivement la distance verticale de l'œil et «C» calcule les dimensions horizontales de l'œil, d'où :

Calcul du EAR :

Le rapport d'aspect oculaire, en anglais « Eye Aspect Ratio [41] » est défini comme le rapport entre la hauteur et la largeur de l'œil, La valeur de cette mesure va nous permettre de définir et de mesurer l'ouverture et la fermeture de l'œil (plus elle est grande, plus l'œil est ouvert et le contraire est vrai).

3.1.3 Algorithme de détection d'état des yeux

Notre système proposé vise en premier lieu à détecter le visage et les yeux dans des images issues d'une vidéo capturée via une caméra. Cette détection est réalisée à l'aide d'un classifieur de HaarCascade [13] formé à reconnaître les visages humains ainsi que les yeux. De plus, à partir de l'implémentation de la librairie Dlib [38] formés pour estimer les points de repères d'une manière rapide, on détermine l'emplacement des principaux points caractéristiques du visage parmi eux la région des yeux pour notre système. Ces points sont stockés dans un tableau indexé comme nous l'avons précisé avant, ils spécifient les coordonnées des régions entourant chaque trait du visage afin de garantir la précision. Apres l'étape de la détection, on passe à analyser l'état des yeux repérés par les points caractéristiques utilisés, pour cela la forme convexe de l'œil ainsi que le calcul du EAR comme illustré dans la section précédente sont prise en considération.

3.2 Méthode des réseaux de neurones convolutifs

Il est possible d'identifier l'état des yeux d'une personne sur une séquence de vidéo par une autre méthode, cette méthode se concentre sur l'utilisation des réseaux de neurones pour reconnaître les images qui comportent des yeux ouverts ou fermés. Un algorithme est mis en mesure pour distinguer les deux états de manière précise, quel que soit l'angle sous lequel il est capté. Ce processus exige une phase d'apprentissage où il doit être entraîné sur un ensemble d'images contenant des yeux ouverts et fermés, cette étape fait partie de l'apprentissage profond, après cela l'algorithme pourra déduire l'état des yeux de l'utilisateur. Les images utilisées lors de la phase d'apprentissage sont ensuite converties en données et transférées sur le réseau, les RNA assignent des informations aux différents éléments qui vont être rassembler à la fin de ce réseau afin de déduire l'état des yeux.

3.2.1 La conception de la méthode des réseaux de neurones convolutifs

Dans cette partie, on va mettre en avant notre conception concernant cette opération, et montrer comment réaliser notre classifieur afin de créer notre modèle à partir d'une base de données d'images [42]. Pour cela, on va travailler avec le langage de programmation Python en ligne sur Google Colab, en adoptant la librairie Keras [43] ainsi que d'autres techniques dans le but d'améliorer les performances du modèle qui va être exploiter par la suite pour la détermination d'état des yeux de l'utilisateur.

Notre programme est conçu par la conception du classifieur illustrée ci-dessus :

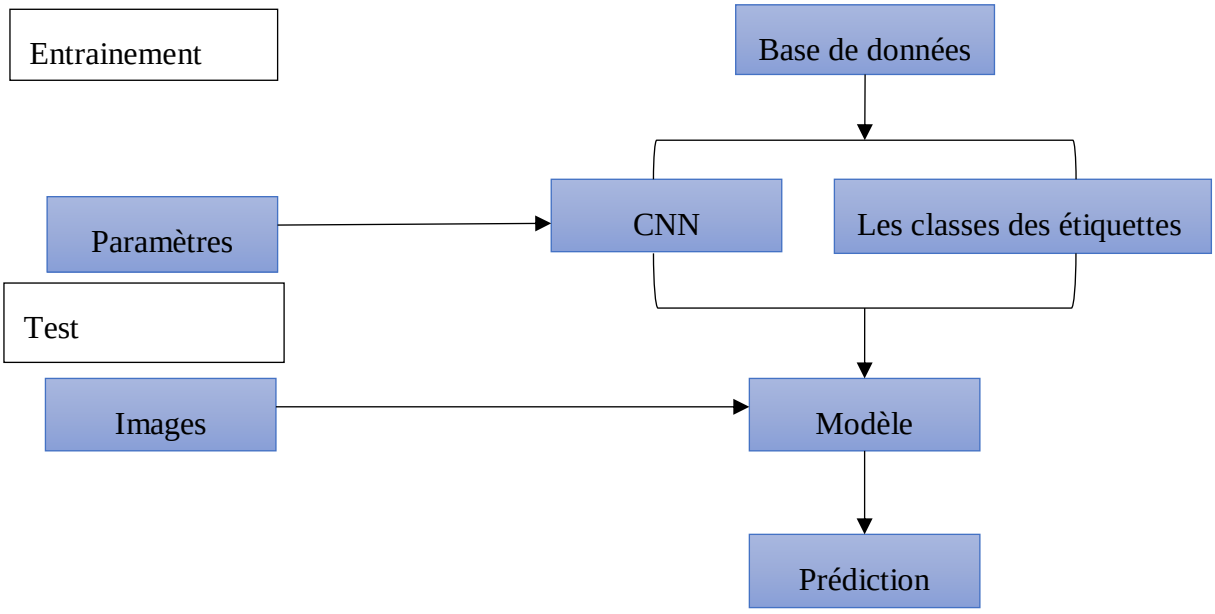

Figure 17 : Conception du classifieur

Le schéma présenté est organisé sous deux grands processus qui englobe notre conception :

-**La partie d'entrainement** : c'est à partir de ce processus qu'on va réaliser une configuration afin de créer notre modèle.

-**La base de données** : pour notre cas on a utilisé une base de données d'images des yeux ouverts et fermés créé par Parasad V Patil, [42] elle est répertoriée en deux classes de 2000 images.

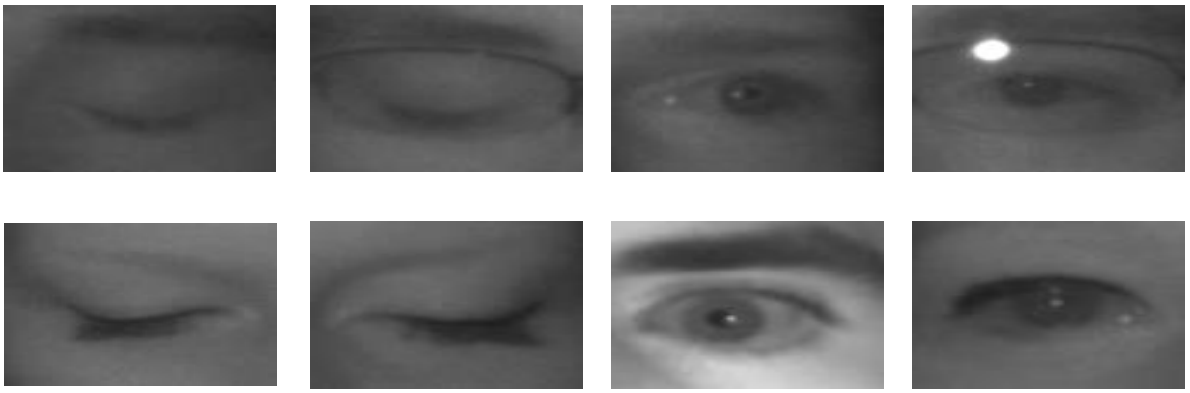

Figure 18: Images de la base de données [42] utilisée.

- **Les classes des étiquettes :** ce fichier va porter les noms des classes de notre base de données, qui sont les classes des yeux ouverts et fermés.

- **Paramètres et CNN :** c'est l'algorithme configuré avec certains paramètres tel que : le nombre d'époques, le nombre de filtre, le nombre de couche… etc., il se charge de créer le réseau de neurones convolutionnels.

L'algorithme CNN paramétrée ainsi que la base de données vont être exécuter pour générer un modèle qu'on va l'enregistrer puis l'utiliser pour la phase de Test.

La partie Test : dans ce processus on trouve :

L'image : désigne un ensemble d'images ou une seule, elle représente l'entrée du Test.

Le modèle : le fichier créé auparavant dans l'étape de l'entrainement.

La prédiction : comme son nom l'indique, il représente le résultat de la sortie du modèle.

3.3 Implémentation basée sur Viola et Jones

3.3.1 Méthodologie de détection de visage :

Dans cette étape, nous détectons la région contenant le visage d'un utilisateur en se concentrant sur l'existence ou non d'un visage dans l'image capturée en temps réel à l'aide d'un détecteur (classifieur). L'algorithme principal utilisé pour ce processus est l'algorithme de Viola et Jones [12] en utilisant la partie cascade de OpenCv [39] qui donne comme résultat une liste de fichiers « .xml » dits classifieurs de HaarCascade. Lorsque nous obtenons une réponse positive pour la détection d'un visage, le processus passe à l'étape suivante, sinon l'algorithme est conçu de manière à capturer l'image jusqu'à ce qu'un indice de visage soit trouvé. Les coordonnées du visage détecté sont passées à une fonction chargée d'englober ce dernier dans un rectangle, tout en traitant les séquences vidéo image par image, seules les structures ou caractéristiques liées au visage sont détectées et tous les autres types d'objets sont libéré sous forme de vidéo. Cette fonction permet de suivre implicitement le mouvement du visage détecté.

3.3.2 Méthodologie de détection des yeux :

Après une détection réussite du visage, l'œil doit être détecté pour la suite du traitement. Dans notre application, l'objectif principal est de déterminer l'état des yeux, pour cela nous avons localiser la région où se trouve l'œil en utilisant la librairie Dlib [38] et Haar cascades de la méthode de Viola et Jones [12]. Nous présentons ci-dessous une figure obtenue par l'intermédiaire de notre programme sur une séquence de vidéo téléchargé, qui montre comment nous avons effectués la détection du visage et des yeux en déterminant les coordonnées du rectangle englobants les yeux et ce après avoir récupéré les coordonnées du visage.

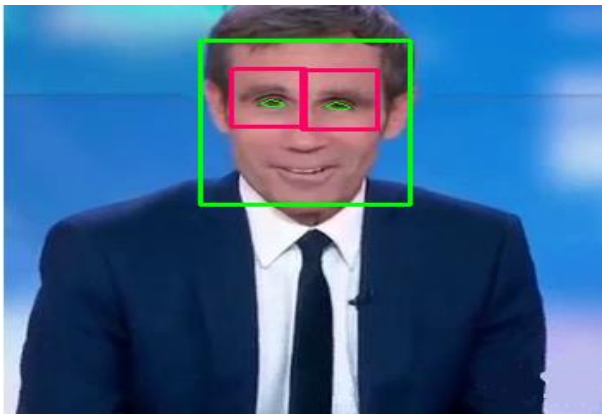

Figure 19 : Détection du visage et des yeux.[44]

3.4 Reconnaissance d'état des yeux

3.4.1 Implémentation basée sur EAR

À ce stade, nous déterminons l'état réel de l'œil, à savoir s'il est ouvert ou fermé. Elle est réalisée par la comparaison de la valeur EAR avec un seuil défini par l'utilisateur. Lorsque l'œil est fermé cela implique que la valeur du EAR est au-dessous du seuil (<0.26), un message s'affiche indiquant cela, et le contraire est vrai, lorsque l'œil est ouvert cela implique que la valeur du EAR est au-dessus du seuil (>0.26). Le rapport EAR est partiellement insensible à la forme de l'œil de la personne et invariant à une mise de rotation dans le plan du visage. Le changement d'état des yeux, le clignement par exemple est effectué par les deux yeux de manière synchrone, d'où le calcul de la moyenne du signal EAR est nécessaire.

3.4.2 Implémentation basée sur les réseaux de neurones convolutionnels

Cette opération est basée sur la méthode précédente. Nous passons à expliquer en détails comment notre modèle de prédiction est créé. Pour ce faire, on la devise en deux étapes :

- **La première étape : La création et l'enregistrement du modèle.**

-Création des tableaux :

On réalise l'enregistrement de deux tableaux en utilisant la librairie Numpy [45], sur lesquels sont stoker les images et leurs étiquettes, tout d'abord nous avons chargé notre base de données [42] 'train' qui contient 4000 images de yeux ouverts/ fermés, ainsi que déterminer des étiquettes (1 pour l'œil ouvert et 0 pour l'œil fermé) pour chaque image en fonction de leurs noms de fichiers afin d'accélère le processus d'apprentissage.

Après cela, nous avons créé une architecture simple du modèle basé sur trois blocs.

-Images en entrée :

Les images utilisées lors des phases d'apprentissage et de test ont été redimensionnées à la taille de 70x70 pixels. Ensuite, ces images sont données en entrée pour alimenter notre réseau.

-Architecture du réseau :

L'architecture est construite sous forme d'un ensemble de couche formant ainsi un bloc pouvant se répéter avec un nombre de filtre qui diffère au niveau de succession de filtres convolutionnels qui a leurs tours s'étend de 32 filtres jusqu'à 512 filtres [46]. Le premier bloc contient une couche de convolution composée de 32 filtres de taille 3x3 accompagnée d'une fonction d'activation non linaire ReLU qui permet au neurone de retourner des valeurs positives, et la méthode de He [47] pour l'initialisation des poids, cette méthode fournit un compromis entre la mise à jour des valeurs des poids durant la phase d'apprentissage et le calcul de la fonction coût-calcule des valeurs de gradient descendant. Suivie de la couche Max-Pooling appliquée pour réduire la taille de l'image. Dans les deux blocs qui suivent se répète la même composition de couche en changeant le nombre de filtre seulement, qui devient 64 et 128 respectivement à la place de 32. Les dernières couches du réseau de neurones convolutionnels représente l'étape de classification qui se compose d'une couche entièrement connectée nécessitant des données a une dimension, pour ce faire nous utilisons la fonction `Flatten`, cela permettra de convertir les donnes en un vecteur a une dimension et la couche Dropout qui se charge de réduire le sur-ajustement dans le réseau. Le problème est une tâche de classification binaire, nécessitant la prédiction d'une valeur de 0 ou 1. Une couche de sortie avec 1 nœud et une activation sigmoïde sera utilisée où la valeur de la fonction sigmoïde (1) est comparée à un seuil (si la valeur est <0.5 => 0 œil fermé, si la valeur est >0.5 => 1 œil ouvert) et le modèle sera optimisé en utilisant la fonction binaire de perte d'entropie croisée.

La figure ci-dessous représente le modèle insérer :

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 70, 70, 32)        896
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 35, 35, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 35, 35, 64)        18496
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 17, 17, 64)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 17, 17, 128)       73856
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 8, 8, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 8192)              0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               1048704
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 129
=================================================================
Total params: 1,142,081
Trainable params: 1,142,081
Non-trainable params: 0
```

Figure 20 : Le modèle utilisé

-Augmentation des données :

Après avoir réalisé notre architecture souhaitée, il est important de procéder à une augmentation des données qui agit comme une technique de régularisation afin d'améliorer la capacité des modèles à généraliser ce qu'ils ont appris à de nouvelles images. Cette technique nécessite d'effectuer des légères de modification aux images d'entrées, alors nous avons fait des

décalages horizontaux et verticaux aléatoires et de retournements horizontaux aléatoires de (10 %) qui créent une image miroir.

Une fois le modèle est prêt nous formons notre réseau pour 1000 époques avec une taille de lot 64 et nous sauvegardons notre modèle formé avec Keras [43].

Notre modèle nous a permis d'obtenir des grandes précisions arrivant jusqu'à 99%, ce qui permet de classer l'état des yeux correctement dans les images d'une séquence de vidéo. Les figures ci-dessous montrent des courbes linéaires exécuter sur 1000 époques.

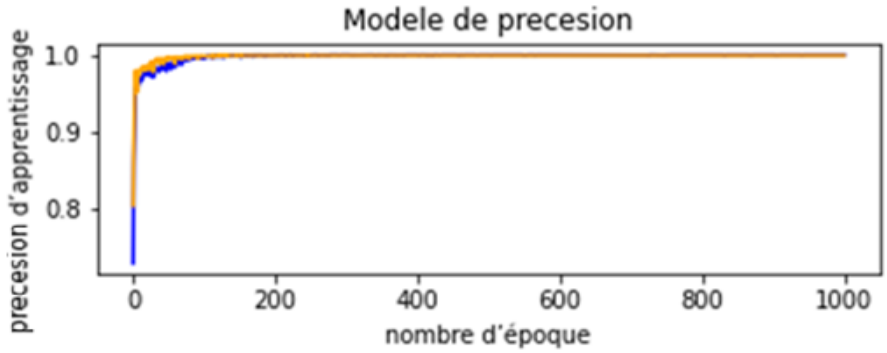

Figure 21 : Tracés linéaires des courbes d'apprentissage de la précision lors de la formation de notre réseau.

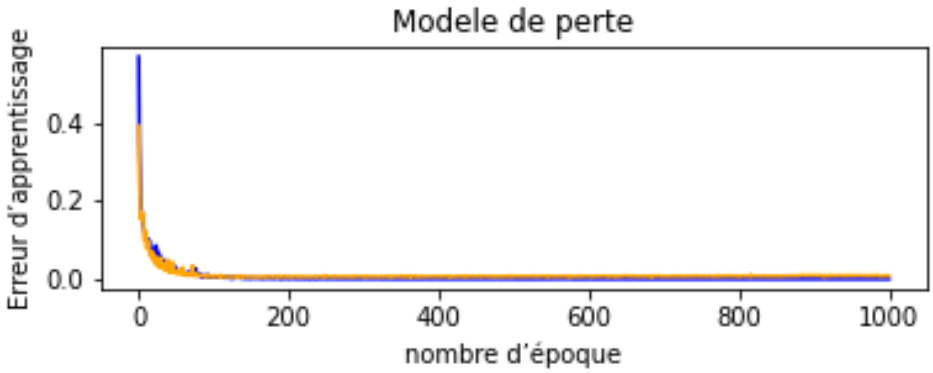

Figure 22 : Tracés linéaires des courbes d'apprentissage de la perte lors de la formation de notre réseau.

Discussion des figures :

Les figures ci-dessus montrent des courbes de précision et autres de perte du modèle sur les ensembles de données d'entrainement (bleu) et de test (orange).

Notre modèle était capable de classer correctement une séquence d'images. Il détectait l'état des yeux avec un niveau de précision de 99,901%. Pour visualiser la fonction de précision, nous avons exécuté 1000 époques, ce qui a donné le graphique de la figure 21, tandis que la perte a une valeur inférieure à 10%, comme le montre la figure 22.

On constate que la précision augmente avec le nombre d'époques jusqu'à l'époque 30. De même pour l'erreur d'apprentissage diminue avec le nombre d'époque jusqu'à l'époque 30.

- **La deuxième étape : la détection d'état des yeux par le modèle enregistrer.**

Dans cette étape, le visage et les yeux sont détectés et les points caractéristiques sont localisés par la méthode de HaarCascade [13] et Dlib [38] expliquer précédemment. D'abord, on intègre notre modèle sauvegardé dans le programme de détection déjà réalisé pour pouvoir l'utiliser et prédire le résultat souhaité, les images consécutives extraites de notre vidéo téléchargé ou obtenu en temps réel doivent être :
- de taille 70x70 ; même format que les images avec lesquelles notre modèle s'est entrainé.
- normaliser ou les valeurs de pixel de ces images seront deviser par 255.0, afin que chaque valeur de pixel ait une valeur comprise entre 0 et 1 pour que le réseau de neurones puisse les traiter.

- Apres l'exécution, le programme nous affiche la prédiction sous forme de 0 et 1 successive.

Le tableau ci-dessous illustre les résultats attribués aux quelques valeurs d'état des yeux obtenues lors de l'exécution :

Méthodes utilisées		Valeurs obtenues							
		0,4	0,1	0,35	0,28	0,09	0,33	0,32	0,12
EAR	Au-dessous du seuil (<0.26)		Fermés			Fermés			Fermés
	Au-dessus du seuil (>0.26).	Ouverts		Ouverts	Ouverts		Ouverts	Ouverts	
CNN		1	0	1	1	0	1	1	0

Tableau 3: Résultat d'exécution des deux méthodes.

4 Résultats

Détection de visage et des yeux avec les HaarCascade :

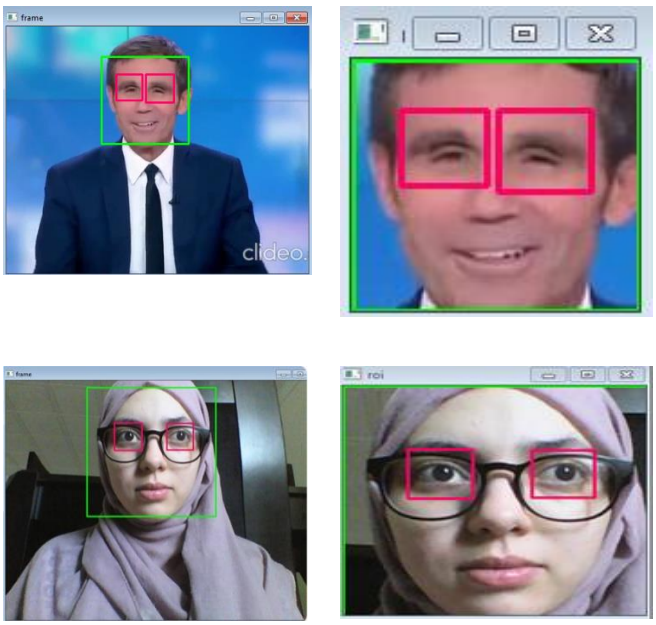

Figure 23 : Détection du visage et les yeux par les HaarCascade.

Localisation de l'œil avec Dlib pour le calcul du EAR :

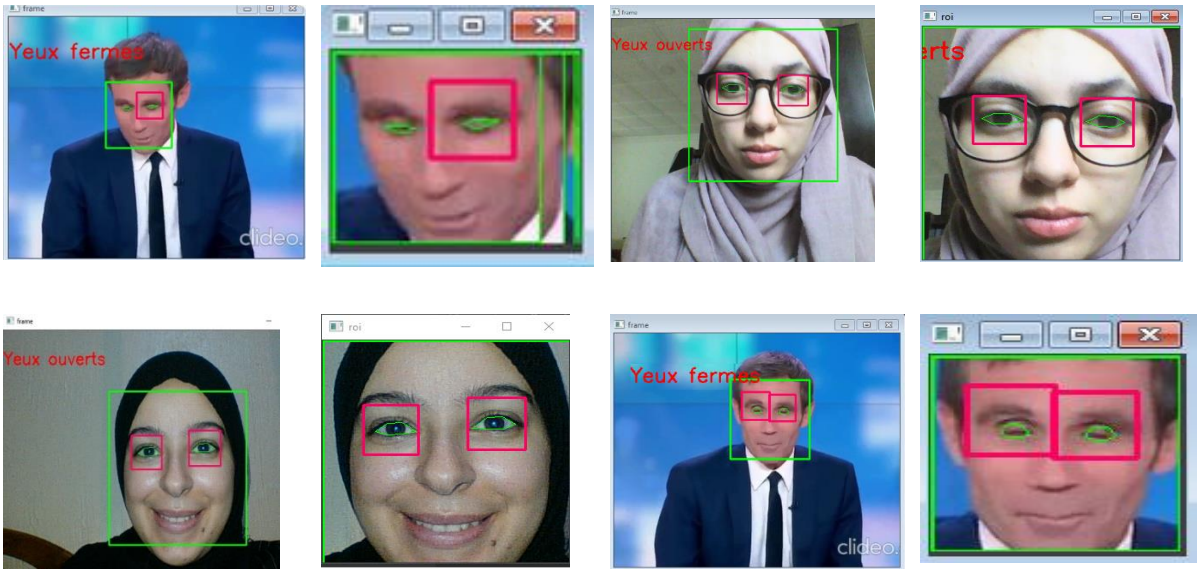

Figure 24 : Détection d'état des yeux par Dlib.

La reconnaissance de l'état des yeux avec les deux méthodes (CNN, EAR) :

Test pour des yeux fermés à l'aide des CNN :

Figure 25 : Reconnaissance d'état des yeux par les CNN et EAR pour des yeux fermés.

Test pour des yeux ouverts à l'aide des CNN :

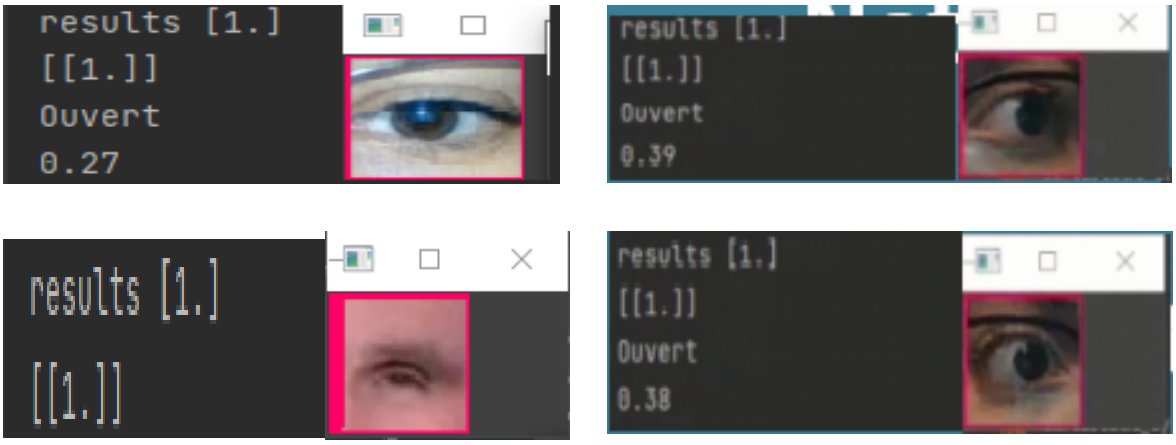

Figure 26 : Reconnaissance d'état des yeux par les CNN et EAR pour les yeux ouverts.

5 Conclusion

Ce chapitre met l'accent sur la conception et les détails de l'implémentation de notre système proposé, où l'analyse d'état des yeux se fait à partir de la détection de visage et l'extraction des points caractéristiques qui signent au changement de leurs états, en utilisant deux méthodes : une adoptant l'apprentissage en profondeur par les CNN qui utilise comme entrée une image d'un œil extraite à partir d'un visage détecté par les HaarCascade de Viola et Jones et l'autre méthode adopte la librairie Dlib pour trouver les points caractéristiques des yeux afin de calculer la valeur du EAR. En combinant ces deux méthodes pour plus de précision lors de la reconnaissance de l'état des yeux.

Conclusion générale

Le réseau de neurones convolutionnels est l'un des algorithmes qui ont connu une évolution remarquable ces dernières années dans le domaine de la vision par ordinateur en raison de leurs grandes capacités à apprendre, ce qui a contribué à remplacer l'êtres humains dans certaines tâches par des machines intelligentes. Dans ce mémoire, nous nous sommes intéressés à introduire les méthodes d'apprentissage approfondie CNN pour la reconnaissance d'état des yeux suite à leurs grands succès dans la classification des images et l'identification des objets.
L'objectif principal de notre système est la reconnaissance d'état des yeux d'une personne dans une succession d'images en temps réelle, cette opération est loin d'être réalisée parfaitement, surtout dans des environnements non contrôlés. A guise de ce problème, nous avons proposé une application adoptant les CNN et EAR afin de pouvoir répondre aux exigences de rapidité et de robustesse des résultats. Pour ce faire, la première phase réalisée consiste à détecter le visage et les yeux d'une personne à l'aide des HaarCascade en minimisant l'espace de recherche, cela nous a permis de gagner en temps de calcul et réduire dans certain cas les fausses détections. Ensuite, la seconde étape concerne l'extraction de la zone des yeux depuis le visage détecté, qui va nous permettre d'analyser les caractéristiques de l'œil en appliquant des mesures contribuant à déterminer son état. Ces mesures prennent en compte l'utilisation du métrique EAR qui calcule la distance entre les points des yeux localiser par Dlib et l'intégration de l'apprentissage approfondie qui a comme entrée la zone extraite, ces opérations rajoute plus de précision et de confiance à notre système.
Dans ce mémoire, nous avons présenté les différentes étapes pour la réalisation de notre système qui sont : la détection de visage et des yeux, ces étapes représentent la base de notre système et contribuent par la suite pour analyser et déterminer l'état de l'ouverture ou la fermeture des yeux. De plus, nous avons mis en œuvre des méthodes développer par des chercheurs dont le but est de trouver celle qui répond le mieux aux contraintes de robustesse, temps de traitement, indépendance d'environnement extérieur et fiabilité.

Notre travail ouvre des perspectives scientifiques à court et à long terme. Nous soulignons dans ce qui suit les perspectives qui nous semblent pertinentes pour l'évolution des systèmes développés dans ce projet.

- Utiliser une application plus précise pour détecter les yeux dans leurs différents états.
- Il serait intéressant de travailler avec une base de données plus large et plus variante.
- Enfin, Nous imaginons d'aller en profondeur sur l'investigation et la recherche des différentes architectures neuronal qui sont efficace pour la reconnaissance d'état des yeux.

Ce sujet reste au centre de recherche vu son utilité dans divers champs d'applications y compris la détection de la somnolence du conducteur et les systèmes d'assistance aux personnes handicapées. L'application développer a pu répondre aux objectifs du projet, et ses performances en termes de reconnaissance reste à améliorer dans des travaux futurs.

Bibliographies

[1] EDDINE, Benrachou Djamel, DOS SANTOS, Filipe Neves, BOULEBTATECHE, Brahim, et al. Eyelsd a robust approach for eye localization and state detection. Journal of Signal Processing Systems, 2018, vol. 90, no 1, p. 99-125.

[2] Ren, Y., Wang, S., Hou, B., & Ma, J. (2014). A novel eye localization method with rotation invariance. IEEE Transactions on Image Processing, 23(1), 226–239.

[3] Song, F., Tan, X., Chen, S., & Zhou, Z.H. (2013). A literature survey on robust and efficient eye localization in real-life scenarios. Pattern Recognition, 46(12), 3157–3173

[4] Ming-Hsuan Yang, David J. Kriegman et Narendra Ahuja. Detecting faces in images : A survey. Dans IEEE Transaction on Pattern Analysis and Machine Intelligence, 2002, 24(1), 34-58.

[5] Luhong Liang, Haizhou Ai et Guangyou Xu"Face Detection based on Template Matching and Neural Network Verification". Acta Electronica Sinica.2001

[6] T.P. NGUYEN « détection de visage dans images de couleur en utilisant une caractéristique invariante » 3rd International conference : sciences of electronic March 27-31, 2005

[7] Z. HAMMAL & l. COUVREUR & A.CAPLIER & M. ROMBANT « Facial expression classification: An approuch based on the fusion of facial deformations using the transferable belief model. Int. J.Approx. Reason, 46(3)

[8] L. C. De Silva, K. Aizawa, and M. Hatori, Detection and tracking of facial features by using a facial feature model and deformable circular template, IEICE Trans. Inform. Systems E78–D (9), 1195–1207, 1995.

[9] Cheng-Chin, Wen-Kai Tai, Mau-Tsuen Yang, Yi-Timg Huang, and chi-Janng Huang. A novel method for detecting lips, eyes and faces in real time. Real-Time Imaging, 9(4) : 277-287, 2003.

[10] Wenlong Zheng and Suchendra M. Bhandarkar. Face detection and haking using a boosted adaptative particle filter. Journal of visual communication and Imog Representation, 20(1) : 9-27, 2009.

[11] H. A. Rowley, S. Baluja, and T. Kanade. Neural Network-based face detection. IEEE Transachen on Pattern Analysis and Machine Intelligence, 20(1) : 23-38, 1998.

[12] P. Viola and M. Jones, "Robust Real-Time Face Detection," International Journal of Computer Vision, vol. 57, pp. 137--154, 01 May 2001.

[13] Viola. P, Jones. M, “Rapid object detection using a boosted cascade of simple features”, Proceedings of the 2001 IEEE Computer Society Conference, vol 1, pp 511-518, 2001

[14] P. Wang et al., “Automatic eye detection and its validation,” in IEEE Comput. Soc. Conf. Comput. Vision.

[15] Z. Ye et al., “Detecting eye contact using wearable eye-tracking glasses,” in ACM Conf. Ubiquitous Comput., pp. 699–704 (2012).

[16] W. B. Horng et al., “Improvements of driver fatigue detection system based on eye tracking and dynamic template matching,” WSEAS Trans. Inf. Sci. Appl. 9(1), 14–23 (2012).

[17] F. Alonso-Fernandez and J. Bigun, “A survey on periocular biometrics research,” Pattern Recognit. Lett. 82, 92–105 (2016).

[18] Z. H. Zhou and X. Geng, “Projection functions for eye detection,”Pattern Recognit. 37(5), 1049–1056 (2004).

[19] M. Hassaballah, T. Kanazawa, and S. Ido, “Efficient eye detection method based on grey intensity variance and independent components analysis,” IET Comput. Vision 4(4), 261–271 (2010).

[20] S. Zhao and R. R. Grigat, “Robust eye detection under active infrared illumination,” in Proc. 18th Int. Conf. Pattern Recognit., IEEE Computer Society, Vol. 4, pp. 481–484 (2006).

[21] B. Kroon et al., “Eye localization in low and standard definition content with application to face matching,” Comput. Vision Image Understanding 113(8), 921–933 (2009).

[22] J. Wang et al., “A novel eye localization method based on Log-Gabor transform and integral image,” Appl. Math. 6(2S), 323S–329S (2012).

[23] L. Liang et al., “Face alignment via component-based discriminative search,” Lect. Notes Comput. Sci. 5303, 72–85 (2008).

[24] X. Tan et al., “Enhanced pictorial structures for precise eye localization under incontrolled conditions,” in IEEE Conf. Comput. Vision and Pattern Recognit., pp. 1621–1628 (2009).

[25] M. Q. Jing and L. H. Chen, “A novel method for horizontal eye line detection under various environments,” Int. J. Pattern Recognit.Artif. Intell. 24(3), 475–498 (2010).

[26] M. Hassaballah, T. Kanazawa, and S. Ido, “Efficient eye detection method based on grey intensity variance and independent components analysis,” IET Comput Vision 4(4), 261–271 (2010).

[27] Gonzalez-Ortega, D., D ‘’ıaz-Pernas, F., Anton-Rodr ‘’ıguez, M., Mart’ınez-Zarzuela, M., & D’ıez-Higuera, J. (2013). Real-time vision-based eye state detection for driver alertness monitoring. Pattern Analysis and Applications, 16(3), 285–306.

[28] Vapnik, V.N., & Vapnik, V. (1998). Statistical learning theory (Vol. 1). New York: Wiley.

[29] Xiang, C., Ding, S.Q., & Lee, T.H. (2005). Geometrical interpretation and architecture selection of mlp. IEEE Transactions on Neural Networks/a Publication of the IEEE Neural Networks Council, 16(1), 84–96.

[30] Xu, C., Zheng, Y., & Wang, Z. (2008). Eye states detection by boosting local binary pattern histogram features. In 15th IEEE international conference on image processing, 2008. ICIP 2008 (pp. 1480–1483).

[31] Kalbkhani, H., Shayesteh, M.G., & Mohsen Mousavi, S. (2013). Efficient algorithms for detection of face, eye and eye state. Computer Vision, IET, 7(3), 184–200.

[32] Sirohey, S., Rosenfeld, A., & Duric, Z. (2002). A method of detecting and tracking irises and eyelids in video. Pattern Recognition, 35(6), 1389–1401.

[33] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, pages 65–386, 1958.

[34] W.C. Mc Culloh, W.H.Pitts, A logical calculus ideas imanent in nervous activity, Bulletin of Mathematical Biophysics, 5, p.115, 1943.

[35] C.TOUZET « Les réseaux de neurones artificiels introduction au connexionnisme cours, exercices et travaux pratiques », Humaine, Marseille, Juillet 1992.

[36] N. RIZKALLA « Nanoparticules et réseaux de neurones artificiels : de la préparation à la modélisation », thèse de doctorat de l'université de Montréal Canada, février 2005.

[37] Mokri Mohammed Zakaria, « Classification des images avec les réseaux de neurones,Convolutionnels», Mémoire de fin d'études Pour l'obtention du : diplôme de Master eninformatique, Université Abou BakrBelkaid Tlemcen, 2017.

[38] Davis E. King. Dlib-ml: A Machine Learning Toolkit. Journal of Machine Learning Research 10, pp. 1755-1758, 2009

[39] Gary Bradski and Adrian Kaehler 1. ed. edition, (2008). "Learning OpenCV: Computer vision with the OpenCV library"

[40] Jachym, Marc, "Cours de Programmation avec le langage Python: Niveau débutant en programmation",Licence professionnelle – Métrologie dimensionnelle et qualité IUT de St Denis, Université Paris 13.

[41] K. C. Patel, S. A. Khan, and V. N. Patil, "Real-Time Driver Drowsiness Detection System Based on Visual Information," vol. 8, no. 3, pp. 16200– 16203, 2018

[42] Prasad V Patil (2020-12-20, maj 2020-12-22). Drowsiness Detection Dataset
Classify based on whether Eyes are Closed or Open. , sur le site kaggle. Consulté le 16-05-2021 https://www.kaggle.com/prasadvpatil/mrl-dataset/metadata

[43] Chollet, F. & autres, 2015. Keras. Disponible sur : https://github.com/fchollet/keras.

[44] Vidéo téléchargé, URL : https://www.youtube.com/watch?v=A8FqX9lm2EE. Consulté le : 15-06-2021

[45] Harris, C.R. et al., 2020. Array programming with NumPy. Nature, 585, pp.357–362.

[46] Classification ImageNet avec des réseaux de neurones à convolution profonde (2012), In Advances in Neural Information Processing Systems 25 (NIPS 2012)

[47] He Kaiming et al. (2015): Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, ICCV.

[48] JetBrains, 2017. PyCharm. [online] JetBrains. Disponible sur: https://www.jetbrains.com/pycharm/>

[49] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G. & Isard, M. (2016). TensorFlow : A System for Large-Scale Machine Learning. OSDI, 16, 265–283.

Annexes

Annexes a :

Détection de visage et des yeux avec les HaarCascade :

```
import cv2

faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")
eye_cascade = cv2.CascadeClassifier('haarcascades/haarcascade_eye.xml')

cap = cv2.VideoCapture('test-video.mp4')

while True:
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    # face detection
    faces = faceCascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=4, minSize=(30, 30))
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
        # face extracting
        roi_color = frame[y:y + h, x:x + w]
        # eye detection
        eyes = eye_cascade.detectMultiScale(frame, scaleFactor=1.1, minNeighbors=4, minSize=(30, 30))
        for (ex, ey, ew, eh) in eyes:
            cv2.rectangle(frame, (ex, ey), (ex + ew, ey + eh), (100, 0, 255), 2)

        # afficher roi
        cv2.imshow("roi", roi_color)
        cv2.imshow("frame", frame)
        # Exit when escape is pressed
    if cv2.waitKey(delay=1) == 27:
        cv2.waitKey(0)
        break
    # When everything done, release the video capture and video write objects
cap.release()
```

Localisation de l'œil avec Dlib pour le calcul du EAR :

```
import cv2
import dlib
from imutils import face_utils
from scipy.spatial import distance as dist

faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")
eye_cascade = cv2.CascadeClassifier('haarcascades/haarcascade_eye.xml')

def eye_aspect_ratio(eye):
    A = dist.euclidean(eye[1], eye[5])
    B = dist.euclidean(eye[2], eye[4])
    C = dist.euclidean(eye[0], eye[3])
    ear = (A + B) / (2.0 * C)
    return ear

EYE_AR_THRESH = 0.26
EYE_AR_CONSEC_FRAMES = 3
COUNTER = 0
TOTAL = 0

cap = cv2.VideoCapture('test-video.mp4')

while True:
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    # face detection
    faces = faceCascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=4, minSize=(30, 30))
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
        # face extracting
        roi_color = frame[y:y + h, x:x + w]
        # eye detection
```

```
        # eye detection
        eyes = eye_cascade.detectMultiScale(frame, scaleFactor=1.1, minNeighbors=4, minSize=(30, 30))
        predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')
        for (ex, ey, ew, eh) in eyes:
            cv2.rectangle(frame, (ex, ey), (ex + ew, ey + eh), (100, 0, 255), 2)

        rects = faceCascade.detectMultiScale(frame)

        (lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
        (rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]

        for (ex, ey, ew, eh) in rects:
            rect = dlib.rectangle(ex, ey, ex + ew, ey + eh)
            shape = predictor(frame, rect)
            shape = face_utils.shape_to_np(shape)
            leftEye = shape[lStart:lEnd]
            rightEye = shape[rStart:rEnd]
            leftEAR = eye_aspect_ratio(leftEye)
            rightEAR = eye_aspect_ratio(rightEye)
            EAR = (leftEAR + rightEAR) / 2
            EAR = round(EAR, 2)
            if EAR < 0.26:
                cv2.putText(frame, "You are drowsy !", (20, 50),
                            cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
                cv2.putText(frame, "Take a rest ", (20, 80),
                            cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
            print(EAR)

            ear = (leftEAR + rightEAR) / 2.0
```

```
            ear = (leftEAR + rightEAR) / 2.0

            leftEyeHull = cv2.convexHull(leftEye)
            rightEyeHull = cv2.convexHull(rightEye)
            cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)
            cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)
            if ear < EYE_AR_THRESH:
                COUNTER += 1
            else:
                # if the eyes were closed for a sufficient number of
                # then increment the total number of blinks
                if COUNTER >= EYE_AR_CONSEC_FRAMES:
                    TOTAL += 1

                # reset the eye frame counter
                COUNTER = 0
                cv2.putText(frame, "EAR: {:.2f}".format(ear), (300, 30),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

        # afficher roi
        cv2.imshow("roi", roi_color)
        cv2.imshow("frame", frame)
        # Exit when escape is pressed
    if cv2.waitKey(delay=1) == 27:
        cv2.waitKey(0)
        break
    # When everything done, release the video capture and video write objects
cap.release()
```

La création du modèle sur Google Colab avec les résultats obtenus :

```
# load Closed_Eyes vs Open_Eyes dataset, reshape and save to a new file
from os import listdir
from numpy import asarray
from numpy import save
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
# define location of dataset
folder = '/content/drive/MyDrive/train/'
photos, labels = list(), list()
# enumerate files in the directory
for file in listdir(folder):
  # determine class
  output = 0.0
  if file.startswith('Closed_Eyes'):
    output = 1.0
  # load image
  photo = load_img(folder + file, target_size=(70, 70))
  # convert to numpy array
  photo = img_to_array(photo)
  # store
  photos.append(photo)
  labels.append(output)
# convert to a numpy arrays
photos = asarray(photos)
labels = asarray(labels)
print(photos.shape, labels.shape)
# save the reshaped photos
save('Closed_Eyes_vs_Open_Eyes.npy', photos)
save('Closed_Eyes_vs_Open_Eyes_labels.npy', labels)
```

```
(4000, 70, 70, 3) (4000,)
```

```
[ ]  # load and confirm the shape
     from numpy import load
     photos = load('Closed_Eyes_vs_Open_Eyes.npy')
     labels = load('Closed_Eyes_vs_Open_Eyes_labels.npy')
     print(photos.shape, labels)
```

```
     (4000, 70, 70, 3) [0. 1. 0. ... 1. 1. 1.]
```

```
[ ]  import os
     # create directories
     dataset_home = 'dataset_Closed_Eyes_vs_Open_Eyes/'
     subdirs = ['train/', 'test/']
     for subdir in subdirs:
       # create label subdirectories
       labeldirs = ['Open_Eyes/', 'Closed_Eyes/']
       for labldir in labeldirs:
         newdir = dataset_home + subdir + labldir
         os.makedirs(newdir, exist_ok=True)
```

```
from shutil import copyfile
from random import seed
from random import random
from os import listdir

# seed random number generator
seed(1)
# define ratio of pictures to use for validation
val_ratio = 0.25
# copy training dataset images into subdirectories
src_directory = '/content/drive/MyDrive/train/'
for file in listdir(src_directory):
  src = src_directory + '/' + file
  dst_dir = 'train/'
  if random() < val_ratio:
    dst_dir = 'test/'
  if file.startswith('Closed_Eyes'):
    dst = dataset_home + dst_dir + 'Closed_Eyes/'  + file
    copyfile(src, dst)
  elif file.startswith('Open_Eyes'):
    dst = dataset_home + dst_dir + 'Open_Eyes/'  + file
    copyfile(src, dst)
```

```
import sys
from matplotlib import pyplot
from keras.models import Sequential
from keras.layers import Conv2D, Dropout
from keras.layers import MaxPooling2D
from keras.layers import Dense
from keras.layers import Flatten
from keras.optimizers import SGD
from keras.preprocessing.image import ImageDataGenerator
# define cnn model
def define_model():
  model = Sequential()
  model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', input_shape=(70, 70, 3)))
  model.add(MaxPooling2D((2, 2)))
  model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
  model.add(MaxPooling2D((2, 2)))
  model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
  model.add(MaxPooling2D((2, 2)))
  model.add(Flatten())
  model.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))
  model.add(Dropout(0.5))
  model.add(Dense(1, activation='sigmoid'))
  # compile model
  opt = SGD(learning_rate=0.001, momentum=0.9)
  model.compile(optimizer=opt, loss='binary_crossentropy', metrics=['accuracy'])
  return model
```

```
  # prepare iterators
train_it = datagen.flow_from_directory('dataset_Closed_Eyes_vs_Open_Eyes/train/',class_mode='binary', batch_size=64, target_size=(70, 70))
test_it = datagen.flow_from_directory('dataset_Closed_Eyes_vs_Open_Eyes/test/',class_mode='binary', batch_size=64, target_size=(70, 70))
  # fit model
history = model.fit(train_it, steps_per_epoch=len(train_it),
                              validation_data=test_it, validation_steps=len(test_it), epochs=1000, verbose=1)
  # evaluate model
_, acc = model.evaluate(test_it, steps=len(test_it), verbose=0)
print('> %.3f' % (acc * 100.0))
# learning curves
summarize_diagnostics(history)
model.summary()
model.save('model22.h5')
```

```
Found 2986 images belonging to 2 classes.
Found 1014 images belonging to 2 classes.
Epoch 1/1000
47/47 [==============================] - 4s 54ms/step - loss: 0.6802 - accuracy: 0.5769 - val_loss: 0.4421 - val_accuracy: 0.8521
Epoch 2/1000
47/47 [==============================] - 2s 46ms/step - loss: 0.4326 - accuracy: 0.8165 - val_loss: 0.3789 - val_accuracy: 0.7722
Epoch 3/1000
47/47 [==============================] - 2s 46ms/step - loss: 0.3216 - accuracy: 0.8633 - val_loss: 0.2344 - val_accuracy: 0.9606
Epoch 4/1000
47/47 [==============================] - 2s 46ms/step - loss: 0.2362 - accuracy: 0.9275 - val_loss: 0.1764 - val_accuracy: 0.9773
Epoch 5/1000
47/47 [==============================] - 2s 46ms/step - loss: 0.1882 - accuracy: 0.9417 - val_loss: 0.1636 - val_accuracy: 0.9655
Epoch 6/1000
47/47 [==============================] - 2s 47ms/step - loss: 0.1915 - accuracy: 0.9412 - val_loss: 0.1572 - val_accuracy: 0.9586
Epoch 7/1000
```

Le test du modèle sur Google Colab :

```
# make a prediction for a new image.
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.models import load_model

# load and prepare the image
def load_image(filename):
  img = load_img(filename, target_size=(70, 70))
  img = img_to_array(img)
  img = img.reshape(1, 70, 70, 3)
  return img
def run_example():
  img = load_image('/content/drive/MyDrive/eye1.jpg')
  model = load_model('/content/model22.h5')
  #model.summary()
  result = model.predict(img)
  print(result[0])
run_example()
```

```
[0.]
```

-Zéro pour les yeux fermés.

La reconnaissance de l’état des yeux avec les deux méthodes (CNN, EAR) :

```python
import cv2
import dlib
from imutils import face_utils
from keras.layers import Conv2D
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers import MaxPooling2D
from scipy.spatial import distance as dist
from keras.models import Sequential
from keras.layers import Dropout
import numpy as np
import tensorflow as tf
faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")
eye_cascade = cv2.CascadeClassifier('haarcascades/haarcascade_eye.xml')

def eye_aspect_ratio(eye):
    A = dist.euclidean(eye[1], eye[5])
    B = dist.euclidean(eye[2], eye[4])
    C = dist.euclidean(eye[0], eye[3])
    ear = (A + B) / (2.0 * C)
    return ear

EYE_AR_THRESH = 0.26
EYE_AR_CONSEC_FRAMES = 3
COUNTER = 0
```

```python
TOTAL = 0
cap = cv2.VideoCapture('test-video.mp4')

# define cnn model
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform',
                 padding='same', input_shape=(70, 70, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))
model.summary()
model.load_weights('model22.h5')
print("My weights", model)
while True:
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

```
    # face detection
    faces = faceCascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=4, minSize=(30, 30))
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
        # face extracting
        roi_color = frame[y:y + h, x:x + w]
        print("[INFO] Object found.")
        # eye detection
        eyes = eye_cascade.detectMultiScale(frame, scaleFactor=1.1, minNeighbors=4, minSize=(30, 30))
        print('check size',np.shape(eyes))
        predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')
        for (ex, ey, ew, eh) in eyes:
            cv2.rectangle(frame, (ex, ey), (ex + ew, ey + eh), (100, 0, 255), 2)
            roi_color_eyes = frame[ey:ey + eh, ex:ex + ew]
            roi_color_eyes_70 = cv2.resize(roi_color_eyes, (70, 70), interpolation=cv2.INTER_AREA)
            roi_color_eyes_70 = roi_color_eyes_70 / 255.0
            cv2.imshow("ab", roi_color_eyes_70)
            roi_color_eyes_70 = tf.reshape(roi_color_eyes_70, [-1, 70, 70, 3])
            # predict the class
            results = model.predict(roi_color_eyes_70)
            print("results", results[0])
            print(np.round(results))
        rects = faceCascade.detectMultiScale(frame)
        (lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
        (rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
```

```
        for (ex, ey, ew, eh) in rects:
            rect = dlib.rectangle(ex, ey, ex + ew, ey + eh)
            shape = predictor(frame, rect)
            shape = face_utils.shape_to_np(shape)
            leftEye = shape[lStart:lEnd]
            rightEye = shape[rStart:rEnd]
            leftEAR = eye_aspect_ratio(leftEye)
            rightEAR = eye_aspect_ratio(rightEye)
            EAR = (leftEAR + rightEAR) / 2
            EAR = round(EAR, 2)
            if EAR < 0.26:
                cv2.putText(frame, "Yeux fermes", (1, 60),
                            cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
                print("Ferme")
            else:
                cv2.putText(frame, "Yeux ouverts ", (1, 60),
                            cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
                print("Ouvert")

            print(EAR)
            ear = (leftEAR + rightEAR) / 2.0
            leftEyeHull = cv2.convexHull(leftEye)
            rightEyeHull = cv2.convexHull(rightEye)
            cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)
            cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)
```

```
        # afficher roi
        cv2.imshow("roi", roi_color)
        cv2.imshow("frame", frame)
        # Exit when escape is pressed
    if cv2.waitKey(delay=1) == 27:
        cv2.waitKey(0)
        break
    # When everything done, release the video capture and video write objects
cap.release()
```