الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

Mémoire de Master

Filière : Électronique
Spécialité : Electronique des systèmes embarqués

Présenté par

Zineb Benaissa

&

Rania Lardjane

Implémentation sur Raspberry pi de modèles de détection de la somnolence par deep learning

Promotrice : Mme Bougherira Nadia.

2022-2023

Remerciements

Tout d'abord, nous aimerions exprimer notre gratitude envers Dieu de nous avoir donné le courage et la volonté nécessaires pour mener à bien ce travail. Nous sommes profondément reconnaissants envers notre mentor, Mme. Bougherira Nadia, pour ses conseils inlassables, son soutien constant et ses précieuses orientations. Sa contribution a été d'une importance capitale pour la réussite de notre projet.

Nous souhaitons également adresser nos chaleureux remerciements à nos enseignants, dont la sagesse et l'expertise nous ont guidés tout au long de notre parcours universitaire. Leurs connaissances partagées et leurs encouragements nous ont permis de progresser et de développer nos compétences.

Nous tenons à exprimer notre reconnaissance envers les membres du jury qui ont accordé leur temps et leur attention à évaluer notre travail. Leur engagement et leur bienveillance nous ont honorés et nous en sommes profondément reconnaissants.

Nos familles méritent également un grand remerciement pour leur soutien constant tout au long de nos études universitaires. Leur amour, leurs encouragements et leur compréhension ont été d'une importance cruciale dans notre réussite académique.

Enfin, nous tenons à exprimer notre gratitude envers tous nos amis qui ont partagé notre parcours. Leurs encouragements, leurs conseils et leur soutien moral nous ont aidés à surmonter les difficultés et à maintenir notre motivation.

Nous sommes conscients que sans le soutien de toutes ces personnes, notre parcours aurait été plus ardu. Leurs contributions ont façonné notre réussite et nous ont permis de réaliser ce travail avec fierté.

À chacun d'entre vous, nous vous adressons un sincère et chaleureux merci. Vos efforts et votre soutien resteront gravés dans notre mémoire.

ملخص:

يهدف هذا المشروع إلى تنفيذ نظام كشف التعب بناءً على تعلم الآلة العميق على(Raspberry Pi (Tensorflow يوفر هذا الدمج حلاً فعالًا للكشف الدقيق وفي الوقت الفعلي. يدرس البحث جوانب الأجهزة والبرمجيات المحددة لـRaspberry Pi ، بالإضافة إلى مقارنة أداء النظام بطريقة أخرى لكشف التعب(Dlib). الهدف من هذا هو تعزيز السلامة واليقظة في مختلف المجالات مثل قيادة السيارات والمراقبة.

الكلمات المفتاحية :كشف التعب، تعلم الآلة العميق، الوقت الحقيقي، قيادة السيارات، Tensorflow ، Dlib.

Résumé :

Notre projet vise à implémenter un système de détection de somnolence basé sur l'apprentissage profond sur Raspberry Pi(Tensorflow). Cette combinaison offre une solution portable et économe en énergie pour une détection précise et en temps réel. L'étude examine les aspects matériels et logiciels spécifiques à Raspberry Pi, ainsi que les performances du système par rapport à une deuxième méthode de détection de somnolence (Dlib). L'objectif est d'améliorer la sécurité et la vigilance dans des domaines tels que la conduite automobile et la surveillance.

Mots clés : somnolence, tensorflow, apprentissage profond, Raspberry pi, dlib, conduite.

Abstract :

Our project aims to implement a drowsiness detection system based on deep learning on Raspberry Pi(Tensorflow). This combination offers a portable and energy-efficient solution for accurate and real-time detection. The study examines the hardware and software aspects specific to Raspberry Pi, as well as the system's performance compared to another drowsiness detection method (Dlib). The goal is to enhance safety and vigilance in domains such as automotive driving and surveillance.

Key words: drowsiness, detection, tensorflow, raspberry pi, deep learning,dlib, driving.

Listes des acronymes et abréviations

CCD: Charge Coupled Device.

CNN: Convolutional Neural network.

CNN 3D : Réseaux de neurones à convolution 3D.

CEW: Closed Eyes in Wild.

DL: Apprentissage profond**.**

EAR: Eye Aspect Ratio.

ECG: Electrocardiography.

EEG: Electroencéphalograpе.

EOG: Electro-oculographie.

HDMI: Interface multimédia haute définition.

HSV: Teinte, saturation, luminosité.

IA: Intelligence artificiel.

IDE: Environnement de développement intégré

GPIO: General Purpose Input/Output.

LSTM: Long short-term memory.

MAR: Mouth Aspect Ration.

ML: Apprentissage machine.

MLR: Multiple Linear Regressions.

MLP: multilayer perceptron.

PPP: Point par pouce.

RAM: Random access memory.

ROI: Region of Interest.

RNN: Recurrent Neural Network.

RGB: Rouge, vert, bleu.

SVM: SupportVectorMachie.

USB: Universal Serial Bus.

Table des matières :

Table de figure :

Liste des tableaux

Introduction générale

Dans un monde où la sécurité et la vigilance sont essentielles, la détection de la somnolence est d'une importance cruciale, notamment dans des domaines tels que la conduite automobile, les opérations de surveillance ou encore la prévention des accidents liés à la fatigue. Les avancées récentes dans le domaine de l'apprentissage profond offrent de nouvelles opportunités pour développer des systèmes de détection de somnolence plus précis et réactifs.

L'objectif de ce projet est d'implémenter un système de détection de somnolence basé sur l'apprentissage profond, en utilisant Raspberry Pi comme plateforme de déploiement. Raspberry Pi, avec sa taille compacte et sa facilité d'utilisation, est devenu une solution populaire pour des projets embarqués et l'implémentation de systèmes d'intelligence artificielle à petite échelle.

La combinaison de l'apprentissage profond et de Raspberry Pi permettra de réaliser une détection de somnolence en temps réel, en utilisant des techniques avancées de vision par ordinateur pour analyser les signes de fatigue et d'inattention. Cette implémentation sur Raspberry Pi offre une solution portable, économe en énergie et pouvant être intégrée dans diverses applications.

Au cours de cette étude, nous aborderons les aspects matériels et logiciels spécifiques à l'utilisation de Raspberry Pi, ainsi que les bibliothèques et outils nécessaires pour développer et déployer le système de détection de somnolence. De plus, nous évaluerons les performances du système et comparerons les résultats obtenus avec d'autres méthodes de détection de somnolence.

Ce mémoire est structuré en 3 chapitres de la manière suivante :

- Le premier chapitre aborde les notions générales sur le traitement d'image et l'acquisition de données. Nous y discutons également de l'intelligence artificielle et des réseaux de neurones convolutifs (CNN).
- Dans le deuxième chapitre, nous présentons le développement de deux systèmes de détection de la somnolence. Le premier système utilise un réseau de neurones CNN avec la bibliothèque TensorFlow, tandis que le deuxième système utilise le modèle shape predictor avec la bibliothèque dlib.
- Dans le troisième chapitre, nous conclurons notre travail en mettant en œuvre les deux systèmes de détection de la somnolence sur raspberry pi et en évaluant les résultats.

Chapitre 1 Généralité

1.1Introduction

Aujourd'hui, le traitement d'images est un domaine très vaste qui a connu un développement important depuis quelques dizaines d'années. Ce domaine a contribué au développement de plusieurs domaines tels que la médecine, l'astronomie, l'informatique, etc. Le traitement des images est une méthode permettant d'effectuer certaines opérations sur une image afin d'obtenir une image améliorée ou d'en extraire des informations utiles. On peut le définir aussi comme un type de traitement du signal dans lequel l'entrée est une image et la sortie peut être une image dans laquelle des caractéristiques associées à cette image.

Le traitement d'image est basé sur trois étapes essentielles : l'importation de l'image via des outils d'acquisition d'image comme la caméra, l'analyse et la manipulation de l'image, et finalement, l'extraction des informations, on peut visualiser cela sur la figure 1.1.

Dans ce chapitre, nous abordons les notions de base nécessaires à la compréhension des techniques de traitement d'images et Vidéos.

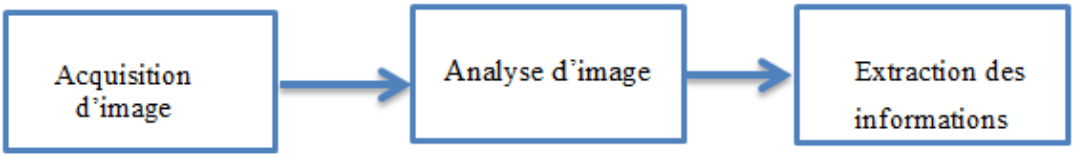

Figure 1.1: Les étapes principales de traitement d'image.

1.2 Définition de l’image numérique :

Une image numérique peut être considérée comme une représentation discrète de données possédant à la fois des informations spatiales (mise en page) et d'intensité (couleur), on peut aussi considérer comme un signal lumineux multidimensionnel **[1]**. L'image peut aussi être représentée mathématiquement selon de fonction bidimensionnelle (x, y), De $\Re \times \Re$ dans $\Re$ où le couplet d'entrée est considéré comme une position spatiale et le singleton de sortie comme l'intensité de l’image.

1.2.1 Caractéristique d’une image numérique :

a-Pixel :

Le pixel est le plus petit élément contrôlable d'une image. Il est souvent abrégé p ou px qui possède une valeur qui peut être un scalaire qui représente un niveau de gris, ou un vecteur représentant une couleur, ou toute autre chose **[2]**.

b-Résolution d’une image :

La résolution d'une image est définie par le nombre de pixels dans une longueur donnée (en pouce) dans cette image. Elle est exprimée en PPP (Point par pouce) où le pouce est 25.4 mm, ce nombre à une relation directe avec la qualité d'une image, ça signifie qu’une image avec haute qualité a un grand nombre de pixels dans un pouce (figure 1.2).

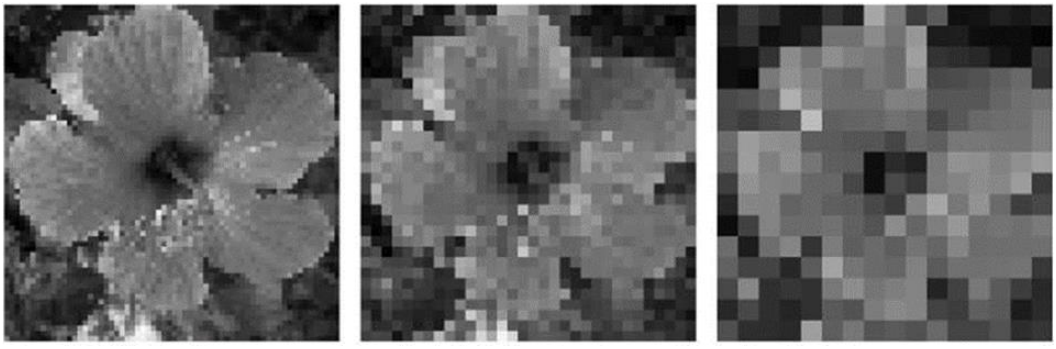

Figure 1.2 : Une image a trois résolutions différentes **[3].**

c- Dimension d'une image :

Il s'agit de la taille de l'image, qui représente le nombre total de pixels représentant l'image. Ce nombre de pixels représente une matrice bidimensionnelle (C x

R) où le C le nombre de colonnes et R est le nombre de lignes (ex. 640 x 480, 800 x 600,1024 x 768, etc.) **[1]**.

d- Espace de couleur :

Chaque image numérique contient un ou plusieurs canaux de couleur, qui définissent l'intensité ou une certaine couleur aux emplacements de l'image ($\boldsymbol{x}$, $\boldsymbol{y}$). Et la transformation d'une image d'une représentation abstraite en une représentation réelle nécessite l'existence d'une palette qui associe chaque valeur de la représentation abstraite à une couleur de cette palette.

Les palettes de couleurs les plus connues sont les niveaux de gris (grayscale), RGB et HSV.

- **Images à niveaux de gris (grayscale) :** C'est un système de codage informatique des couleurs avec un seul Canal où chaque valeur de ce canal varie de 0 (noir) à 255(blanc) (figure 1.3).

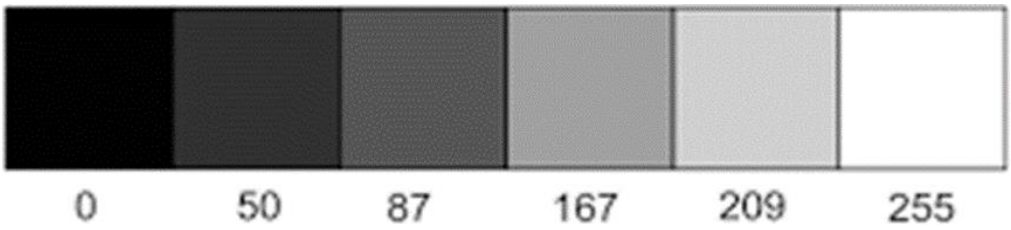

Figure 1.3 : Palette de niveaux de gris (grayscale).

- **RGB (rouge, vert, bleu) :** C'est un système de codage informatique des couleurs avec trois canaux où le premier représente le rouge, le deuxième représente le vert et le dernier représente le bleu. Chaque valeur de ces canaux varie de 0 à 255 (figure 4)

- **HSV (Teinte, saturation, luminosité) :** aussi, c'est un système de codage informatique des couleurs avec trois canaux où le premier représente la teinte, le deuxième représente la saturation et le dernier représente la luminosité, (figure1.4)

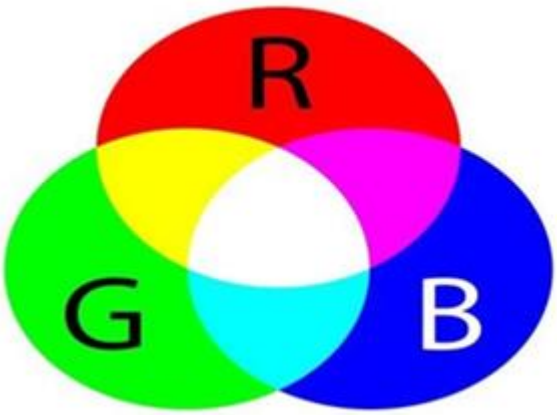

Figure 1.4 : Palette de RGB.

1.3 Définitions de vidéo :

Une vidéo est une source multimédia qui combine une série d'images à une certaine fréquence d'images FPS (l'œil humain est capable de résoudre environ 20 images par seconde), et généralement, elle a une composante audio qui correspond à l'image affichée à l'écran. Il existe deux types de vidéos :

1.3.1 Les vidéos entrelacées : ce type de vidéo est divisé en deux champs, un pour les lignes impaires de l'image et l'autre pour les lignes paires **[4]**.

1.3.2 Les vidéos numériques : ce genre de vidéo contient un seul champ qui inclut les lignes impaires et paires de l'image **[4]**.

1.4 Acquisition des données :

L'acquisition d'images constitue un des maillons essentiels de toute chaîne de conception et de production d'images. Pour pouvoir manipuler une image sur un système informatique, il est nécessaire de lui faire subir une transformation qui la rendra lisible et manipulable par ce système. La transformation de l'image réelle en une image numérique capable a interprété par le système se fait grâce à une procédure de numérisation. Ces systèmes de saisie peuvent être classés en deux types principaux : les scanners et les caméras numériques **[5].**

1.4.1 Scanner: C'est un périphérique informatique qui numérise des documents ou d'autres choses comme les empreintes digitales.

1.4.2 Camera numérique : Une caméra numérique est un appareil numérique qui utilise un capteur électronique (CCD) pour convertir les informations lumineuses en signaux électriques

Dans notre système de détection de somnolence basé sur l'apprentissage profond, l'utilisation d'une caméra pour l'acquisition d'images est essentielle pour identifier les signes de fatigue chez le conducteur. La caméra est généralement placée sur le tableau de bord ou à proximité du rétroviseur intérieur et est utilisée pour capturer des images en temps réel du visage du conducteur. Ces images sont ensuite analysées par un réseau de neurones profonds, qui sont entraînés à identifier les expressions faciales, les mouvements de la tête et les clignements d'yeux qui indiquent un état de somnolence. Grâce à cette analyse en temps réel, le système peut déclencher des alertes visuelles ou sonores pour avertir le conducteur en cas de risque de somnolence ou de perte de vigilance. La combinaison de l'acquisition d'images et de l'apprentissage profond permet de détecter les signes de fatigue de manière fiable et en temps réel, aidant ainsi à prévenir les accidents de la route.

1.5 La somnolence et la conduite

La somnolence est définie comme une diminution du niveau de conscience dépeint par la somnolence et la difficulté à rester alerte, mais la personne se réveille avec une simple excitation provoquée par les stimuli. Cela peut être dû à une absence de repos, à des médicaments, à un abus de substances ou à un problème cérébral. La somnolence est principalement le résultat de la fatigue qui peut être à la fois mentale et physique. D’autre part, n'importe quelle activité poursuit suffisamment longtemps, Causera à une difficulté à maintenir des performances qualifiées, y compris la conduite automobile. Et pour arrêter ou réduire le nombre des accidents, l'état de somnolence du conducteur devrait être surveillé en permanence.

1.5.1 Approches de détection de somnolence :

Selon les dernières recherches menées par l'Association Record Safety international Road Travel (ASIRT) **[6]**, on trouve que les accidents de la circulation sont l'une des principales causes de mort dans le monde dans lesquels 1.3 personnes sont mortes dans des accidents de voiture en 2021, soit une augmentation de 10,5 % par rapport à 2020. La situation en Algérie n'est pas différente de celle d'autres pays. Selon les statistiques fournies par la Direction générale de la Protection civile. Depuis le début

de l'année, l'on déplore en effet 1.105 morts et plus de 40.000 blessés causés par 32.200 accidents dont 12.000**[7]**.

Il existe plusieurs approches pour la détection de la somnolence chez un conducteur. L'une des méthodes les plus courantes sont mentionnées dans la figure 1.5:

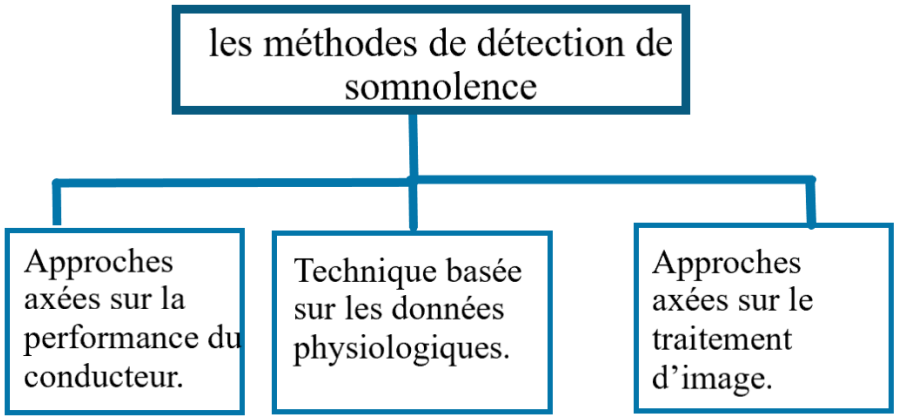

Figure 1.5 : Les différentes méthodes de détection de somnolence.

a-Approche axée sur les données physiologiques :

Quand une personne se sent somnolente ou fatiguée, de nombreux changements apparaissent sur son corps. Ces changements incluent les changements de la vitesse du rythme cardiaque, l'état du cerveau et l'état des muscles. Basant sur ces changements, plusieurs recherches et approches sont proposées. Dans ces approches, généralement, utilisent des capteurs portés par le conducteur comme on a indiqué dans la (figure 1.6), Ces capteurs permettent à capter plusieurs types des signaux EEG (l'état du cerveau), EOG (mouvement des yeux) et ECG (rythme cardiaque). Dans quelques recherches, ils ont utilisé les signaux EOG pour surveiller l'état du conducteur pendant la conduite. Ceci est fait en classifiant les mouvements des yeux à l'aide des signaux vertical et horizontal de l'EOG obtenu à partir des capteurs.

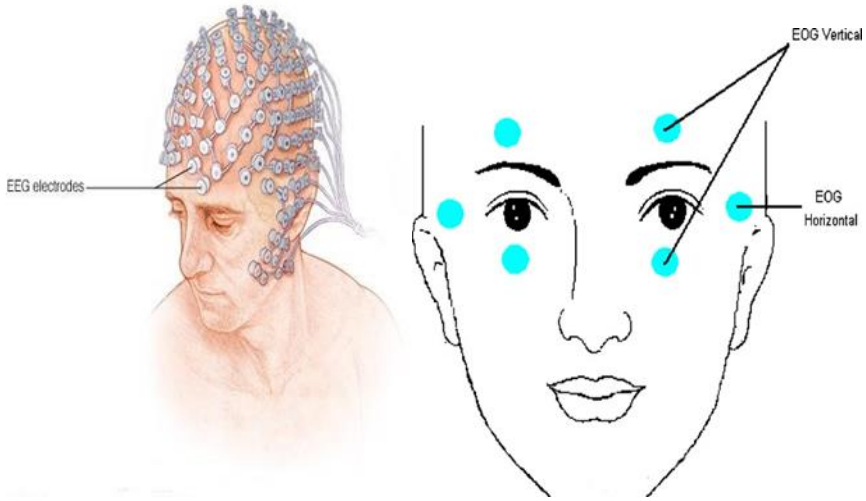

Figure 1.6 : Illustration des capteurs utilisés dans les méthodes axées sur la performance du conducteur **[8]**.

b- Approches axées sur la performance du conducteur :

Le mode de conduite est généralement influencé par une combinaison de facteurs, notamment les tâches de conduite telles que le changement de vitesses, l'accélération, la courbure et la largeur de la voie, ainsi que les caractéristiques du conducteur, telles que son expérience. En utilisant ces facteurs, il est possible de détecter l'état du conducteur. Pour cela, des capteurs sont souvent utilisés pour surveiller ces facteurs en étant placés dans différents composants de la voiture, tels que le volant et la pédale d'accélération. Plusieurs méthodes de détection ont été proposées, mais les mesures les plus couramment utilisées sont la détection des mouvements du volant et la détection de l'écart-type de la position.

Il existe plusieurs techniques basées sur les données comportementales pour la détection de la somnolence, notamment :

- L'analyse de la fréquence cardiaque : La fréquence cardiaque est un indicateur important de la vigilance et de la somnolence. Des études ont montré que la fréquence cardiaque diminue lorsque les personnes commencent à s'endormir. Ainsi, en mesurant la fréquence cardiaque, il est possible de détecter la somnolence.
- L'analyse de l'activité électrodermale : L'activité électrodermale est une mesure de la conductance de la peau, qui est influencée par l'activité du système nerveux autonome. Des études ont montré que l'activité électrodermale réduit lorsque les personnes commencent à s'endormir. Ainsi,

en mesurant l'activité électrodermale, il est possible de détecter la somnolence.

- L'analyse du mouvement des yeux : Le mouvement des yeux est pareillement un indicateur important de la somnolence. Des études ont montré que le taux de clignement des yeux réduit lorsque les personnes commencent à s'endormir. Ainsi, en mesurant le mouvement des yeux, il est possible de détecter la somnolence.
- L'analyse de la vitesse de réaction : La vitesse de réaction est un indicateur de la vigilance et de la capacité à réagir rapidement à des stimuli. Des études ont montré que la vitesse de réaction diminue lorsque les personnes commencent à s'endormir. Ainsi, en mesurant la vitesse de réaction, il est possible de détecter la somnolence.

Cette approche n'a pas été abordée dans notre projet

c- Approches axées sur le traitement d'image :

Les approches précédentes sont classifiées comme des approches intrusives, car elles influencent l'attention du conducteur. Maintenant, en raison du développement technologique, les recherches sont orientées vers les méthodes basées sur les traitements des images où on utilise une caméra pour extraire les symptômes de somnolence du conducteur comme les mouvements des yeux, les bâillements fréquents, les mouvements de tête.

Un système a été proposé **[9]** où la mesure perclose a été utilisé aussi plus d'autres cinq mesures qui sont : la durée maximale de la fermeture, la fréquence de clignement, le niveau moyen d'ouverture des yeux, la vitesse de fermeture des yeux et la vitesse d'ouverture des yeux, et pour classifier ces mesures W. Zhang et al ont utilisé le Discriminant linéaire de Fisher.

Adrian **[10]** a proposé une autre approche où après la détection du visage et ses repères, il calcule l'EAR (Eyes aspect ratio) pour la détection de l'état des yeux (ouvert ou fermé) enfin, l'alarme est lancée si les yeux sont fermés dans 48 trames Consécutives.

Reza Ghoddoosian et al **[11]** propose un autre système où ils ont utilisé seulement trois caractéristiques : la fréquence de clignement, le niveau moyen d'ouverture des

yeux et la vitesse d'ouverture des yeux et pour classifier ces mesures, ils ont utilisé les réseaux de neurones récurrents (l'architecture LSTM).

Plus récemment, certaines approches qui utilisent l'apprentissage en profondeur, par exemple l'utilisation de CNN et RNN sont apparues. L'apprentissage en profondeur basé sur les CNNs et RNNs constitue une avancée décisive, en particulier pour les tâches de vision par ordinateur tel que la classification d'images, la détection d'objets la reconnaissance des émotions, la segmentation des scènes, etc. en fonction de ce concept, plusieurs approches sont proposées pour détecter la somnolence.

1.6 Intelligence artificielle (IA)

L'intelligence artificielle est une discipline de l'informatique qui regroupe l'ensemble des théories et des techniques pour aboutir à la réalisation de machines capables de simuler l'intelligence, en opposition avec l'intelligence naturelle des êtres vivants.

L'objectif de l'intelligence artificielle est de créer des programmes qui exécutent des tâches normalement assignées à l'intelligence humaine.

De L'intelligence artificielle découle l'apprentissage machine qui lui-même comprend le domaine de l'apprentissage profond, (figure 1.7).

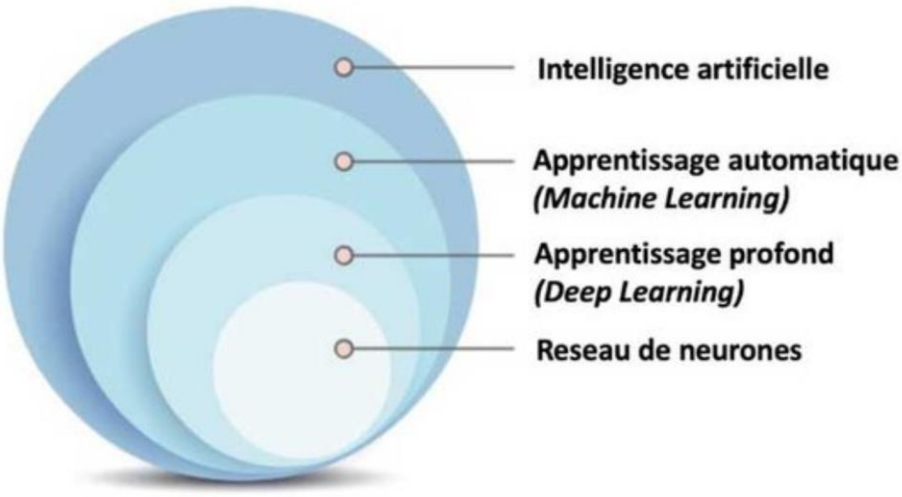

Figure 1.7 : L'intelligence artificielle.

1.6.1 Historique L’intelligence artificielle :

Depuis les années 1950, début de l'IA, les chercheurs s'efforcent de créer un système capable de contenir des données visuelles. Au cours des années qui suivent, le domaine est devenu connu sous le nom de "vision artificielle". En 2012, cette dernière a fait un pas en avant lorsqu'une équipe de chercheurs de l'université de Toronto a développé un modèle d'IA doté des meilleurs algorithmes de reconnaissance d'images **[12]**, en effet, lors d’une compétition organisée par ImageNet, un réseau de neurones est parvenu pour la première fois à surpasser un humain dans la reconnaissance d’image. Il s’agit du réseau de neurones AlexNet (du nom de son principal créateur, Alex Krizhevsky), a remporté le concours d’ImageNet 2012 avec une précision de 85 %.

1.6.2 Apprentissage machine :

L’apprentissage machine (appelé le machine Learning en anglais (ML)) est une branche de l’IA. Il a la capacité d’apprendre à partir de données en utilisant un algorithme d’apprentissage dont l’objectif est d’effectuer des analyses explicatives, prédictives ou préventives.

1.6.3 Réseaux de neurones :

Les réseaux de neurones sont inspirés de réseaux de neurones biologiques. Les réseaux de neurones artificiels permettent de simplifier des problèmes d’apprentissage machine.

Les réseaux de neurones profonds sont très utiles pour un certain nombre de tâches dans le travail humain. Ils sont utilisés dans le domaine de la vidéosurveillance de la technologie de reconnaissance faciale. Les voitures autonomes sont également basées sur cette technologie. Des assistants virtuels comme Siri et Alexa. Par conséquent, nous utilisons peut-être quotidiennement des produits qui reposent sur des réseaux de neurones profonds sans le savoir.

1.6.4 Apprentissage profond :

L’apprentissage profond (appelé le Deep Learning en anglais (DL)) est une discipline de l’apprentissage machine (appelé aussi l’apprentissage profond) qui combine des méthodes basées sur les réseaux neuronaux et des fonctions mathématiques.

L'apprentissage profond, cela veut dire qu'on va créer des réseaux de neurones avec beaucoup de couches intermédiaires. Dans l'apprentissage machine, les paramètres du modèle sont insérés manuellement, par contre dans l'apprentissage profond, ils sont déterminés par les réseaux de neurones. On peut trouver l'apprentissage profond dans plusieurs domaines comme dans le domaine médical, automobile et la télécommunication... Pour notre cas, il va être utile pour aider l'automobile à installer un système de détection de somnolence pour minimiser les accidents de la route.

a-Fonctionnement d'un réseau de neurone :

L'unité de base d'un réseau de neurone est le neurone. Il accueille en entrée des valeurs numériques (xi), ces dernières sont multipliées par des coefficients appelés les poids (wi), et il produit une sortie qui se calcule en deux étapes :

1. Un calcul d'une somme pondérée de toutes les entrées est fait, en plus d'un ajout d'une entrée ayant un poids b appelé le biais **[13]**.

2. Une fonction d'activation y, liée au seuil, calcule la valeur de sortie du neurone. On illustre ce fonctionnement dans la figure 1.8.

$$(y= f\ (\Sigma\ (xi * wi) +b))$$

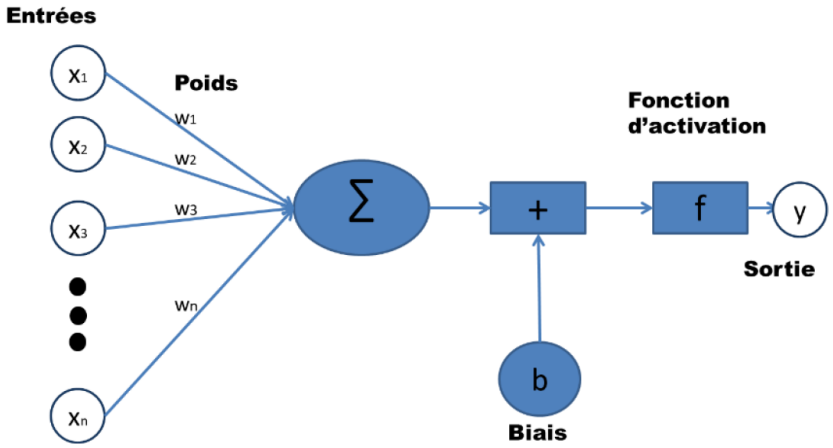

Figure 1.8 : Présentation d'un neurone.

Un réseau de neurones est composé de nombreux processeurs qui travaillent en parallèle et qui sont répartis en couches. La première couche reçoit les données brutes en entrée, Ensuite, chaque couche reçoit les données transformées par la couche précédente. La dernière couche produit les résultats du système. Par exemple, si on veut créer un réseau

de neurones qui peut classifier des images d'animaux. Le réseau prend en entrée des photos de chiens ou de chats, et en sortie, il donne la classe prédite, chien ou chat, selon ce cas (figure 1.9).

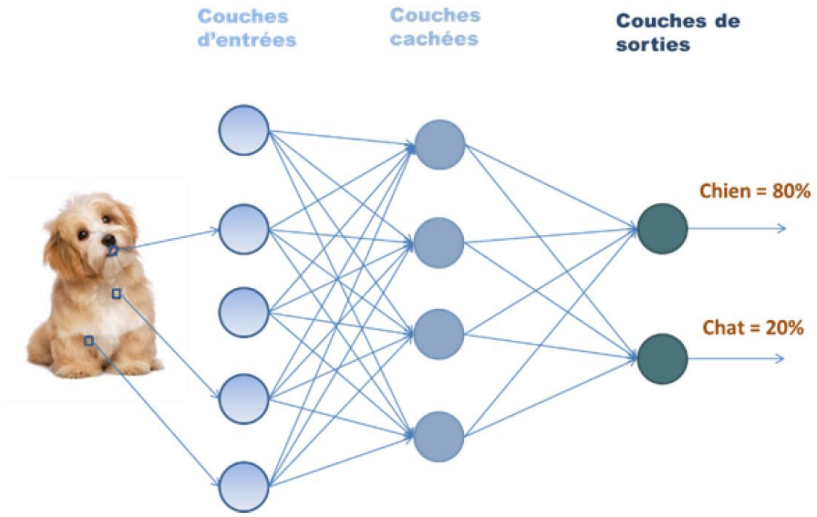

Figure 1.9 : Réseaux de neurone.

b- Les types de réseau de neurones :

On distingue différents types de réseaux de neurones. En règle générale, ils sont groupés en fonction du nombre d'épaisseurs qui séparent l'entrée de données de la production du résultat, en fonction du nombre de nœuds cachés du modèle, ou encore du nombre d'entrées et de sorties de chaque nœud, parmi les réseaux de neurones, on peut citer :

- **Réseaux de neurones récursifs :**

Les réseaux de neurones récursifs sont des réseaux de neurones artificiels qui peuvent traiter des données structurées de manière variable, comme des graphes ou des arbres. Ils utilisent le même ensemble de poids pour combiner les nœuds en parents, en suivant l'ordre topologique de la structure. Ils apprennent à produire une prédiction structurée ou scalaire sur les données, en utilisant un algorithme de rétro propagation à travers la structure. Ils sont inspirés par la mémoire auto-associative récursive (RAAM) et sont très utiles pour le traitement du langage naturel (TAL). Il existe différents types de réseaux de neurones récursifs selon la fonction de composition, le mode d'apprentissage et la modalité des données présenté dans la figure 1.10 ci-dessous.

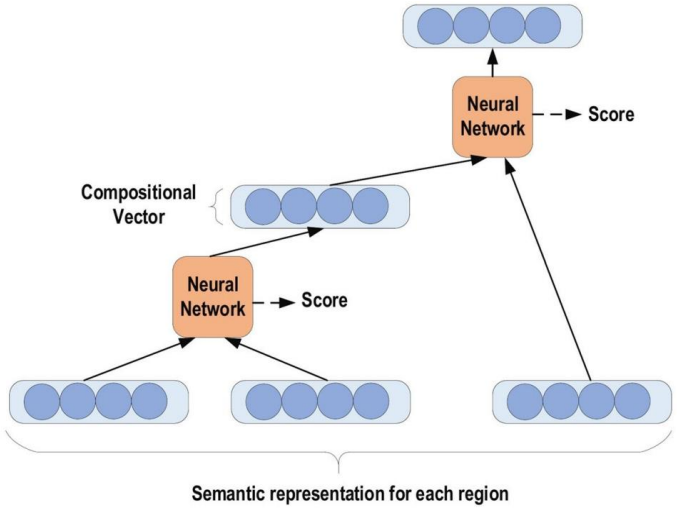

Figure 1.10 : Réseaux de neurone récursifs

- **Réseaux de neurones récurrents :**

Les réseaux de neurones récurrents sont des réseaux de neurones artificiels capables de traiter des données séquentielles, c'est-à-dire des données de taille variable qui évoluent dans le temps. Ils se caractérisent par la présence de connexions récurrentes, c'est-à-dire de connexions entre la sortie d'une couche connectée et son entrée. Ces connexions permettent au réseau de se souvenir des informations passées et de les utiliser pour influencer les décisions actuelles. Les réseaux de neurones récurrents conviennent à des tâches telles que la reconnaissance automatique de la parole, la traduction automatique ou la génération de texte. Il existe plusieurs types de réseaux de neurones récurrents selon la structure des connexions, les fonctions d'activation et les modes d'apprentissage, figure 1.11.

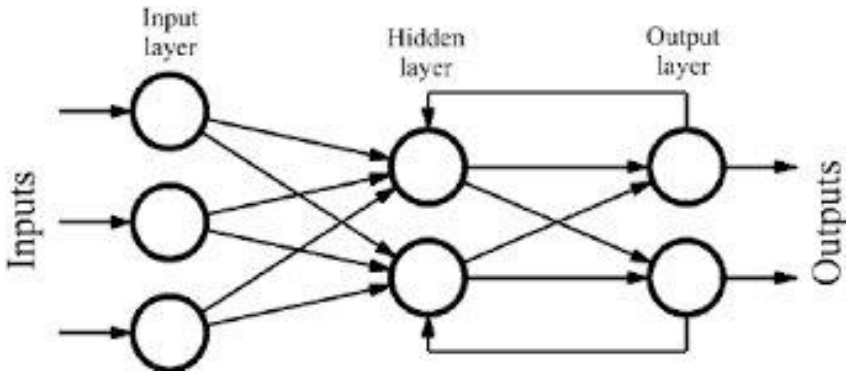

Figure 1.11 : Réseau-de-neurones-récurrents**[14].**

- **Réseau de neurones convolutifs (CNN) :**

Dans le domaine de l'apprentissage profond, le CNN est l'algorithme le plus célèbre et le plus couramment utilisé. Le principal avantage du CNN par rapport à ses prédécesseurs est qu'il identifie automatiquement les caractéristiques pertinentes sans aucune supervision humaine. Les CNN ont été largement appliqués dans un éventail de domaines différents, notamment la vision par ordinateur, le traitement de la parole, la reconnaissance faciale, etc. La structure des CNN a été inspirée par les neurones du cerveau humain et animal, similaire à un réseau neuronal classique. Plus précisément, dans le cerveau d'un chat, une séquence complexe de cellules forme le cortex visuel, cette séquence est simulée par le CNN. Goodfellow et al. **[15]** ont identifié trois avantages clés du CNN : les représentations équivalentes, les interactions clairsemées et le partage des paramètres.

c- Les caractéristiques réseaux de neurones convolutifs (CNN) :

- **Fonctionnement d'un CNN :**

Les CNN sont constitués de plusieurs couches de neurones artificiels. Ce sont des fonctions mathématiques qui calculent la somme de certaines entrées et produisent une sortie. Lorsqu'un CNN admet en entrée une image.

La première couche extrait généralement les caractéristiques de base telles que les bords horizontaux ou diagonaux **[16]**. Cette sortie est transmise à la couche suivante qui a des fonctionnalités plus complexes telles que la détection des coins ou des bords combinés. Plus le réseau est profond, plus il peut identifier des caractéristiques complexes comme des objets, des visages, etc.

La couche de classification produit un ensemble de scores de confiance (valeurs entre 0 et 1), qui spécifient la probabilité que l'image appartient à une classe. Par exemple, si vous avez un CNN qui détecte les chiens, la sortie de la couche finale est la possibilité que l'image d'entrée contienne les images des chiens, figure 1.12 représente l'architecture d'un CNN.

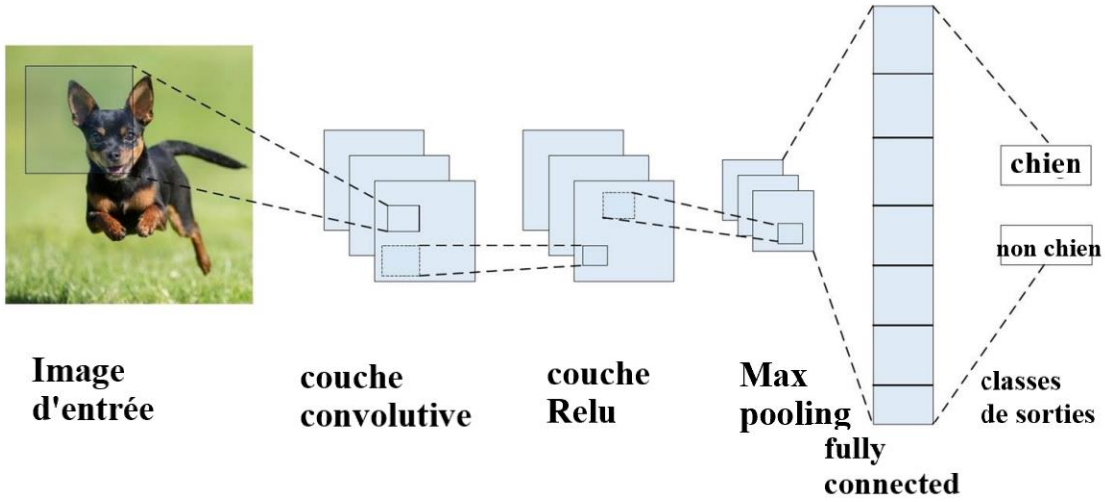

Figure 1.12 : Architecture d'un CNN.

- **Les opérations de CNN :**

Un CNN applique généralement certains types d'opérations différentes à une image afin d'extraire des informations pertinentes. Ces opérations sont les suivantes :

- Couche de convolution.
- La fonction d'activation.
- Le pooling.
- Flattening.
- Couche entièrement connecté (fully connected).

➢ **Couche convolutive :**

Pour ce faire, un filtrage de convolution est effectué le principe est de faire glisser un filtre sur l'image et de calculer le produit de convolution entre les coefficients du filtre et chaque partie de l'image balayée (Figure1.13). Par conséquent, une couche convolutive reçoit plusieurs images en entrée et calcule la convolution de chaque image avec chaque filtre. On obtient une carte d'activation pour chaque paire (image, filtre), qui nous indique où se trouvent les entiers dans l'image

- calcul de convolution :

Le schéma ci-dessus montre une opération de convolution entre une section d'image et un seul filtre. Vous pouvez obtenir des multiplications d'éléments par ligne ou par colonne, puis une sommation.

Le résultat de ce calcul est placé dans la zone correspondante de la carte des entités.

Puis, on fait un autre calcul en déplaçant le filtre sur l'image horizontalement d'un pas vers la droite. Le nombre de pas (pixels) que nous déplaçons le filtre sur l'image d'entrée est appelé Stride. Le déplacement peut se faire aussi bien horizontalement que verticalement. Ici, nous utilisons Stride=1. Stride est également un hyperparamètre qui doit être spécifié par l'utilisateur, (figure 1.13).

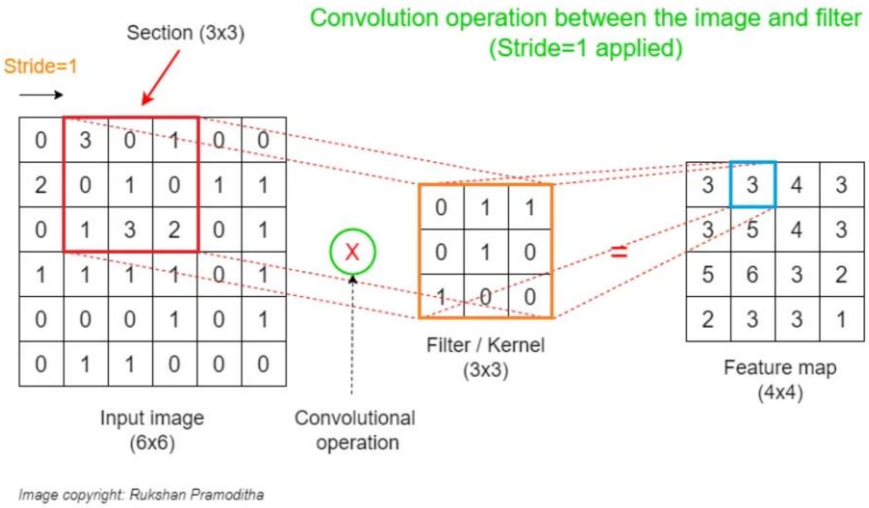

Figure 1.13: Calcul de convolution avec Stride=1.

- **Fonction Relu :**

La fonction Relu (Rectified Linear Unit) est une fonction d'activation non linéaire couramment utilisée dans les réseaux de neurones artificiels. Elle est définie mathématiquement comme suit :

$F(x) = \max(0, x)$

Où x est la valeur d'entrée et max (0, x) renvoie la valeur maximale entre 0 et x. Autrement dit, la fonction Relu renvoie x si x est positif, sinon elle renvoie 0.

En effet, la fonction Relu a une dérivée constante de 1 pour $x > 0$, ce qui permet de conserver un gradient suffisamment grand pour la rétro propagation de l'erreur lors de l'apprentissage du réseau, (figure 1.14).

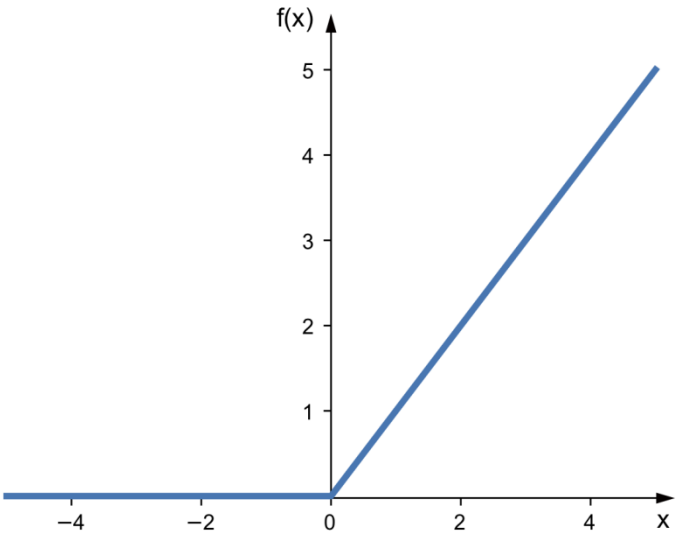

Figure 1.14 : Fonction d'activation Relu.

➢ **Couche pooling :**

La couche de pooling est responsable de la réduction de la taille spatiale de la caractéristique convoluée. Cela réduit la puissance de calcul nécessaire pour traiter les données en réduisant les dimensions. Il existe deux types de pooling : le pooling moyen et le pooling maximum.

- **Objectifs :**

- Extraction des fonctionnalités les plus importantes (pertinentes) en obtenant le nombre maximum ou en faisant la moyenne des nombres.
- réduction de la dimensionnalité (nombre de pixels) de la sortie renvoyée par les couches convolutionnelles précédentes.
- Suppression de tout bruit présent dans les caractéristiques extraites par les couches convolutionnelles précédentes.
- Augmentation de précision des CNN.

➢ **Fonction pooling :**

Il existe deux types de pooling :

- Fonction max pooling :

Le rôle de Max Pooling, c'est de retrouver la valeur maximale d'un pixel dans une partie de l'image couverte par le noyau. Le Max Pooling agit également comme un

suppresseur de bruit et effectue également une réduction de volume ainsi qu'une réduction de taille.

- Fonction average pooling :

I'Average Pooling renvoie la moyenne de toutes les valeurs de la partie de l'image couverte par le noyau. La performance moyenne globale est simplement une réduction de la dimensionnalité en tant que mécanisme pour éliminer le bruit. Par conséquent, le Max Pooling est beaucoup plus performant que l'average pooling (figure1.15).

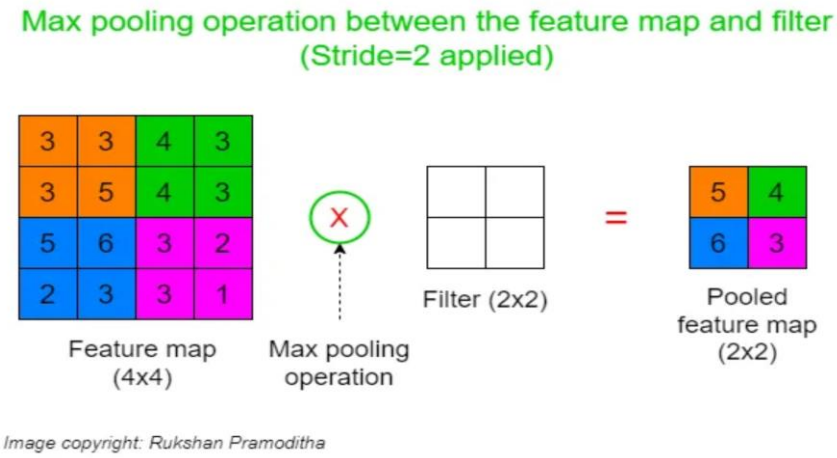

Figure 1.15 : Max Pooling.

- **Opération flatten :**

Dans un CNN, la sortie renvoyée par la couche de pooling finale est transmise à un Perceptron multicouche (MLP) qui peut classer la carte d'entités de pooling finale dans une étiquette de classe, (figure 1.16).

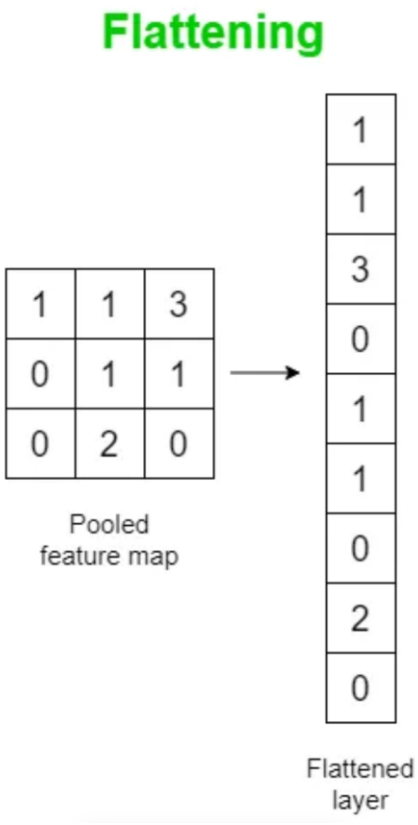

Figure 1.16 : Flattening.

- **Fully connected layers (denses) :**

Ce sont les dernières couches d'un CNN. L'entrée est la couche aplatie précédente (Flatten). Il peut y avoir plusieurs couches entièrement connectées (Fully connected). La dernière couche effectue la tâche de classification (ou autre tâche pertinente). Une fonction d'activation est utilisée dans chaque Fully connected layers. Pami de ces objectifs est de classer les entités détectées dans l'image dans une étiquette de classe, (figure 1.17).

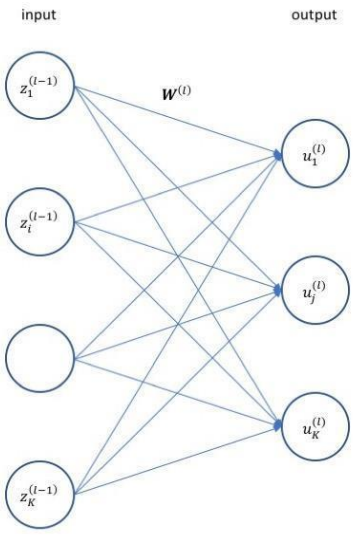

Figure1.6 : Fully connected layers.

1.7 Conclusion

- Dans ce chapitre, nous avons abordé quelques notions de bases et certaines définitions élémentaires concernant le traitement des images et des vidéos après l'acquisition pour les utiliser par
- la suite dans notre système, où nous avons défini l'image numérique et la vidéo, puis nous avons cité leurs caractéristiques (pixel, résolution, taille et espace de couleurs), et enfin les outils d'acquisition.
- Nous avons présenté les différents systèmes de détection automatique de somnolence du conducteur.
- En dernier, nous avons vu comment l'apprentissage profond et les réseaux de neurones convolutifs (CNN) ont révolutionné le domaine de l'intelligence artificielle et de la vision par ordinateur.

Chapitre 2 : Détection de somnolence avec deep learning

2.1 Introduction

La détection de la somnolence au volant peut se faire par différents moyens, tels que l'analyse du comportement du conducteur, la mesure de ses signaux physiologiques ou la surveillance de son environnement. Parmi ces moyens, l'utilisation des réseaux de neurones convolutifs (CNN) pour analyser les images du visage du conducteur présente plusieurs avantages. Les CNN sont des modèles d'apprentissage automatique qui peuvent apprendre à reconnaître des motifs complexes dans les données visuelles, comme les expressions faciales ou les mouvements oculaires. Les CNN peuvent donc détecter des signes de somnolence au volant, tels que le bâillement, le clignement des yeux ou le regard fixe. De plus, les CNN sont capables de s'adapter à différents contextes, comme les variations de luminosité ou d'angle de vue, et de traiter les images en temps réel.

L'objectif de ce travail est de présenter les principes, les méthodes et les applications des CNN pour la détection de la somnolence au volant. Nous verrons comment les CNN sont construits, entraînés et évalués, ainsi que les défis et les perspectives qu'ils soulèvent.

2.2 Présentation du problème de détection de somnolence par deep learning :

La détection de somnolence par deep learning est un problème qui consiste à utiliser des techniques d'apprentissage automatique profond pour analyser des données multimodales (voix, visage, comportement, etc.) et identifier les signes de somnolence chez une personne. Le deep learning utilise des réseaux de neurones artificiels capables d'apprendre à partir de grandes quantités de données et de réaliser des tâches

complexes. La détection de somnolence par deep learning présente plusieurs avantages, comme la possibilité de traiter des données non structurées de capturer des caractéristiques subtiles et de s'adapter à différents contextes et individus. Elle présente aussi des défis, comme la nécessité de disposer de données suffisantes et variées, de choisir les architectures et les paramètres des réseaux de neurones adaptés, ou encore d'interpréter les résultats obtenus.

2.3 Définition de la somnolence et de ses causes :

La somnolence est un état intermédiaire entre la veille et le sommeil, qui se caractérise par un besoin de dormir non désiré et parfois incontrôlable. La somnolence s'accompagne d'une diminution de la vigilance, du tonus musculaire et du rythme cardiaque[24]. La somnolence est normale quand elle survient le soir ou après le déjeuner, mais elle peut être anormale et excessive quand elle se manifeste à tout moment de la journée. La somnolence peut avoir des causes diverses, comme un manque de sommeil, un trouble du rythme circadien, une apnée du sommeil, ou certains médicaments. La somnolence peut avoir des conséquences négatives sur la santé, la performance et la sécurité.

2.3.1 Les facteurs qui favorisent la somnolence :

Les facteurs qui favorisent la somnolence sont nombreux et variés. Parmi eux, on peut citer :

- L'insuffisance ou l'irrégularité de sommeil, qui entraîne une dette de sommeil et une perturbation du rythme circadien.
- Le stress, l'anxiété, la dépression, qui peuvent altérer la qualité du sommeil et provoquer des réveils nocturnes.
- Certains médicaments, comme les antihistaminiques, les antidépresseurs, les anxiolytiques, les hypnotiques, qui peuvent avoir un effet sédatif.
- Certains troubles du sommeil, comme l'apnée du sommeil, le syndrome des jambes sans repos, le syndrome de retard de phase, qui empêchent un sommeil réparateur.
- Un repas trop riche ou trop copieux, qui augmente la production d'insuline et favorise l'endormissement.

- L’alcool ou le tabac, qui modifient le cycle du sommeil et réduisent la vigilance.
- La chaleur ou le bruit, qui diminuent le confort et la relaxation nécessaires au sommeil

2.3.2 Les conséquences de la somnolence sur la santé et la sécurité :

La somnolence peut avoir des conséquences graves sur la santé et la sécurité des personnes qui en souffrent et de leur entourage. Parmi ces conséquences, on peut citer :

- Une augmentation du risque d’accidents de la route, car la somnolence diminue la vigilance, l’attention, la réactivité et la capacité à maintenir une trajectoire.
- Une altération des performances scolaires ou professionnelles, parce que la somnolence réduit la concentration, la mémoire, le raisonnement et la créativité.
- Des complications médicales, comme l’hypertension artérielle, les maladies cardiovasculaires, le diabète ou l’obésité, qui peuvent être favorisées ou aggravées par un manque de sommeil chronique.

2.4 Contexte et motivation de la détection de la somnolence :

L'objectif de ce projet est de développer une nouvelle méthode de détection de la somnolence basée sur l'analyse des expressions faciales telles que la somnolence et le bâillement. Nous proposons d'utiliser un réseau de neurones convolutif (CNN) pour extraire les caractéristiques des expressions facials pour la détection de la somnolence. Un CNN est un type de réseau de neurones artificiels qui peut apprendre à reconnaître des motifs complexes dans des données multidimensionnelles.

2.4.1 Les domaines où la détection de la somnolence est utile :

La détection de la somnolence est utile ou nécessaire dans plusieurs domaines auxquels la vigilance et la performance sont essentielles, parmi ces domaines, on peut citer :

- La sécurité routière, car la somnolence est un facteur de risque majeur d’accidents de la route, qui peuvent être évités par des systèmes d’alerte ou de prévention embarqués dans les véhicules.

- La santé au travail, parce que la somnolence peut affecter la productivité, la qualité et la sécurité des travailleurs, notamment ceux qui exercent des métiers à horaires décalés ou à forte charge cognitive.
- La médecine du sommeil, car la somnolence peut être le symptôme d'un trouble du sommeil ou d'une pathologie sous-jacente, qui nécessite un diagnostic et un traitement adaptés.

2.4.2 Les objectifs et les contraintes de la détection de la somnolence :

➢ Les objectifs de la détection de la somnolence sont de :

- Identifier les signes précurseurs d'un endormissement au volant ou dans d'autres situations à risque.
- Prévenir les accidents de la route ou du travail liés à la somnolence.
- Améliorer la qualité de vie et la performance des personnes souffrant de somnolence.

➢ Les contraintes de la détection de la somnolence sont de :

- Être fiable, valide et sensible aux variations de la somnolence.
- Être non invasive, confortable et respectueuse de la vie privée.
- Être facile à utiliser, à interpréter et à intégrer dans les systèmes existants.
- Être adaptée au contexte, à l'individu et à la tâche.
- Être éthique, responsable et conforme aux normes légales.

2.5 Système de la détection de somnolence

Nous avons développé deux méthodes de détection de somnolence :

2.5.1 Méthode 01 : utilisation d'un modèle CNN :

Notre première méthode implique la préparation d'un ensemble de données soigneusement sélectionné qui comprend des échantillons d'instances somnolentes et non somnolentes. Ensuite, en utilisant Keras et TensorFlow, nous créons et entraînons une architecture de réseau neuronal profond personnalisée en tenant compte des caractéristiques et des exigences de détection de somnolence spécifiques. Le modèle est entraîné sur cet ensemble de données pour apprendre les schémas et les caractéristiques qui différencient les états de sommeil et d'éveil.

En utilisant des mesures appropriées telles que la perte, la précision, le temps de réponse, pour évaluer les performances de notre modèle.

2.5.2 Présentation du système de détection de somnolence par l'architecture de CNN :

L'organigramme présenté ci-dessus (figure2.1) illustre la première méthode de détection de la somnolence utilisant l'architecture du réseau de neurones convolutifs (CNN).

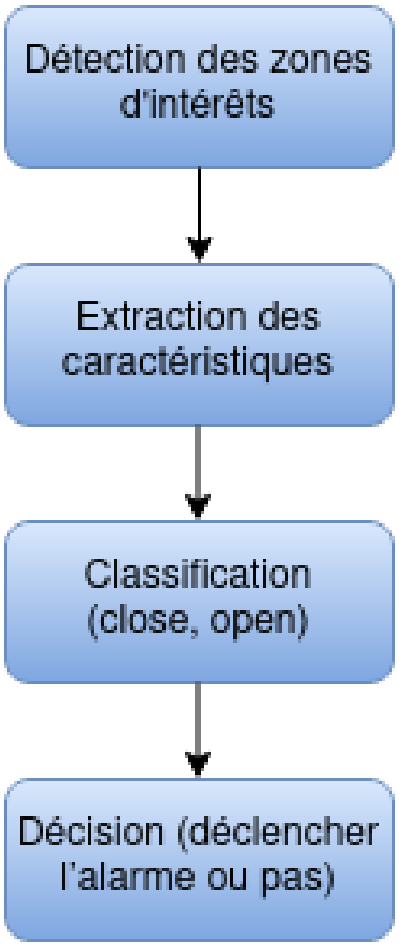

Figure 2.1 : Système de détection de somnolence.

a-Détection des zones d'intérêts :

Pour faire La détection de visage et des yeux, nous avons utilisé une architecture à base de CNN pour extraire des caractéristiques propres à chaque image en les compressant de façon à réduire leur taille initiale. Un CNN se composant d'une partie convolutive (applique des filtres à l'image fournie en entrée, créant ainsi des cartes de convolutions qui représentent les traits saillants du visage et des yeux.) et d'une partie classification (utilise ces cartes de convolutions pour attribuer à l'image une étiquette correspondant à sa classe d'appartenance, par exemple la présence ou l'absence de

visage ou d'yeux.). La détection de visage et des yeux par CNN est utilisée pour diverses applications, telles que la sécurité, la biométrie, le traitement médical ou l'analyse des émotions.

b-Extraction des caractéristiques :

L'extraction des caractéristiques telles que les yeux, le nez… est une étape de prétraitement nécessaire à la reconnaissance faciale. On peut distinguer deux pratiques différentes : la première repose sur l'extraction des régions entières du visage, elle est souvent implémentée avec une approche globale de reconnaissance de visage. La deuxième pratique extrait des points particuliers des différentes régions caractéristiques du visage, tels que la bouche et du nez (ROI).Elle est utilisée avec une méthode locale de reconnaissance et aussi pour l'estimation de la pose du visage, (figure 2.2).

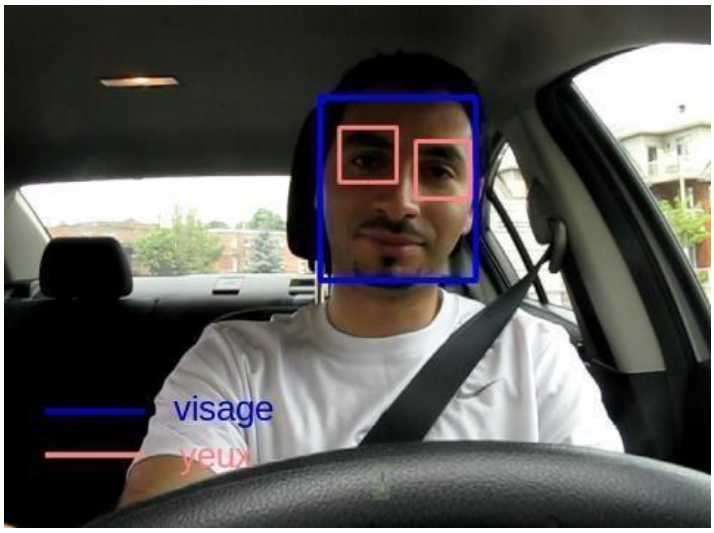

Figure 2.2 : Exemple d'extraction de ROI de visage et des deux yeux.

c- Classification (close, open) :

La classification (close, open) dans le système de détection de somnolence par CNN est une technique qui permet de reconnaître l'état de vigilance d'un conducteur à partir de l'analyse de son visage. Le système utilise un réseau de neurones convolutif (CNN) pour extraire les caractéristiques pertinentes des images du visage capturées par une caméra. Le CNN est entraîné à classifier les images en deux catégories : close (yeux fermés), open (yeux ouverts), par des classificateurs de type **haar**.

- **Haarcascade:**

Haarcascade est un algorithme basé sur les caractéristiques pour la détection d'objets qui a été proposé en 2001 par Paul Viola et Michael Jones dans leur article

"Rapid Object Detection using a Boosted Cascade of Simple Features". L'implémentation originale est utilisée pour détecter le visage frontal et ses caractéristiques comme les yeux, le nez et la bouche. Haarcascade utilise des caractéristiques rectangulaires, similaires à des noyaux de convolution, pour calculer la différence d'intensité entre les régions blanches et noires de l'image. Par exemple, une caractéristique rectangulaire peut mesurer la différence d'intensité entre la région des yeux et la région des joues du visage. Haarcascade utilise une méthode appelée image intégrale pour accélérer le calcul des caractéristiques en utilisant seulement quatre pixels comme illustré dans la figure 2.3. Haarcascade entraîne un classificateur à partir de nombreuses images positives (avec des visages) et négatives (sans visages) en utilisant l'algorithme Adaboost. Le classificateur est ensuite capable de détecter les visages et les yeux dans d'autres images en utilisant la bibliothèque Open cv.

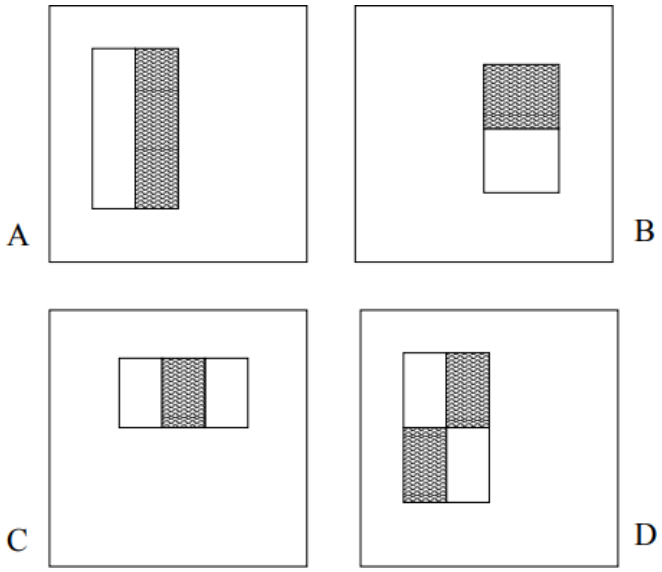

Figure 2.3 : Illustration de Haarcascade.

d- Décision :

Une fois que les yeux ont été classifiés à l'aide d'un classifieur Haar Cascade, le système décide d'activer l'alarme s'il détecte de la somnolence, sinon il ne se déclenche pas en cas de réveil.

2.5.3 Collecte et exploitation de la base des données :

L'un des éléments clés pour développer notre système de détection de somnolence par CNN est la collecte et la préparation de données de haute qualité. Cependant, rassembler des données de somnolence peut être un processus long et fastidieux. C'est là

qu'intervient **Kaggle**, une plateforme en ligne qui permet aux chercheurs et aux développeurs de partager et d'accéder à des ensembles de données pour la recherche et le développement, (figure 2.4).

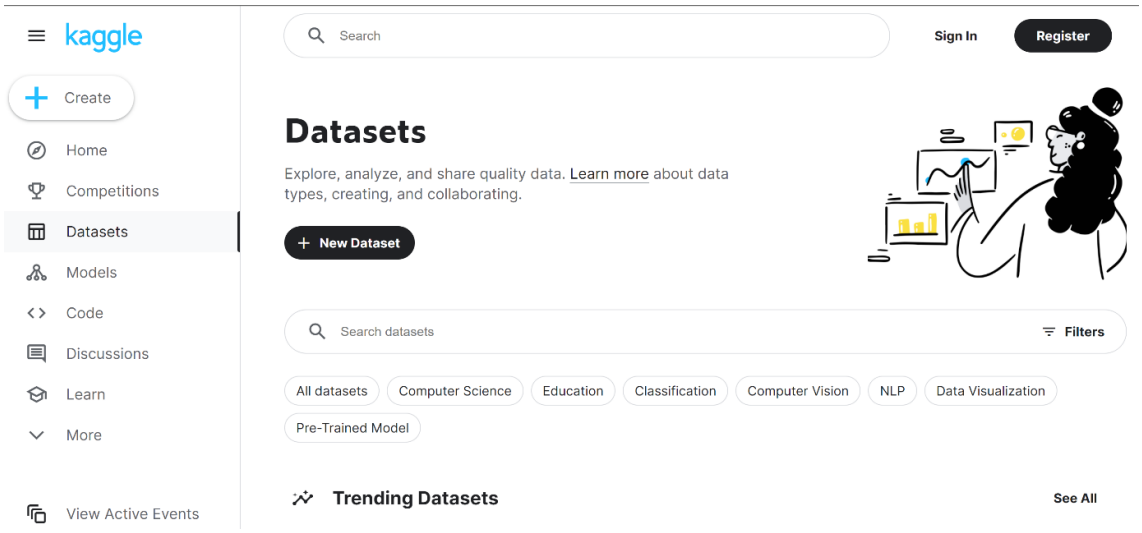

Figure 2.4 : Plateforme de Kaggle.

a- Base de données utilisée :

Pour analyser la détection de la somnolence, nous avons exploité une base de données de 2900 images sous le nom de « **yawn_eye_dataset_new** ». Nous l'avons séparée en deux parties : une partie pour l'entraînement et une partie pour le test. La partie entraînement comprend 2467 images, réparties en quatre catégories : 617 images d'yeux fermés, 617 images d'yeux ouverts, 617 images de visage avec la bouche ouverte et 616 images de visage avec la bouche fermée. La partie test comprend 433 images, également réparties en quatre catégories : 109 images d'yeux fermés, 109 images d'yeux ouverts, 106 images de visage avec la bouche ouverte et 109 images de visage avec la bouche fermée **[16].** Nous avons sélectionné uniquement les fichiers train et tests des yeux ouverts et fermés pour entraîner et évaluer notre modèle de détection de la somnolence (figure 2.5).

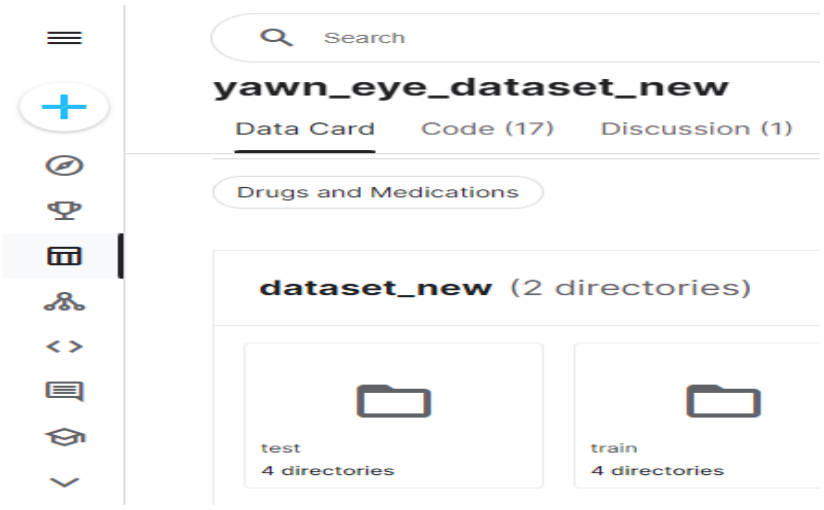

Figure 2.5 : Base de données utilisée.

2.5.4 Sélection de l'architecture de réseau de neurones :

Nous avons choisi l'architecture CNN pour le développement du système de détection de somnolence, car elle présente plusieurs avantages par rapport aux autres méthodes. Tout d'abord, l'architecture CNN est capable d'extraire des caractéristiques pertinentes des images du visage sans avoir besoin d'un prétraitement complexe. Ensuite, l'architecture CNN est robuste aux variations de luminosité, de pose et d'expression faciale, ce qui est essentiel pour un système de détection de somnolence en temps réel. Enfin, l'architecture CNN peut être facilement entraînée et optimisée avec des techniques de deep learning, ce qui permet d'améliorer les performances du système.

2.5.5 Mise en place et configuration de l'environnement de deep learning :

Dans notre projet, nous avons choisi d'utiliser le langage de programmation Python pour ses fonctionnalités avancées en matière de deep learning, sa flexibilité, sa compatibilité avec une grande variété de plateformes, sa grande communauté de développeurs et ses outils de visualisation de données avancés.

a- Langage python :

Python est un langage de programmation de haut niveau utilisé pour la programmation générale. Créé par Guidovan Rossum et Sortien 1991, Python a une philosophie de conception qui met l'accent sur la lisibilité du code, notamment en utilisant des espaces importants. Il fournit des constructions qui permettent une

programmation claire à petite et à grande échelle. Python dispose d'un système de type dynamique et d'une gestion automatique de la mémoire. Il prend en charge de multiples paradigmes de programmation, y compris orienté objet, impératifs, fonctionnels et procéduraux, et dispose d'une bibliothèque standard vaste et complète. Les interpréteurs de Python sont disponibles pour de nombreux systèmes d'exploitation.

Python est un langage très utilisé pour implémenter des CNN, car il offre de nombreuses bibliothèques et modules spécialisés dans le domaine du deep learning, tels que TensorFlow, PyTorch, Keras, etc. Ces outils permettent de créer facilement des modèles de CNN performants et personnalisables, en utilisant des réseaux pré-entraînés ou en créant ses propres architectures (figure 2.6).

Figure 2.6 : Le Logo de python.

b- Environnement de développement :

Nous avons choisi d'utiliser VS Code comme environnement de développement pour notre projet de CNN pour sa popularité, sa compatibilité avec de nombreux langages de programmation, ses fonctionnalités avancées pour le développement de logiciels, sa grande communauté de développeurs et son intégration avec des outils de développement externes.

"Visual Studio Code" est un environnement de développement intégré (IDE) développé gratuitement et de manière ouverte par Microsoft. Il est utilisé principalement pour le développement de logiciels, et prend en charge une grande variété de langages de programmation tels que Python, C++, Java, JavaScript, et bien d'autres. Ce programme vise à fournir une interface utilisateur simple et intuitive, tout en offrant des fonctionnalités avancées telles que la coloration syntaxique, l'intégration Git, la détection d'erreurs et le débogage. VS Code est très populaire auprès des développeurs grâce à son extensibilité, ce qui permet aux utilisateurs d'ajouter des fonctionnalités supplémentaires en installant des extensions, en faisant de Visual studio code un environnement de développement très personnalisable selon les besoins de chaque utilisateur, (Figure2.7).

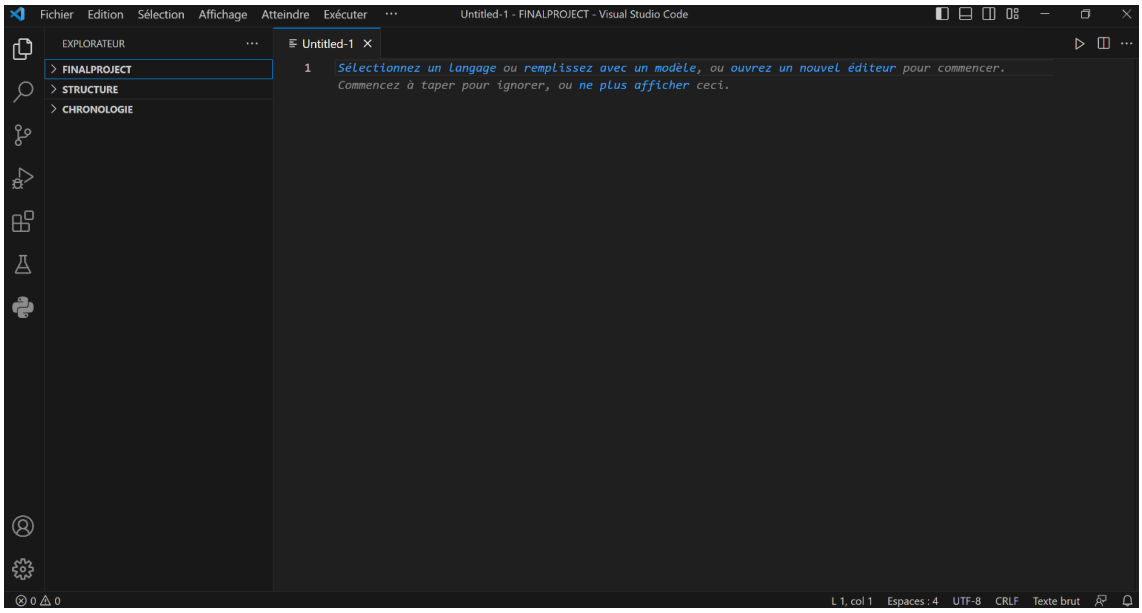

Figure 2.7 : Espace de travail de Visual studio.

2.5.6 Les bibliothèques utilisées :

Dans ce programme en a utilisé plusieurs bibliothèques python pour la détection des yeux et de visage D'une personne dans une image capturée en temps réel depuis la caméra. Voici une brève explication de chaque bibliothèque utilisée dans le programme.

a- Open CV (Open Source Computer Vision Library):

Open CV (pour Open Computer Vision) est une bibliothèque libre de fonctions de programmation principalement pour la vision par ordinateur en temps réel. Elle a été initialement développée par Intel, puis soutenue par Willow Garage, puis Itseez (qui a été ensuite rachetée par Intel). Open CV offre une infrastructure commune pour les applications de vision par ordinateur et accélère l'utilisation de la perception par machine dans les produits commerciaux. Open CV dispose de plus de 2500 algorithmes optimisés, qui comprennent un ensemble complet d'algorithmes classiques et de pointe en vision par ordinateur et en apprentissage automatique. Les algorithmes de vision par ordinateur sont des outils informatiques qui permettent de traiter des images ou des vidéos pour en extraire des informations utiles. Ces informations peuvent être de diverses natures, comme la détection de visages ou d'objets, la classification des actions humaines, la reconnaissance de mouvements de caméra ou d'objets en mouvement, l'extraction de modèles 3D d'objets, la création de nuages de points 3D à partir de caméras stéréo, l'assemblage d'images pour produire une image haute définition d'une

scène entière, la recherche d'images similaires dans une base de donnée .Ces outils sont utilisés dans de nombreux domaines tels que la sécurité, la surveillance, la reconnaissance de formes, la robotique, la réalité augmentée, etc.

- **La détection de visage et yeux avec Open CV :**

Comme mentionné plus haut, la première étape de la reconnaissance de visages est leur détection. La bibliothèque Open CV rend assez facile la détection de visage de face dans une image en utilisant son détecteur "Haar Cascade" (également connu sous le nom de la méthode Viola-Jones). L'algorithme de Viola et Jones, qui est une méthode de détection d'objet (détection de visage, détection de personne, détection d'avion, détection de voiture) dans une image numérique, est implémenté dans Open CV.

Pour la détection de visage de face, nous pouvons utiliser l'un des classificateurs Haar Cascade qui viennent avec Open CV, chacun de ces classificateurs va donner des résultats légèrement différents en fonction de l'environnement sur lequel l'on les utilise, (Figure 2.8).

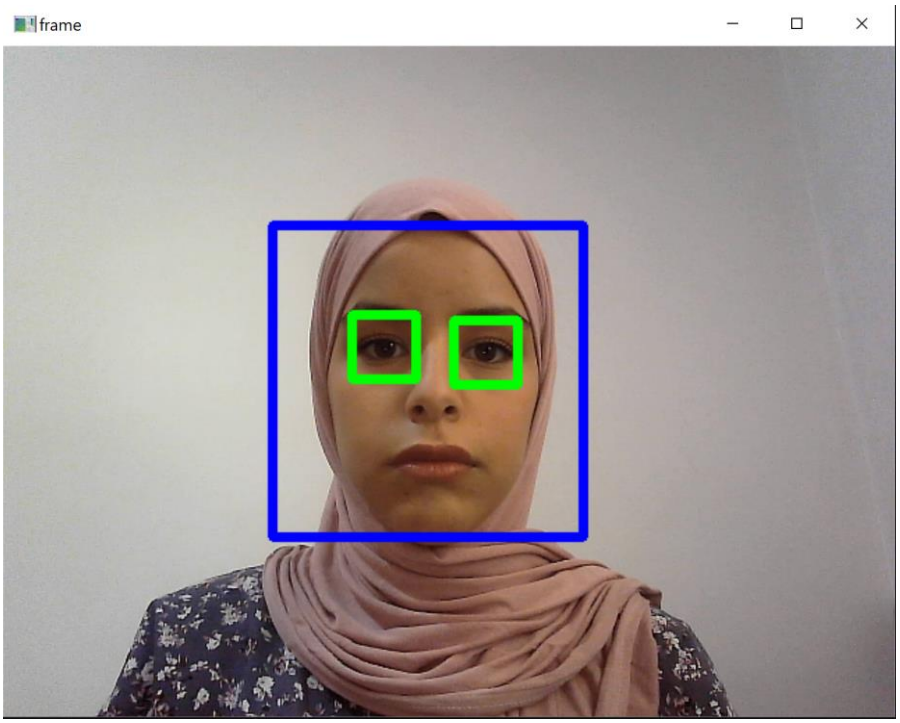

Figure 2.8 : Détection de visage et yeux.

b- Pygame :

C’est une bibliothèque pour la création de jeux en Python. Elle est utilisée ici pour jouer un son d'alarme en cas de détection de somnolence.

c- TensorFlow :

TensorFlow est une bibliothèque de machine learning open source, développée par Google, qui permet de créer et d'exécuter des applications de machine learning et de deep learning. TensorFlow offre une plate-forme de bout en bout pour le machine learning, avec des outils pour préparer les données, créer des modèles, les déployer et les maintenir. TensorFlow propose également des modèles pré-entraînés ou personnalisables, ainsi que des bibliothèques et des extensions spécialisées pour enrichir TensorFlow de nouvelles fonctionnalités.

TensorFlow est utilisé pour résoudre des problèmes concrets avec le machine learning, tels que la détection et reconnaissance d'images, le traitement du langage naturel, la génération de texte, la recommandation ou le renforcement.

d-Keras :

Keras est une bibliothèque de haut niveau pour le deep learning, qui permet de créer et d'entraîner des modèles de réseaux neuronaux de manière simple et intuitive. Elle est basée sur TensorFlow, le Framework de calcul numérique développé par Google. Keras offre plusieurs avantages par rapport aux autres bibliothèques de deep learning :

- Elle est conviviale : elle propose une interface claire et cohérente, optimisée pour les cas d'utilisation courants. Elle fournit également des messages d'erreur clairs et explicites.

- Elle est modulaire et composable : les modèles Keras sont construits en assemblant des blocs configurables, avec peu de contraintes.

Il est possible de créer des couches, des métriques et des fonctions de perte personnalisées. Keras peut être utilisée pour différents types de problèmes de deep learning, tels que la classification d'images, la génération de texte, le traitement du langage naturel ou la recommandation. Elle dispose d'une documentation complète et de nombreux tutoriels pour apprendre à l'utiliser efficacement, (figure 2.9).

Figure 2.9 : Logo de Keras.

- **Les couches de Keras :**

Keras propose une large variété de types de couches prédéfinies. Parmi les principales, on peut citer Dense, Activation, Dropout.

- Les différentes couches de convolution quant à elles vont de 1D à 3D et incluent les variantes les plus communes pour chaque dimensionnalité. La convolution 2D, inspirée par le fonctionnement du cortex visuel, est couramment utilisée pour la reconnaissance des images.
- Les couches de Pooling vont également de la 1D à la 3D et incluent les variantes communes comme le max pooling et l'average pooling. Les couches connectées localement agissent comme des couches de convolution, mais leurs poids ne sont pas partagés.

E-Matplotlib :

Matplotlib est une bibliothèque Python qui permet de créer et de visualiser des graphiques à partir de données. Elle offre une interface de haut niveau pour dessiner des figures statiques ou interactives, ainsi que des outils de bas niveau pour personnaliser les éléments graphiques. Matplotlib peut être utilisée dans des scripts Python, des notebooks Jupyter, des applications web ou des interfaces graphiques.

2.5.7 Développement de modèle CNN :

L'organigramme ci-dessus (figure 2.10) présente le développement de modèle CNN pour la détection de somnolence.

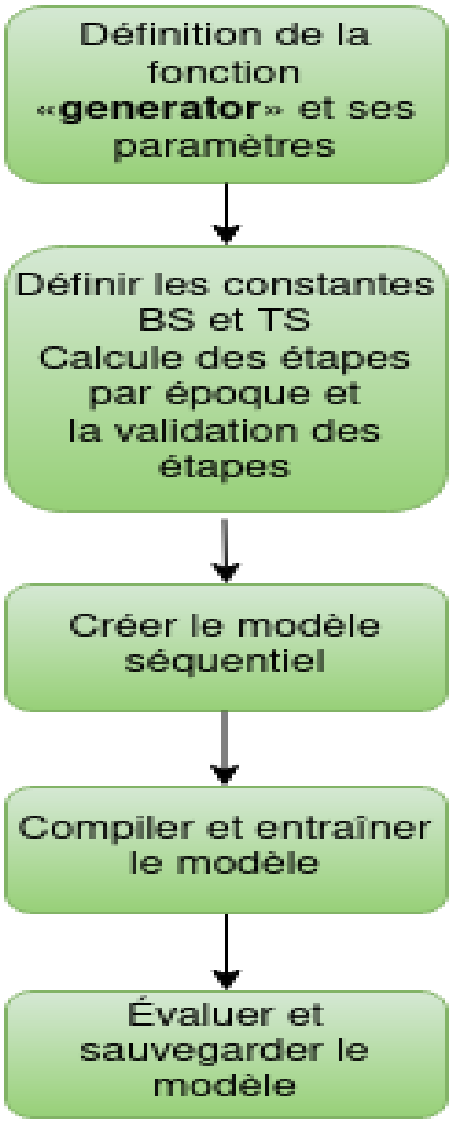

Figure 2.10 : Organigramme de notre modèle

a-L 'architecture de modèle CNN :

Ce modèle est un réseau de neurones convolutifs séquentiel qui utilise la bibliothèque Keras pour la classification d'images, il est présenté dans le digramme suivant figure 2.11:

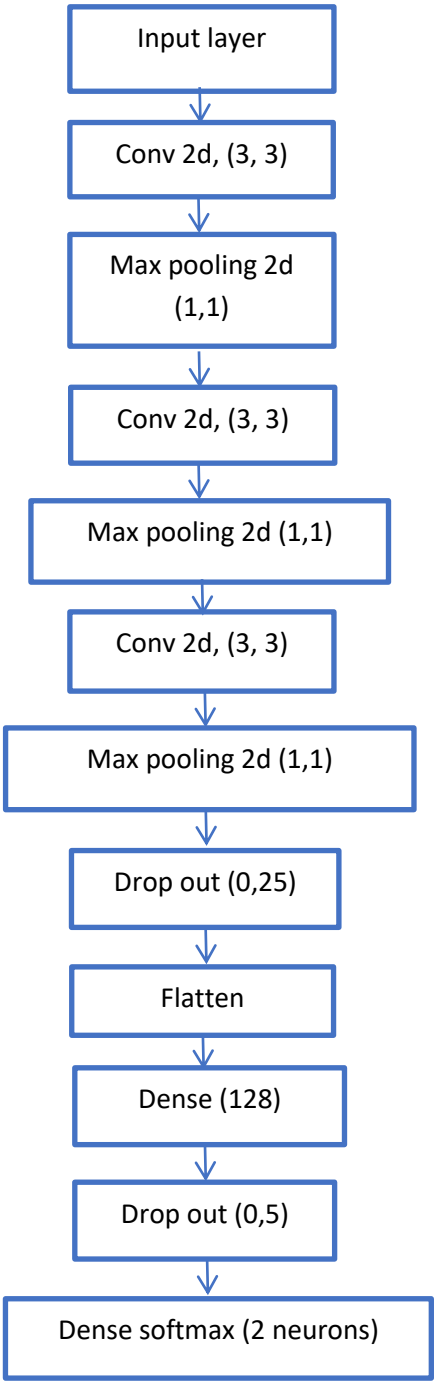

Figure 2.11 : Architecture de notre modèle.

- Il prend en entrée des images en niveaux de gris de taille 24x24 pixels et renvoie une probabilité pour chacune des deux classes possibles.

- Il se compose de trois couches de convolution avec des filtres de taille 3x3 et des fonctions d'activation Relu, suivies chaque une d'une couche de max-pooling avec une taille de pool de 1x1.

- Il utilise ensuite une couche de dropout avec un taux de 0.25 pour réduire le surapprentissage, puis une couche de flatten pour aplatir les données en un vecteur.

- Il ajoute ensuite une couche dense avec 128 neurones et une fonction d'activation ReLU, suivie d'une autre couche de dropout avec un taux de 0.5.

- Enfin, il termine par une couche dense avec deux neurones et une fonction d'activation softmax pour produire les probabilités des classes.

- Il compile le modèle avec l'optimiseur Adam, la fonction de perte entropie croisée catégorielle et la métrique accuracy.

- Il entraîne le modèle avec les données générées par la fonction generator, en utilisant une taille de batch de 32, un nombre d'époques de 20 époques et une validation croisée.

- Il utilise également un objet History pour enregistrer l'historique de l'apprentissage du modèle, (figure 2.12).

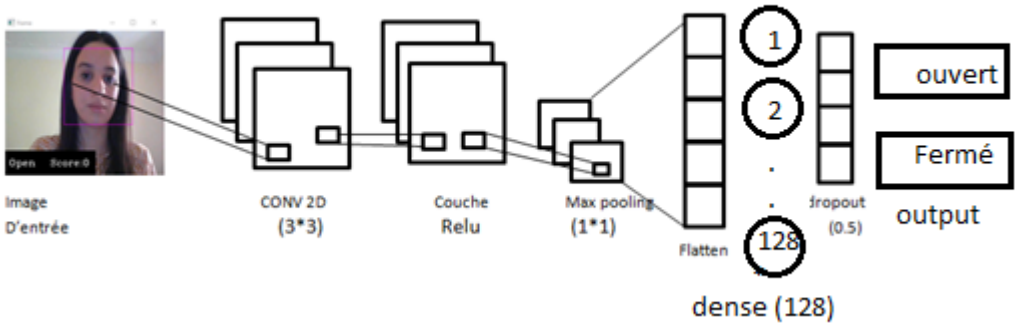

Figure 2.12: Les couches de l'architecture de modèle CNN.

b- Fonctionnement de programme principal :

Une fois que le modèle a été exécuté, nous avons sauvegardé le modèle entraîné sous forme d'un fichier .h5. Ensuite on l'a téléchargé dans le code principal, où nous avons effectué des prédictions sur la somnolence déclencher une alarme, (figure2.13).

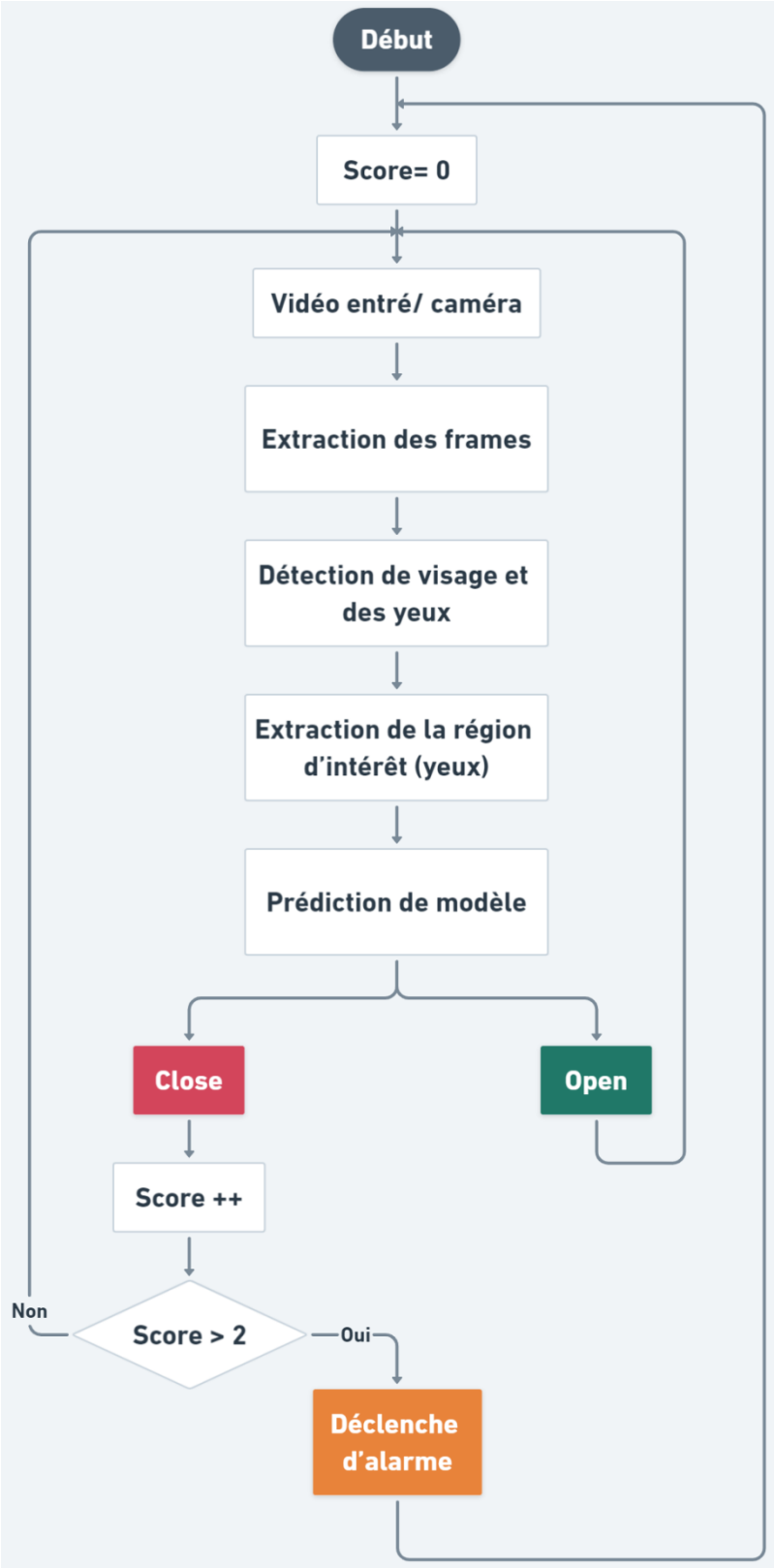

Figure 2.13: Organigramme d'algorithme réalisé.

2.5.8 Résultats expérimentaux :

a- Les résultats d'entrainement :

Au cours de l'entraînement, la fonction de perte et la précision sont calculées et affichées pour chaque lot (batch) de données. À la fin de chaque époque, la fonction de perte et la précision sont calculées pour l'ensemble des données de validation (figure 2.14). L'objectif de l'entraînement est de minimiser la fonction de perte et d'augmenter la précision pour obtenir un modèle performant et précis.

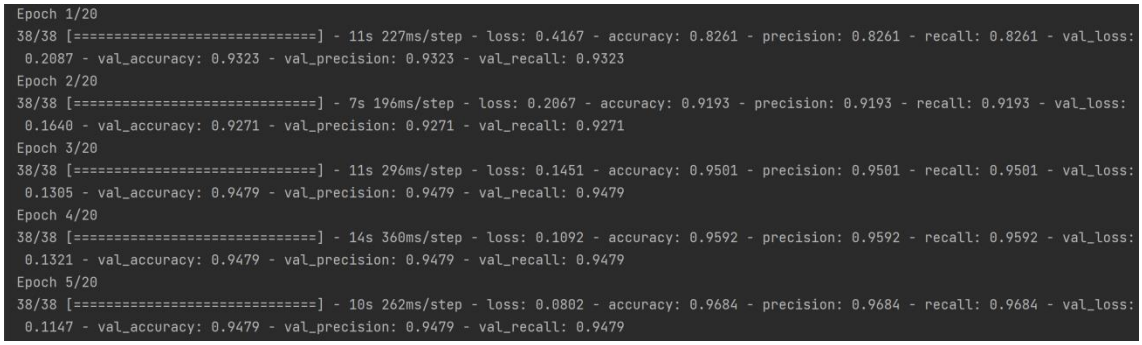

Figure 2.14: Le début de l'entrainement.

Les résultats de l'exécution de notre modèle montrent que nous l'avons entraîné sur un total de 20 époques. À chaque époque, le modèle a été alimenté avec un lot de données d'entraînement et a ajusté ses paramètres internes pour minimiser la fonction de perte, qui mesure la disparité entre les valeurs prédites et réelles.

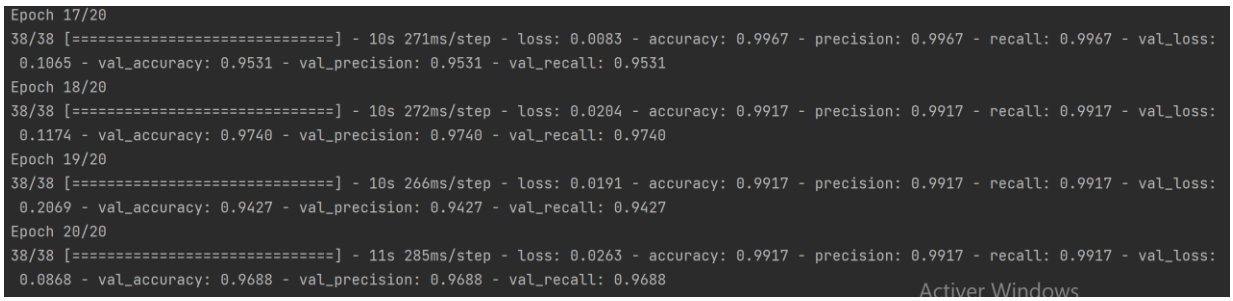

Figure 2.15: Fin de l'entrainement

Les résultats de l'entrainement sont présentés ci-dessous :

Les résultats	D'entrainement	De validation
Loss (perte)	0,0263	0 ,0868
Accuracy (exactitude)	0,9917	0 ,9688
Precision (précision)	0,9917	0,9688
Recall	0,9917	0,9688

Table 2.1 : Résultat d'entrainement de modèle CNN.

Nous avons représenté les résultats d'exécution de notre modèle dans des graphes affichant l'évolution de l'exactitude (accuracy), de la perte (loss) et la précision (precision). (Figure 2.16, 2.17, 2.18).

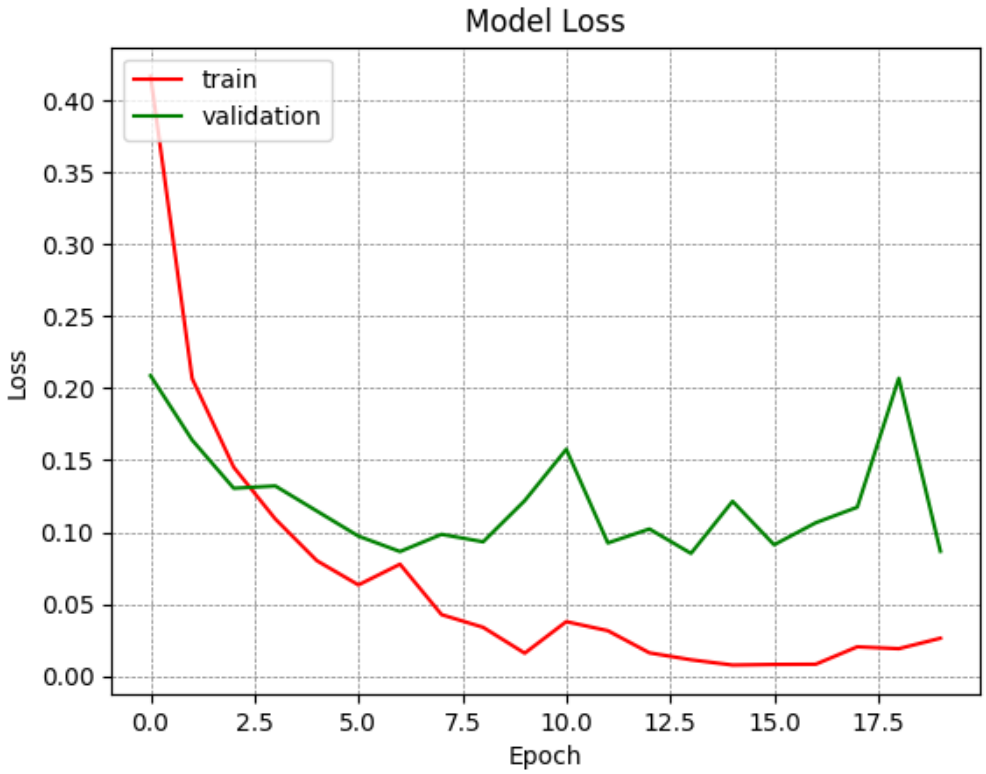

Figure 2.16 : Présentation de perte de l'entrainement.

En ce qui concerne la métrique de perte, le modèle a commencé avec une perte d'entraînement de 0,4167 et une perte de validation de 0,2087 lors de la première époque. Tout au long de l'entraînement, la perte a diminué progressivement, indiquant la capacité du modèle à optimiser ses paramètres et à effectuer de meilleures prédictions. À la fin de l'entraînement, le modèle a atteint une perte d'entraînement de 0,0263 et une perte de validation de 0,0868.

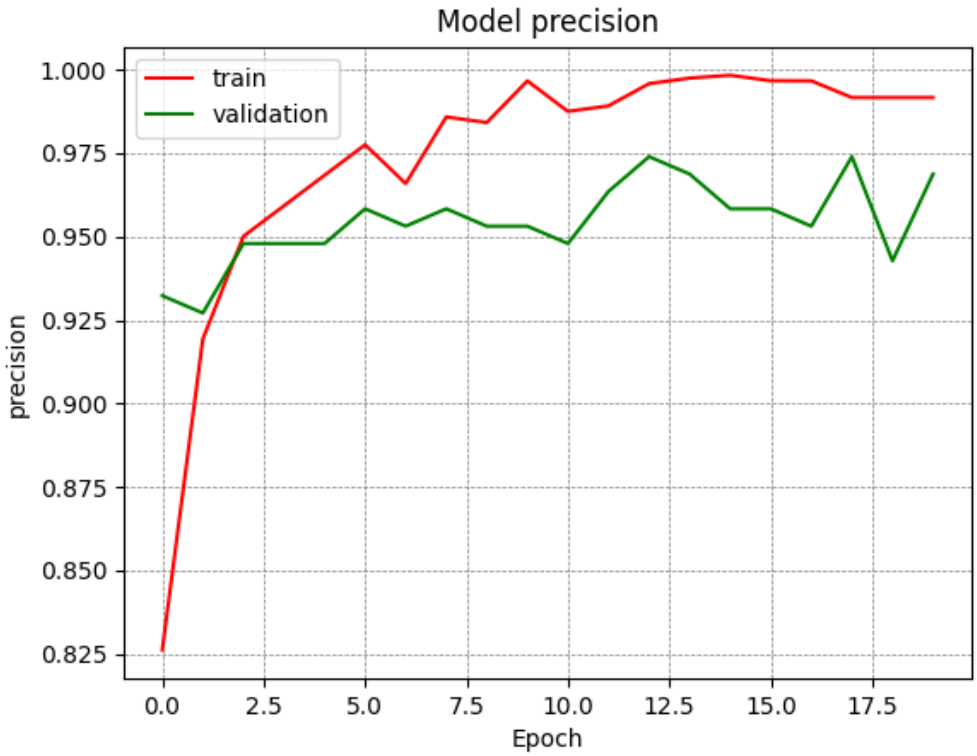

Figure 2.17: Présentation de précision de l'entrainement.

En ce qui concerne les mesures de précision et de rappel, elles suivent une tendance similaire avec une amélioration constante. La précision, qui représente la proportion de prédictions positives correctes parmi toutes les prédictions positives, atteint 96,88% à la dernière époque.

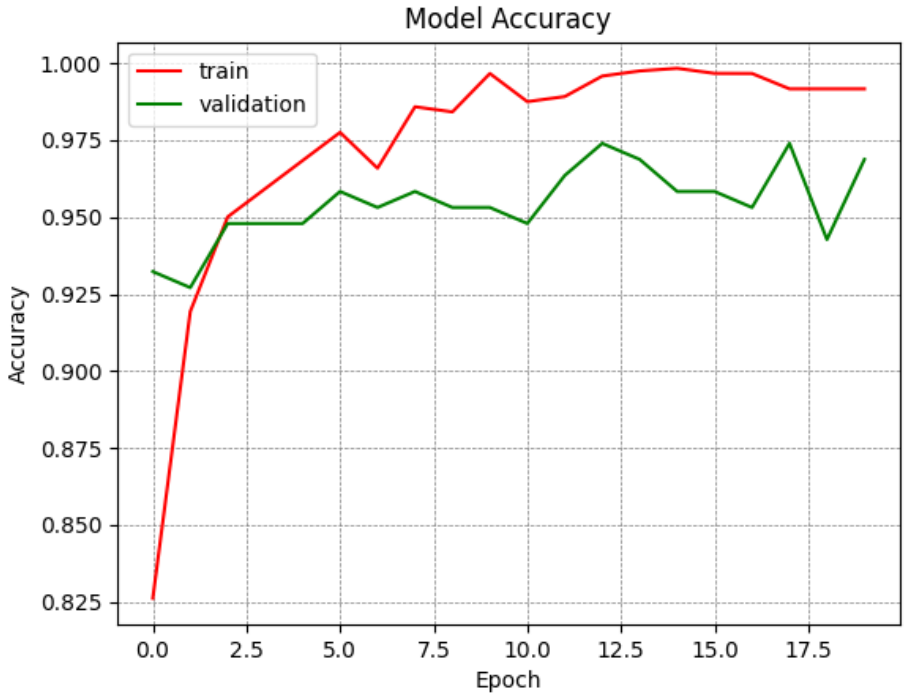

Figure 2.18 : Présentation de l'exactitude de l'entrainement.

L'exactitude, qui mesure la proportion de prédictions correctes, augmente également progressivement, atteignant une valeur de 99,17% à la dernière époque. Et une exactitude légèrement inférieure, mais toujours impressionnante de 0,9688 sur les données de validation. Ce qui signifie que le modèle a correctement prédit l'état de somnolence du conducteur pour 96 ,88% des instances dans l'ensemble de validation.

Ces résultats démontrent que le modèle a efficacement appris à partir des données d'entraînement et s'est bien généralisé aux données de validation. Avec une précision de 96,88 % sur l'ensemble de validation, le modèle a montré sa capacité à effectuer des prédictions précises sur des données inconnues. Les valeurs décroissantes de perte indiquent que le modèle a réussi à converger vers des valeurs de paramètres optimales lors de l'entraînement.

c- Résultats de simulation :

L'exécution de notre programme principale a donné les résultats suivants (figure 2.19, 2.20):

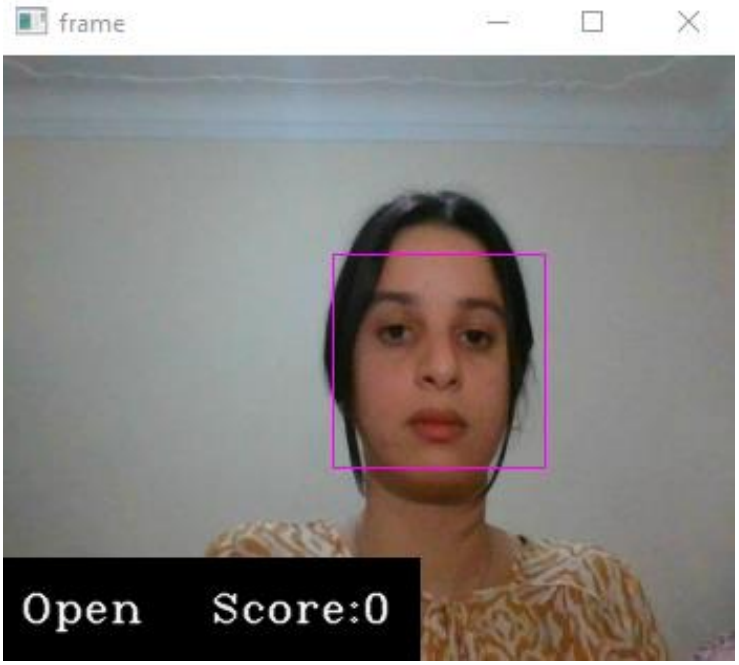

Figure 2.19 : Résultat des yeux ouvert.

Dans cette situation, les yeux ont été détectés dans un état ouvert avec un score de 0. Lorsque les yeux se ferment, Le score incrémente, lorsqu'il atteint 2 ou plus, ce qui correspond à une durée de deux secondes. Dans ce cas, l'alarme est déclenchée.

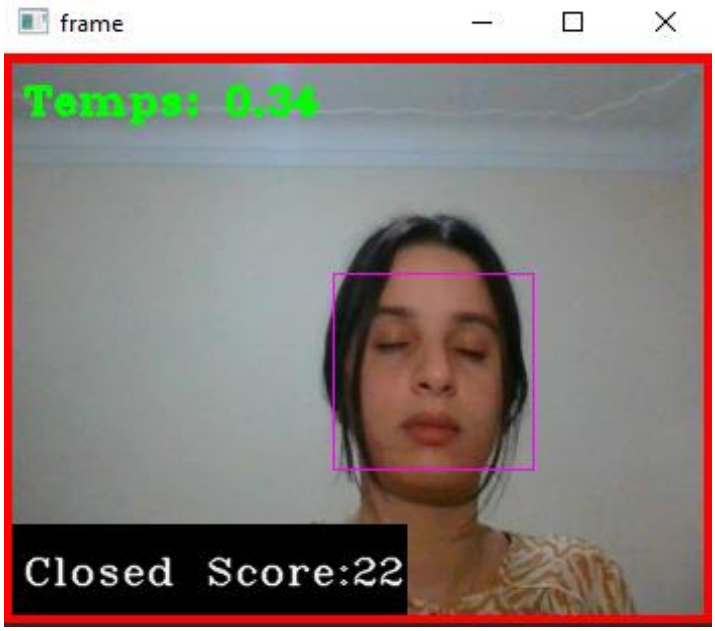

Figure 2.20 : Résultat des yeux fermés.

2.6 Méthode 02 : système de détection de la somnolence en utilisant facial landmarks.

La détection de traits faciaux et la reconnaissance des landmarks (points de repère) sont des tâches essentielles dans de nombreuses applications de vision par ordinateur, notamment la détection de la somnolence. Dans cette étude, nous avons utilisé la bibliothèque Dlib et le modèle shape_predictor_68_face_landmarks.dat pour développer un système de détection de la somnolence précis et robuste.

2.6.1 La détection des points caractéristiques dans un visage :

L'étape de caractérisation consiste à dégager les caractéristiques pertinentes de l'expression faciale étudiée et d'éliminer les informations redondantes. Puisque nous intéressons à la fatigue, nous cherchons à extraire des points caractéristiques spécifiques dans le visage : ce sont ceux des yeux et la bouche. La détection de ces points est implémentée dans la librairie dlib utilisé sous le langage Python. Elle permet de produit 68 point 2D de coordonnées (x, y) qui cartographient des structures faciales spécifiques. Ces points sont stockés dans un tableau indexé. Voici donc les indices de chaque point parmi les 68 points (figure 2.21).

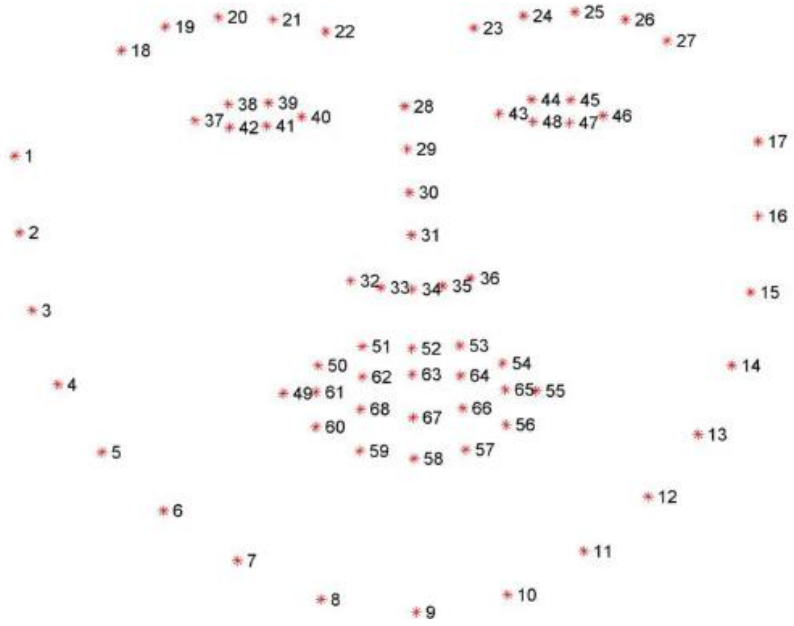

Figure 2.21: Shape predictor land marks 68.

Dans notre implémentation, nous nous concentrons sur les yeux droits et gauches et la bouche. La première tâche de cette étape est de détecter ces caractéristiques en identifiant les points caractéristiques (Landmarks) des yeux et de la bouche dans le visage. La deuxième tâche est de maintenir ces points caractéristiques générés qui contiennent des informations caractéristiques des visages. La dernière tâche est de conserver ces repères pour le calcul de la matrice des poids dans l'espace engendré par les visages propres retenus, (figure 2.22).

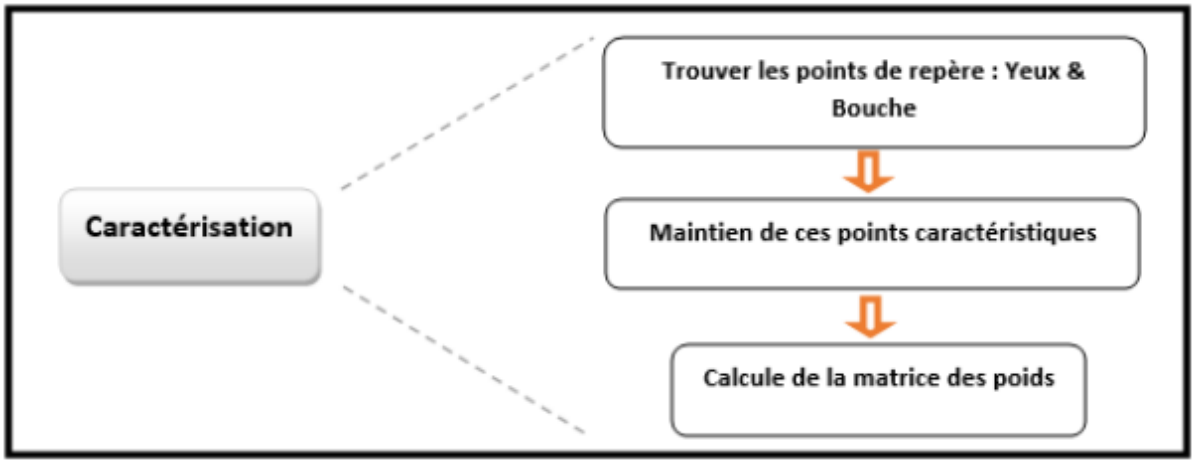

Figure 2.22 : Détail de l'étape de caractérisation.

Pour développer notre code et avoir prédicter les yeux et la bouche correctement, nous avons utilisé la bibliothèque Dlib pour le traitement des images dans notre programme principal.

2.6.2 Dlib :

Dlib est une bibliothèque open source multiplateforme (c'est-à-dire peut être exécutée directement sur n'importe quelle plate-forme sans préparation spéciale) écrite en langage C ++, qui fournit une documentation complète et précise ainsi que de puissants outils de débogage. Dlib vise à fournir un environnement tout aussi riche pour le développement de logiciels d'apprentissage automatique, y compris la classification, la régression, le clustering, Réseaux bayésiens. Et prenant en charge des fonctionnalités, notamment le threading, la mise en réseau, les algorithmes numériques, le traitement d'image et plein d'autres. Les fonctionnalités qui nous intéressent sont celles de traitement d'image, et spécifiquement les outils de détection d'objets dans les images, tel que la détection de face frontale, qui est basé sur l'approche de détection des repères faciaux.

- **Détection des yeux fermés :**

Dans dlib, chaque œil est représenté par 6 points de coordonnées (x, y) comme indiqué dans la (figure 2.23).

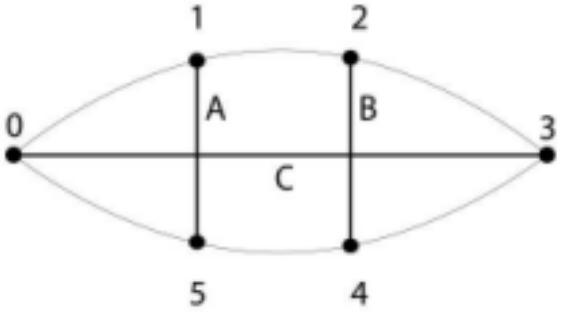

Figure 2.23:Région d'un œil représentée par les points caractéristiques.

Pour décrire et représenter les points caractéristiques décrivant les yeux, nous allons utiliser une mesure appelée Rapport d'Aspect Oculaire, en anglais « Eye Aspect Ratio (EAR) » **[17]**. Cette mesure représente le rapport entre la largeur et la hauteur de l'œil. La valeur de cette mesure va nous permettre de définir et de mesurer l'ouverture de l'œil, plus elle grande, plus l'œil est ouvert. Elle va donc nous permettre plus tard de caractériser la base des exemples (images) pour trouver un modèle d'apprentissage permettant de prédire l'état d'une nouvelle instance. La valeur d'EAR est calculée comme suit :

$$\mathbf{Ae} = d(eye[1], eye[5]) \quad (3.1)$$

$$\mathbf{Be} = d(eye[2], eye[4]) \quad (3.2)$$

$$\mathbf{Ce} = d(eye[0], eye[3]) \quad (3.3)$$

Tel que **Ae** et **Be** mesurent respectivement la distance verticale de l'œil et ce calcule les dimensions horizontales de l'œil, alors :

$$\mathbf{EAR = (Ae+Be)/ (2.Ce)} \quad (3.4)$$

- **Seuil oculaire (Te):**

C’est la valeur qui permet de définir l’état de l’œil. Si la valeur de l’EAR est en dessous de cette valeur, l’œil est considéré comme fermé.

Etat des yeux = Closed, EAR < Te , Open, EAR ≥ Te (3.5)

Dans la littérature, la plupart des travaux essaient de trouver cette valeur empiriquement. La valeur de Te peut être déterminée par essais et erreurs, en recherchant les différentes valeurs de Te, de sorte que le système puisse correctement classer les différentes instances, la valeur la plus représentative est : Te = 0,3. Dans ce travail.Pour cela, nous allons utiliser un algorithme d’apprentissage permettant de calculer les différentes valeurs de l’EAR et pour chaque état parmi les deux états possibles, puis comparer les résultats.

b- Détection de la bouche et du bâillement :

Après avoir détecté l’œil et calculé la valeur de EAR, la prochaine étape est la détection des bâillements qui consiste à localiser la bouche et les lèvres. Pour ce faire, la zone de la bouche marquée par des points caractéristiques est calculée aussi par Dlib, ils sont indiqués dans la (figure 2.24).

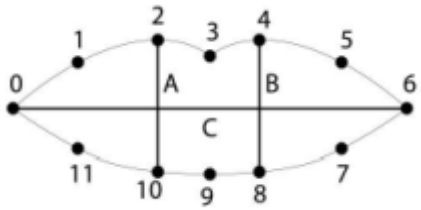

Figure 2.24: Une bouche représentée par les points caractéristiques.

De la même façon, nous calculons une mesure appelée Rapport d’Aspect de la Bouche, en anglais « Mouth Aspect Ration (MAR) », qui définit le rapport entre la hauteur et la largeur de la bouche. Sur la base des valeurs du MAR, nous pouvons savoir si la bouche est ouverte (dans un état de bâillement), ou fermée. Cette valeur est calculée comme suit :

$$Am = d(mouth[2], mouth[10])$$

$$Bm = d(mouth[4], mouth[8])$$

$$Cm = d(mouth[0], mouth[6])$$

Am et Bm mesurent l'ouverture verticale de la bouche et Cm calcule la largeur de la bouche, alors :

$$MAR = (Am+Bm)/ (2.0 *Cm)$$

- **Seuil de la bouche (Tm) :**

C'est la valeur seuil de MAR qui permet de définir si la bouche est ouverte en situation de bâillement ou autres. Si la valeur de MAR dépasse cette valeur, la bouche est considérée comme étant dans un état de bâillement.

$$\text{Etat de bouche} = \text{Yawn, MAR} > T_m$$

$$\text{Etat de bouche= Closed/Talking, MAR} \leq T_m$$

La littérature utilise des essais et erreurs pour trouver la valeur Tm afin que le système puisse correctement classifier une instance de bâillement et de bouche fermée. Un problème de détection de faux positifs a été rencontré lors de la détermination du seuil optimal pour la bouche béante, car le conducteur pouvait avoir la bouche ouverte tout en parlant. Il ne faut pas le considérer comme un état de bâillement. Le seuil est choisi de manière à être suffisamment élevé pour ignorer la bouche ouverte lorsque vous parlez dans des conditions normales, parce que la bouche humaine s'ouvre beaucoup plus largement sur un certain nombre d'images consécutives en bâillant qu'en parlant.

2.6.3 L'organigramme du programme principal :

Le programme est utilisé pour détecter la somnolence et le bâillement sur le visage d'une personne. Il capture la vidéo à partir d'une webcam et utilise la bibliothèque Open CV pour détecter les visages et les repères faciaux. Le script calcule ensuite le rapport d'aspect des yeux (EAR) pour chaque œil, et si l'EAR est inférieur à un certain seuil, la personne est considérée comme somnolente. De même, le script détermine la distance entre les lèvres supérieure et inférieure, et si la distance est supérieure à un seuil, la personne est considérée comme bâillant comme présenté dans la, figure 2.25.

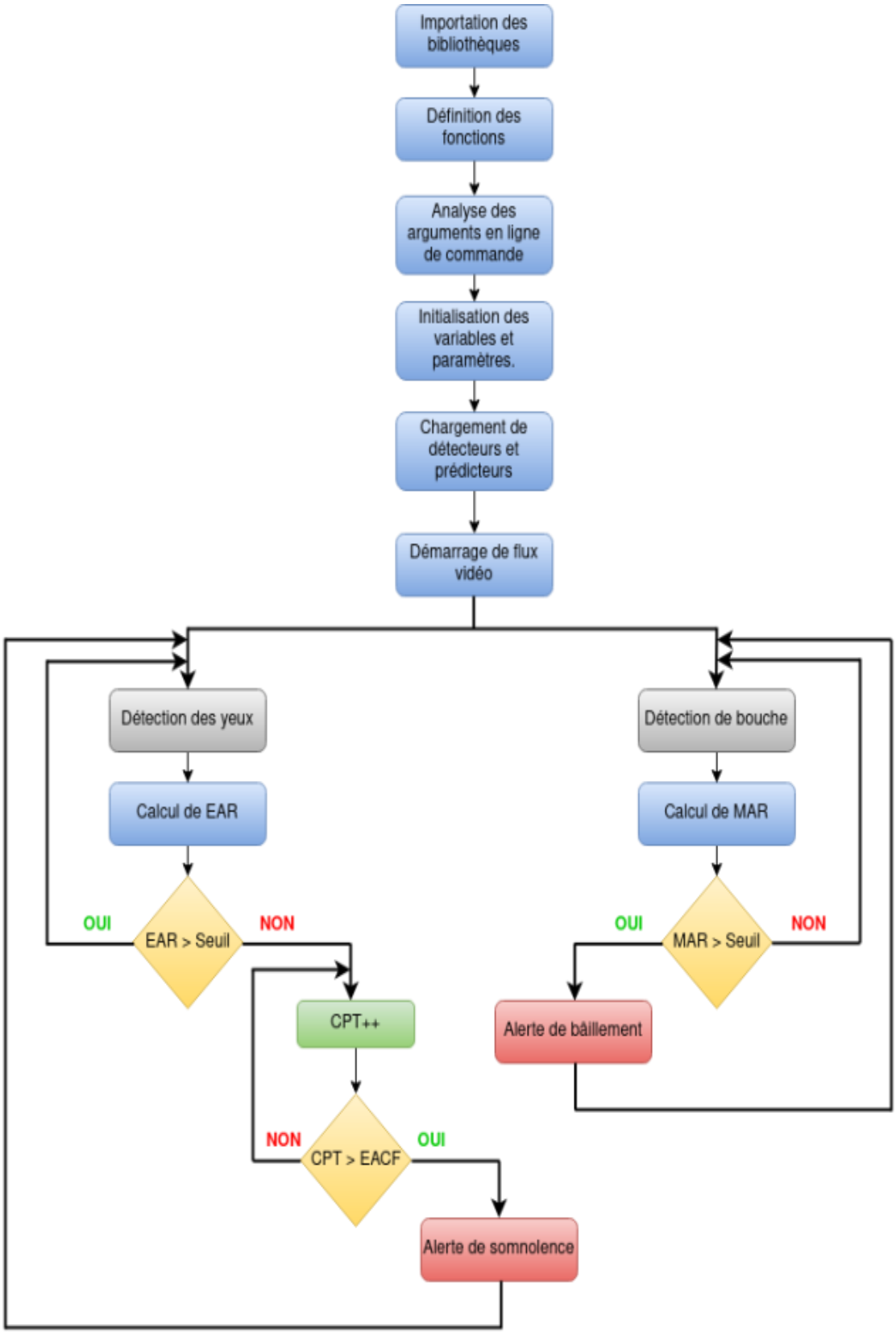

Figure 2.25 : Organigramme du programme principal de la méthode 02.

2.7 Conclusion

Dans ce chapitre, nous avons abordé différents aspects de la détection de la somnolence, tels que la collecte et le prétraitement des données, la conception et l'entraînement du modèle CNN, ainsi que l'évaluation des performances de ce dernier.

Nous avons commencé par d'écrire la problématique, les défis de notre système de détection de somnolence ainsi que les objectifs et les contraintes de ce système.

Après cela, nous avons présenté les étapes pour collecter des données pertinentes, pour les rendre adaptées à l'entraînement de notre modèle CNN, et concevoir une architecture de réseau adaptée à la détection de la somnolence.

Il est noté que la deuxième méthode n'a pas pu être testé, ceci étant de la non-adaptation entre les bibliothèques utilisées (Dlib, etc.) et l'environnement de développement (visual studio code et Pycharm), mais cette dernière fera l'objet du prochain chapitre lors de l'implémentation sur Raspberry pi.

Enfin, nous avons évalué les performances du modèle de détection de la somnolence et discuté les résultats obtenus, et nous sommes arrivées à la conclusion suivante : le modèle CNN développé s'est avéré très satisfaisant lors la simulation vu qu'il a été entrainé de manière efficace. Nous avons conclu dans ce chapitre que les résultats de notre simulation est impressionnante et que notre modèle a été entrainé de manière efficace.

Chapitre 3 : Implémentation de deux systèmes de détection de la somnolence sur Raspberry pi

3.1Introduction

Dans ce chapitre, nous aborderons l'implémentation de notre système de détection de somnolence dans Raspberry Pi 3, une plate-forme informatique polyvalente et accessible. L'utilisation de Raspberry Pi 3 offre de nombreux avantages, tels que sa taille compacte, sa faible consommation d'énergie et sa capacité à exécuter des applications en temps réel.

L'objectif principal de ce chapitre est de présenter comment intégrer un système de détection de somnolence basé sur le deep learning dans Raspberry Pi 3. Nous explorerons les différentes étapes clés, de l'acquisition des données à l'entraînement du modèle, en passant par l'intégration du modèle dans la Raspberry Pi 3 et les tests de performance.

Nous commencerons par décrire la méthodologie et le matériel, tels que les logiciels et plateforme utilisés.

Nous poursuivrons avec la modélisation et l'entraînement du modèle de détection de somnolence basé sur le deep learning. Nous discuterons l'optimisation du modèle pour obtenir des performances optimales.

Ensuite, nous détaillerons l'intégration du modèle entraîné dans Raspberry Pi 3. Nous expliquerons la configuration de l'environnement de développement, l'adaptation du modèle pour une utilisation en temps réel et l'optimisation des performances et des ressources pour la Raspberry Pi 3.

Enfin, nous présenterons les tests et l'évaluation du système de détection de somnolence. Nous discuterons des résultats obtenus et des perspectives d'amélioration pour de futures recherches dans ce domaine.

3.2 Matériels et méthodes :

Dans cette section, nous détaillerons la méthodologie et le matériel utilisés pour l'implémentation du système de détection de somnolence au volant dans Raspberry Pi 3.

3.2.1 Description hardware :

On a utilisé dans l'implémentation de notre modèle une carte Raspberry pi 3 modèle B, une caméra web et haut-parleur pour le déclenchement d'alarme.

a- Raspberry pi 3 modèle B :

Raspberry Pi 3 est une carte informatique mono carte extrêmement populaire et largement utilisée dans le domaine de l'informatique embarquée et du prototypage de projets électroniques. Elle est conçue pour être abordable, polyvalente et facile à utiliser, ce qui en fait un choix idéal pour les étudiants et les professionnels.

La Raspberry Pi 3 modèle B est une version améliorée et plus puissante de la gamme de cartes Raspberry Pi. Il offre une multitude de fonctionnalités intégrées qui en font une plateforme polyvalente pour une large gamme d'applications. Voici une description des caractéristiques principales de la Raspberry Pi 3 modèle B, (figure3.1).

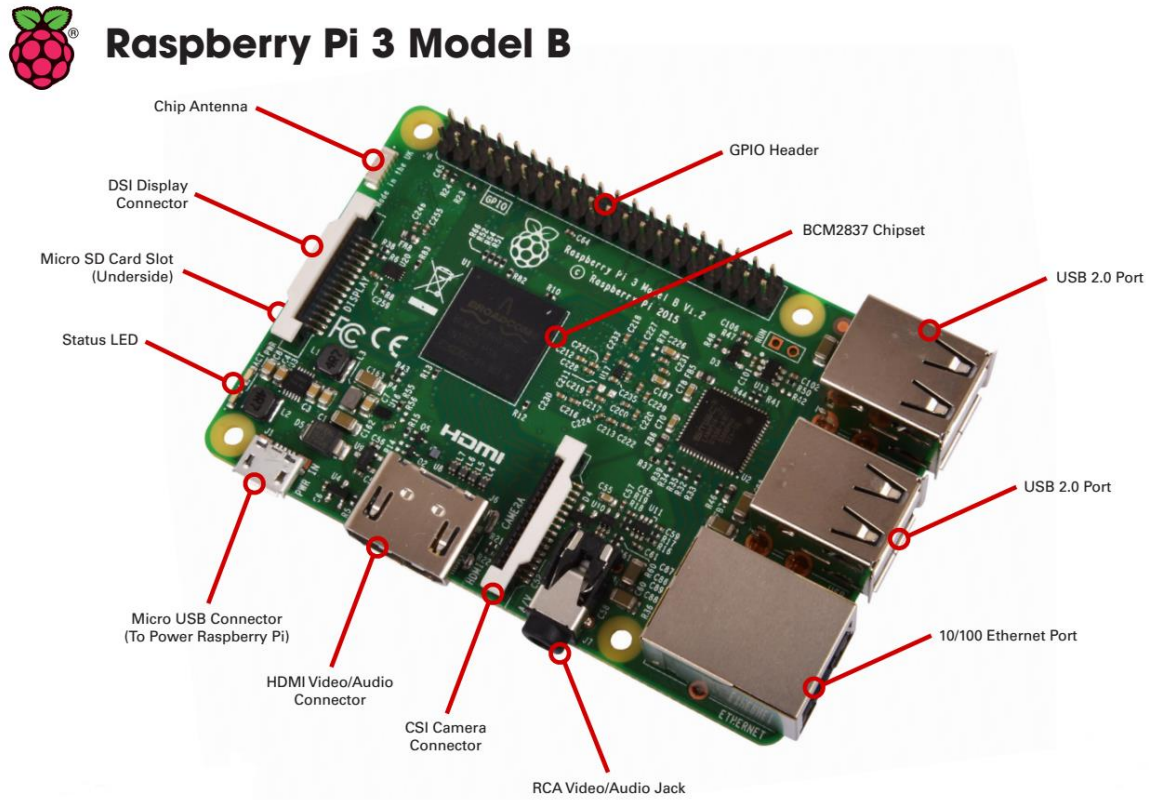

Figure 3.1: Carte de Raspberry pi 3.

- **Processeur :** Le modèle B est équipé d'un processeur Broadcom BCM2837 quadri cœur ARM Cortex-A53 cadencé à 1,2 GHz. Cette augmentation de puissance par rapport aux versions précédentes permet d'exécuter des applications plus exigeantes et de réaliser des tâches de calcul plus complexes.
- **Mémoire :** Il dispose de 1 Go de mémoire LPDDR2, ce qui permet une exécution fluide des applications et une multitâche efficace.
- **Connectivité sans fil :** Le modèle B intègre une connectivité sans fil grâce à la prise en charge du Wi-Fi 802.11n et du Bluetooth 4.2/BLE. Cela permet une connexion facile à des réseaux sans fil et la communication avec des périphériques compatibles Bluetooth.
- **Ports USB :** Il est équipé de 4 ports USB 2.0, ce qui permet de connecter une variété de périphériques tels que des claviers, des souris, des disques durs externes et des dongles Wi-Fi.
- **Port Ethernet :** Le modèle B est doté d'un port Ethernet 10/100 Mbps pour une connectivité réseau filaire.
- **HDMI et sortie audio :** Il est équipé d'un port HDMI pour la sortie vidéo en haute définition, ainsi que d'une prise audio 3,5 mm pour la sortie audio stéréo.
- **Carte microSD :** La Raspberry Pi 3 modèle B utilise une carte microSD comme support de stockage principal, sur laquelle le système d'exploitation et les applications sont installés.
- **GPIO :** Il dispose de 40 broches GPIO (General Purpose Input/Output) qui permettent de connecter des composants électroniques externes tels que des capteurs, des actionneurs et des modules d'extension.
- **Alimentation :** Il est alimenté par un adaptateur micro USB standard, avec une tension d'entrée de 5 V.
- **Ports d'affichage :** Outre le port HDMI, le modèle B comprend également une sortie vidéo composite et une interface pour caméra CSI (Caméra Serial Interface) pour la connexion de caméras compatibles.

b- Haut –parleur/ Buzzer :

Un haut-parleur est un dispositif électroacoustique utilisé pour convertir un signal électrique en un son audible. Il est conçu pour produire des vibrations mécaniques qui génèrent des ondes sonores dans l'air, permettant ainsi la reproduction sonore. Il est

connecté à la Raspberry Pi et représente la sortie audio pour avertir le conducteur via la synthèse vocale ou le son du buzzer, (figure 3.2).

Figure 3.2: Hautparleur.

c- Caméra web :

Une webcam est un périphérique électronique utilisé pour capturer des images ou des vidéos en direct et les transmettre via un réseau informatique ou Internet. Elle est généralement connectée à un ordinateur ou à un appareil mobile et est utilisée pour des activités telles que les appels vidéo, les conférences en ligne, les diffusions en direct, les enregistrements vidéo, etc.

Les images capturées par la webcam sont ensuite prétraitées, souvent en les redimensionnant, en ajustant la luminosité, le contraste ou en appliquant d'autres transformations nécessaires. Ensuite, elles sont utilisées comme entrées pour les modèles de deep learning, qui peuvent être des réseaux de neurones convolutifs (CNN), des réseaux de neurones récurrents (RNN), des réseaux de neurones à convolution 3D (CNN 3D), ou d'autres architectures adaptées à la tâche spécifique, (figure3.3).

Figure3.3: Webcam HAMA C-400.

3.2.2 Description software:

a- Préparation de l'environnement Raspberry pi :

Pour notre implémentation, nous avons utilisé Raspbian GNU Linux comme un système d'exploitation pour le Raspberry pi .**GNU/Linux** est un système d'exploitation : ne série de programmes qui vous permettent d'interagir avec votre ordinateur et d'exécuter d'autres programmes, (figure 3.4).

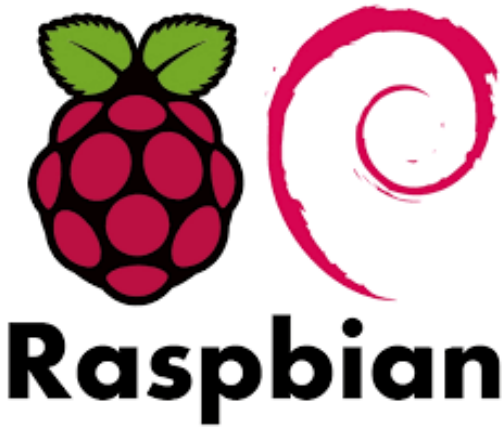

Figure 3.4: Logo de Raspberry pi avec OS Raspbian.

On a utilisé la commande **cat /etc/os-release** pour afficher les informations de système d'exploitation utilisé comme illustré dans la figure 3.5.

```
ayemen@pi:~ $ cat /etc/os-release
PRETTY_NAME="Raspbian GNU/Linux 11 (bullseye)"
NAME="Raspbian GNU/Linux"
VERSION_ID="11"
VERSION="11 (bullseye)"
VERSION_CODENAME=bullseye
ID=raspbian
ID_LIKE=debian
HOME_URL="http://www.raspbian.org/"
SUPPORT_URL="http://www.raspbian.org/RaspbianForums"
BUG_REPORT_URL="http://www.raspbian.org/RaspbianBugs"
ayemen@pi:~ $
```

Figure 3.5: Système d'exploitation utilisé sur notre Raspberry pi.

.

b- Connexion au Raspberry pi :

nous avons établi une connexion au Raspberry pi via protocole SSH (Secure shell) qui est un protocole de réseau qui permet d'établir une connexion sécurisée et chiffré entre deux ordinateurs, nous avons utilisé PUTTY qui est une interface conviviale permet de se connecter à distance à des systèmes UNIX, Linux et Raspberry Pi depuis un ordinateur exécutant Windows.

Pour cela, nous avons obtenu l'adresse IP de notre Raspberry Pi dans notre réseau en utilisant la commande "ifconfig" qu'elle est : **192.168.43.219**, (figure 3.6).

```
ayemen@pi:~ $ ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether b8:27:eb:30:68:55  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 24  bytes 2466 (2.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 24  bytes 2466 (2.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.43.219  netmask 255.255.255.0  broadcast 192.168.43.255
        inet6 fe80::71e0:3ca9:eda6:4948  prefixlen 64  scopeid 0x20<link>
        ether b8:27:eb:65:3d:00  txqueuelen 1000  (Ethernet)
        RX packets 238  bytes 43811 (42.7 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 180  bytes 27570 (26.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ayemen@pi:~ $
```

Figure 3.6: Affichage de l'adresse IP avec la commande ifconfig

3.3 Implémentation sur Raspberry pi des deux systèmes de détection de somnolence :

3.3.1 Méthode 1:implémentation du modèle CNN :

Pour implémenter le modèle CNN sur Raspberry pi, il nous a fallu faire dans un premier temps l'entrainement de notre modèle sur Raspberry pi, puis le télécharger dans le programme principal pour effectuer la détection.

a- L’entraînement de modèle CNN sur Raspberry pi :

Nous avons créé un environnement virtuel python comme un IDE pour notre projet sur Raspberry pi (figure 3.7) et installer les bibliothèques nécessaires après les avoir téléchargées (figure 3.8).

```
ayemen@pi:~ $ source drowsiness-env/bin/activate
(drowsiness-env) ayemen@pi:~ $
```

Figure 3.7: Activation de l’environnement virtuel.

```
(drowsiness-env) ayemen@pi:~/drowsiness-env/pythonproject $ python3
Python 3.7.12 (default, May 19 2023, 12:38:16)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow
>>> import keras
>>> import cv2
>>> import numpy
>>> import matplotlib
>>> quit()
(drowsiness-env) ayemen@pi:~/drowsiness-env/pythonproject $ scrot
```

Figure 3.8:Les bibliothèques installées sur Raspberry pi.

Pour La base de données utilisée (dataset_new) nous l’avons téléchargée sur Raspberry Pi et placée dans le même répertoire que l'environnement virtuel (Figure3.9).

```
(drowsiness-env) ayemen@pi:~/drowsiness-env/pythonproject $ ls
 2023-06-10-145211_1360x768_scrot.png  'haar cascade files'   models
 alarm.wav                              main1.py              model.tflite
 d-10                                   main2.py              new.py
 d-10_000                               main5.py              thread.py
 d-10_001                               main.py
 dataset_new                            model.py
(drowsiness-env) ayemen@pi:~/drowsiness-env/pythonproject $
```

Figure 3.9:L’emplacement de base de données dans Raspberry pi..

- Après avoir installé toutes les bibliothèques nécessaires et télécharger la base de données sur Raspberry pi, nous avons entrainé le modèle avec la même architecture utilisée dans la simulation sur pc (Figure 3.10).

```
29/38 [=====================>........] - ETA: 7s - loss: 0.0180 - accuracy:
30/38 [======================>.......] - ETA: 6s - loss: 0.0188 - accuracy:
31/38 [======================>.......] - ETA: 6s - loss: 0.0188 - accuracy:
32/38 [=======================>......] - ETA: 5s - loss: 0.0190 - accuracy:
33/38 [=======================>......] - ETA: 4s - loss: 0.0193 - accuracy:
34/38 [========================>.....] - ETA: 3s - loss: 0.0189 - accuracy:
35/38 [========================>.....] - ETA: 2s - loss: 0.0185 - accuracy:
36/38 [=========================>....] - ETA: 1s - loss: 0.0182 - accuracy:
37/38 [=========================>....] - ETA: 0s - loss: 0.0180 - accuracy:
38/38 [==========================] - ETA: 0s - loss: 0.0198 - accuracy:
38/38 [==========================] - 39s 1s/step - loss: 0.0198 - accura
cy: 0.9942 - val_loss: 0.0552 - val_accuracy: 0.9896
Epoch 18/20
 1/38 [..............................] - ETA: 3:59 - loss: 0.0163 - accuracy
 2/38 [>.............................] - ETA: 32s - loss: 0.0476 - accuracy:
 3/38 [=>............................] - ETA: 26s - loss: 0.0353 - accuracy:
 4/38 [==>...........................] - ETA: 24s - loss: 0.0322 - accuracy:
 5/38 [==>...........................] - ETA: 24s - loss: 0.0289 - accuracy:
 6/38 [===>..........................] - ETA: 24s - loss: 0.0257 - accuracy:
 7/38 [====>.........................] - ETA: 26s - loss: 0.0244 - accuracy:
 8/38 [=====>........................] - ETA: 24s - loss: 0.0241 - accuracy:
 9/38 [======>.......................] - ETA: 22s - loss: 0.0225 - accuracy:
 0.9965
```

Figure 3.10: Au cours de l'entrainement de modèle CNN.

- Après 20 époques d'entrainement de notre model, on trouve les résultats suivants (Figure 3.11)

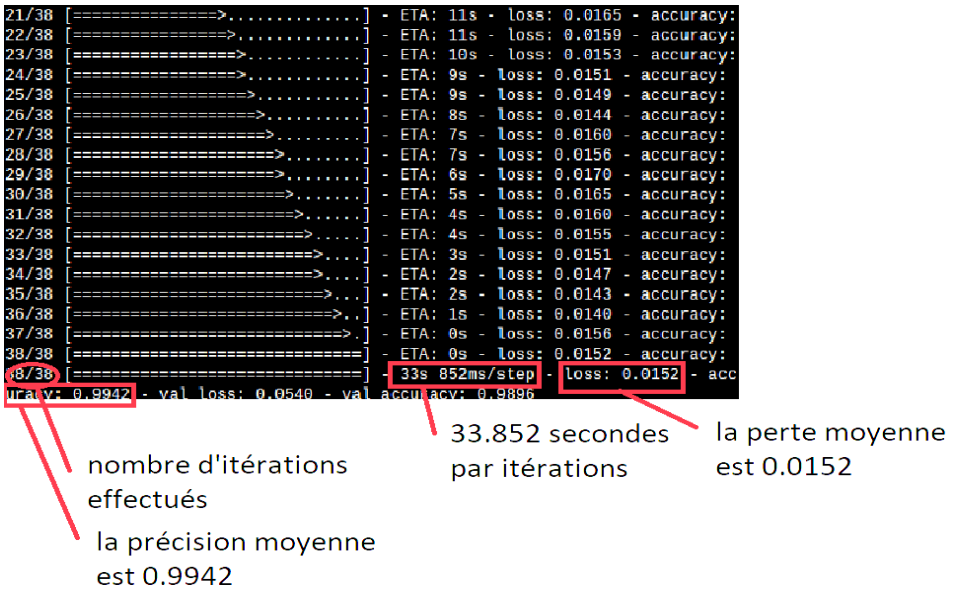

Figure 3.11: Fin de l'entrainement de modèle CNN.

Pour une meilleure illustration on a présenté les résultats dans les graphes de la figure 3.12:

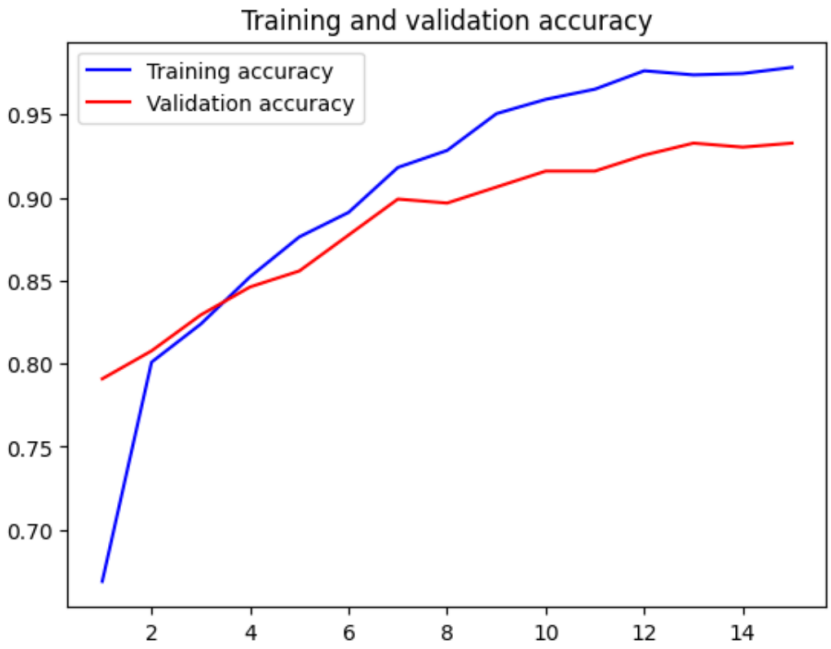

Figure 3.12: Présentation de l'entrainement de model par la précision.

L'exactitude moyenne obtenue sur les données d'entraînement est de 0.9942, soit environ 99,42% qu'indique une meilleure capacité du modèle à classifier correctement les données.

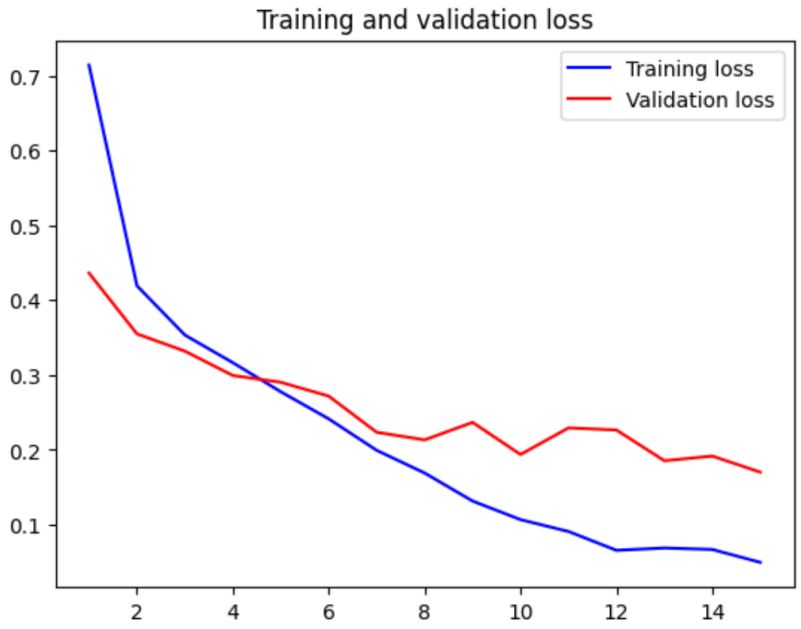

Figure 3.13: Présentation de l'entrainement de model par la perte.

La perte moyenne obtenue sur les données d'entraînement est de 0.0152, qu'indique une meilleure adéquation du modèle aux données.

En résumé, ces résultats indiquent que le modèle atteint une bonne précision et une faible perte lors de l'entraînement, ce qui suggère qu'il ait réussi à bien s'ajuster aux données d'entraînement. De plus, les résultats de validation montrent que le modèle conserve une performance élevée lorsqu'il est évalué sur des données qu'il n'a pas vues auparavant. Ces derniers montrent que le modèle a un fort potentiel pour la détection de la somnolence.

- **Comparaison des résultats d'entrainement entre la simulation sur PC et l'implémentation sur Raspberry pi :**

Résultat d'entrainement	Simulation	Implémentation sur Raspberry Pi
Perte (loss)	0.0169	0.0152
Précision (accuracy)	0.9950	0.9942
Perte sur les données de validation (val_loss)	0.0677	0.0540
Précision sur les données de validation (val_accuracy)	0.9740	0.9896

Table 3.2 : Comparaison des résultats entre la simulation sur PC et l'implémentation sur Raspberry pi.

Les différences entre les deux ensembles de résultats sont relativement petites et les deux approches semblent donner des résultats solides et impressionnants.

b- Résultats d'exécution du programme principal :

On a exécuté notre programme principal sur le terminal de Raspberry pi, (figure 3.14).

```
ayemen@pi:~ $ source drowsiness-env/bin/activate
(drowsiness-env) ayemen@pi:~ $ cd  drowsiness-env/
(drowsiness-env) ayemen@pi:~/drowsiness-env $ cd pythonproject
(drowsiness-env) ayemen@pi:~/drowsiness-env/pythonproject $ python3 main.py
```

Figure 3.14: Commande d'exécution de programme.

Une fois que le programme principal a été exécuté, une fenêtre s'est ouverte, affichant le cadre (frame) qui présente la situation de la personne, qu'elle soit somnolente ou non. Les résultats de cet affichage sont illustrés dans la (figure 3.15).

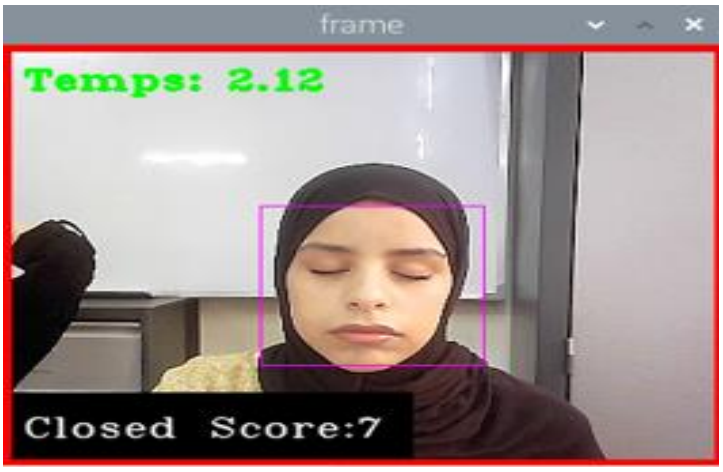

Figure 3.15: Affichage de personne somnolent.

Le modèle a détecté efficacement la somnolence et a déclenché l'alarme lorsque le score atteint une valeur supérieure à 2 (correspondant à 2 secondes). Cependant, le temps de réponse observé se situe entre 1,17 et 2,18 secondes, ce qui indique que la détection n'est pas totalement en temps réel.

- **Comparaison des résultats de la détection entre la simulation sur PC et l'implémentation sur Raspberry pi:**

Dans la simulation, nous avons obtenu un temps de réponse compris entre 0,26 et 0,43 secondes, ce qui prouve que la prédiction a été réalisée avec succès en temps réel (figure 3.16).

```
time: 0.2615149
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 34ms/step
[[1.6823227e-03 9.9831772e-01 2.8379213e-12 1.0739262e-09]] [[1.6956864e-04 9.9983037e-01 1.6384964e-10 1.2186808e-08]]
time: 0.2622416
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 31ms/step
[[9.9999952e-01 4.1918790e-07 5.8847126e-14 1.6755259e-13]] [[9.9928027e-01 7.1974559e-04 7.8865248e-12 3.7520545e-10]]
time: 0.2986291
```

Figure 3.16: Temps de réponse lors de simulation sur PC.

L'implémentation sur Raspberry Pi, a donné un temps de réponse qui se situe entre 1,15 et 2,20 secondes. Ce dernier est nettement plus élevé que celui de la simulation et n'est en aucun cas en temps réel (figure 3.17). Ceci étant dû au fait que le PC utilisé pour la simulation est équipé d'un processeur Core i5, qui est plus performant et plus rapide, ce qui permet d'effectuer efficacement des opérations et des calculs, En comparaison avec Raspberry Pi, qui possède une architecture armv7 (présente des performances limitées).

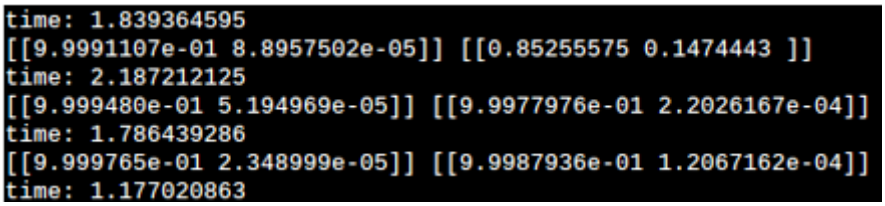

Figure 3.17: Temps de réponse lors de l'implémentation.

3.3.2 Méthode 2 : implémentation du système de détection de somnolence en utilisant facial landmarks :

Nous avons implémenté le deuxième système de détection de somnolence sur Raspberry pi, on important toutes les bibliothèques nécessaires ainsi que le modèle « facial landmarks ».

a- Importation des bibliothèques nécessaires:

Pour cette méthode nous avons dû importer les bibliothèques mentionnées dans la Figure 3.18.

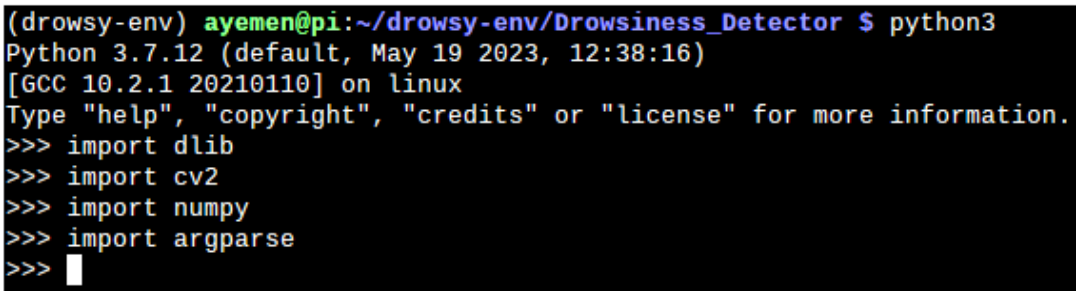

Figure 3.18: Les bibliothèques nécessaires pour la deuxième.

b- Résultats d'exécution de la deuxième méthode :

Les résultats de L'exécution du programme principal développé avec dlib sur Raspberry pi, sont montrés dans la figure 3.19, on constate qu'il y a détection puisqu'il il y a affichage des alertes.

Figure 3.19: Affichage d'une détection de somnolence et de bâillement.

- Dans ce système, nous avons défini une valeur de EAR fixée à 0,3. Ainsi, si EAR est inférieur à 0,3, une alerte de somnolence est déclenchée après le traitement de 3 images.
- En ce qui concerne le bâillement, nous observons que lorsque MAR dépasse 20, cela indique une détection de bâillement (yawing) et entraîne également une alerte.

Pour cette méthode, le temps de réponse obtenu se situe dans l'intervalle [0,45, 0,48], (figure 3.20).

```
Average response time: 0.48 seconds
Average response time: 0.47 seconds
Average response time: 0.47 seconds
Average response time: 0.47 seconds
Average response time: 0.47 seconds
Average response time: 0.47 seconds
Average response time: 0.47 seconds
Average response time: 0.47 seconds
Average response time: 0.46 seconds
Average response time: 0.46 seconds
Average response time: 0.46 seconds
Average response time: 0.46 seconds
Average response time: 0.46 seconds
Average response time: 0.46 seconds
Average response time: 0.46 seconds
Average response time: 0.46 seconds
Average response time: 0.46 seconds
Average response time: 0.45 seconds
```

Figure 3.20: Temps de réponse moyen pour chaque trame.

3.3.3 Comparaison des résultats de détection entre les deux méthodes :

Lors de la comparaison entre la première méthode, développée avec la bibliothèque TensorFlow, et la deuxième, basée sur la bibliothèque dlib, nous avons obtenu les temps de réponse suivants :

Les méthodes	première méthode	deuxième méthode
Temps de réponse	[1,17. 2,18] secondes	[0,45. 0,49] secondes

Table 3.3 : Comparaison des résultats de détection entre les deux méthodes.

Ces résultats indiquent que la méthode dlib est plus rapide dans ce contexte spécifique.

- On note que l'utilisation de TensorFlow sur une Raspberry Pi peut poser des défis en termes de ressources, car TensorFlow est une bibliothèque relativement lourde et nécessite une puissance de calcul significative. Cela peut limiter les capacités du Raspberry Pi en termes de vitesse d'exécution et de consommation d'énergie.
- Alors que dlib est une bibliothèque de vision par ordinateur populaire, en raison de sa simplicité, dlib peut être plus facile à mettre en œuvre et à exécuter sur une Raspberry Pi par rapport à TensorFlow.

La comparaison du temps de réponse entre la méthode dlib et celle de la simulation sur Pc est donnée ci-dessous :

Les méthodes	Méthode avec Dlib	Simulation sur PC avec TensorFlow
Temps de réponse	[0.45, 0.48] seconde	[0.26, 0.43] seconde

Table 3.4 : La comparaison du temps de réponse entre la méthode dlib et celle de la simulation sur PC.

Nous avons effectué la simulation sur un pc équipé d'un processeur Intel Core i5 de 4^{e} génération 4200U / 1.6 GHz AVEC 4Go DE RAM qui est plus puissant que le processeur Broadcom BCM2837 quadricœur ARM Cortex-A53 cadencé à 1,2 GHz avec une ram de 1 Go du Raspberry Pi 3 B. Le Core i5 est conçu pour des tâches plus

intensives en matière de calcul et de multitâche, tandis que le processeur du Raspberry Pi 3 B est un processeur ARM plus basse consommation destiné à des tâches moins exigeantes. Ceci explique clairement les résultats obtenus ci-dessous c.à.d. La simulation sur PC offre des temps de réponse légèrement plus rapides que l'exécution sur Raspberry Pi, mais ces derniers (i.e. temps de réponse) restent comparables.

3.4 Conversion de système à fichier exécutable :

Pour embarqué notre système sur Raspberry pi, nous avons généré un fichier exécutable .exe en regroupant le programme principal et les fichiers requis en un seul fichier. Pour ce faire, nous avons suivi plusieurs étapes mentionnées dans l'organigramme ci-dessus pour obtenir cet exécutable, (figure 3.21).

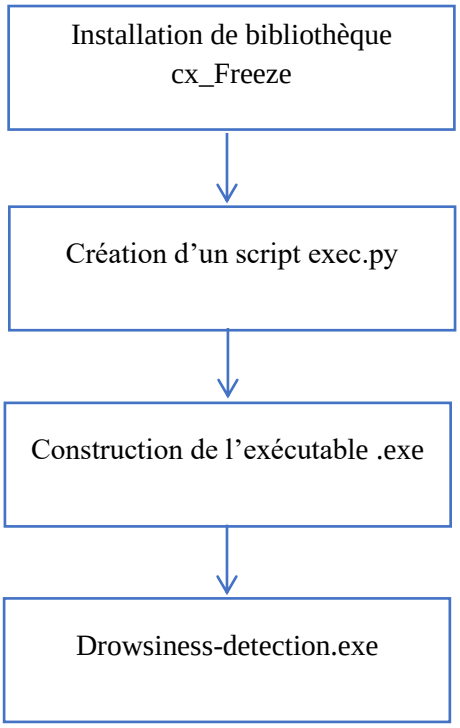

Figure 3.21 : Organigramme de conversion de système à exe.

3.4.1 Installation de LA bibliothèque cx_Freeze:

Dans le cadre de notre projet qui utilise LA Raspberry Pi, nous avons intégré la bibliothèque cx_Freeze, un outil open-source largement utilisé, afin de convertir le script Python en exécutable autonome. Cette installation a été réalisée dans l'environnement virtuel spécifique à notre projet sur Raspberry Pi (figure 3.22).

```
ayemen@pi:~ $ source drowsy-env/bin/activate
(drowsy-env) ayemen@pi:~ $ cd drowsy-env/
(drowsy-env) ayemen@pi:~/drowsy-env $ cd drowsiness-detection
(drowsy-env) ayemen@pi:~/drowsy-env/drowsiness-detection $ cd detection
(drowsy-env) ayemen@pi:~/drowsy-env/drowsiness-detection/detection $ pip install
 cx_Freeze
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Requirement already satisfied: cx_Freeze in /home/ayemen/drowsy-env/lib/python3.
7/site-packages (6.15.1)
Requirement already satisfied: setuptools<68,>=62.6 in /home/ayemen/drowsy-env/l
ib/python3.7/site-packages (from cx_Freeze) (67.8.0)
Requirement already satisfied: patchelf>=0.17.2.1 in /home/ayemen/drowsy-env/lib
/python3.7/site-packages (from cx_Freeze) (0.17.2.1)
(drowsy-env) ayemen@pi:~/drowsy-env/drowsiness-detection/detection $
```

Figure 3.22 : Installation de cx_Freeze sur Raspberry pi.

3.4.2 Créations d'un script exec.py:

Nous avons rédigé un script de configuration qui détaille les paramètres nécessaires à la création de notre projet Python. Ce script de configuration répertorie les dépendances, intègre tous les fichiers ou ressources supplémentaires requis et définit les propriétés de l'exécutable.

3.4.3 Construction du fichier exécutable .exe :

Pour construire le fichier exécutable .exe, nous avons utilisé la commande build, (Figure 3.23).

```
(drowsy-env) ayemen@pi:~/drowsy-env/drowsiness-detection/detection $ python3 exec.py
build
running build
running build_exe
```

Figure 3.23 : Construction de fichier exécutable .exe.

3.4.4 Résultats de conversion au fichier exécutable :

Après la construction de fichier exec.py, nous avons obtenu l'exécutable .exe (Figure3.24).

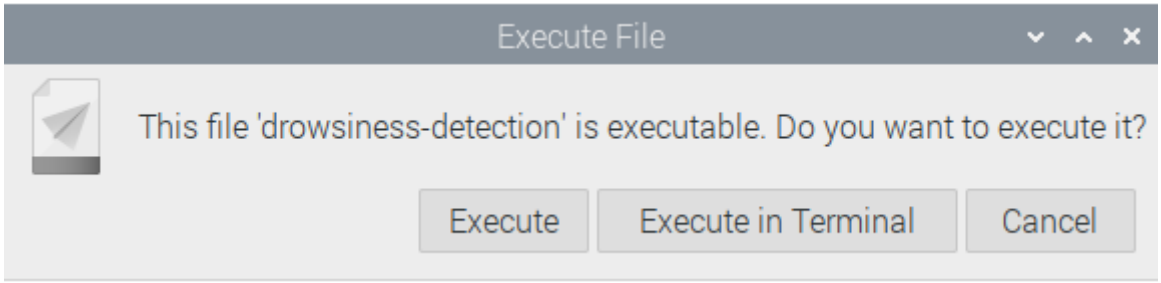

Figure 3.24 : Le fichier exécutable .exe.

Lors de l'exécution du fichier drowsiness-detection.exe, une fenêtre est affichée, nous permettant d'observer l'état de somnolence et du bâillement, comme illustré dans la figure 3.25.

Figure 3.25: Exécution avec le fichier exécutable.

Notre exécutable, détecte le bâillement et la fermeture des Yeux, il affiche plusieurs informations telles que le l'alerte à la somnolence ainsi le EAR et le MAR (yawn).On constate que la détection se fait de manière très acceptable.

3.5 Conclusion

Nous avons abordé plusieurs aspects, notamment le matériel hardware tel que le Raspberry Pi lui-même la caméra. Etc.Ainsi que le logiciel Raspbian.

Lors de l'implémentation sur Raspberry Pi, nous avons constaté que le temps de réponse était plus élevé par rapport à la simulation. Cela s'explique par le fait que le processeur du Raspberry Pi est moins puissant que celui du PC utilisé pour la simulation.

De plus, nous avons constaté que la méthode de détection de la somnolence utilisant la bibliothèque dlib était plus performante que la méthode basée sur TensorFlow.

Enfin, pour rendre les modèles développés sur Raspberry Pi utilisable d'une façon optimale, nous l'avons converti en un fichier exécutable en utilisant la bibliothèque cx_Freeze, permettant ainsi d'embarquer le système sur le Raspberry Pi.

Ces résultats soulignent l'importance de prendre en compte les spécificités matérielles lors de l'implémentation d'un projet sur Raspberry Pi, ainsi que le choix judicieux des bibliothèques pour optimiser les performances du système.

Conclusion générale

Ce projet nous a permis d'explorer plusieurs méthodes de détection de somnolence au volant du conducteur pendant la conduite. Nous avons choisi deux méthodes pour développer un système de détection de somnolence basé sur l'apprentissage profond, en particulier l'architecture CNN.

Dans la simulation du modèle CNN, nous avons obtenu des résultats impressionnants. Cependant, lors de l'implémentation sur Raspberry Pi, nous avons rencontré des limitations en raison de l'incompatibilité de la bibliothèque TensorFlow avec cette plateforme et des performances limitées du Raspberry Pi 3. Cela a affecté les performances du système et n'a pas atteint nos attentes.

Parallèlement, nous avons réalisé une étude comparative avec une autre méthode basée sur le modèle shape predictor. Cette méthode nous a donné des résultats acceptables et se rapproche des performances observées en simulation.

Pour conclure notre développement, nous avons converti le système en un fichier exécutable afin de créer un système embarqué qui peut être utilisé sans avoir à installer les bibliothèques et dépendances nécessaires.

Bien que des défis aient été rencontrés lors de l'implémentation sur Raspberry Pi, ce projet nous a permis d'explorer différentes approches de détection de somnolence par deep learning. Les résultats obtenus en simulation ainsi que les performances acceptables de la méthode basée sur le modèle shape predictor montrent le potentiel de ces approches pour la détection de la somnolence dans des contextes réels.

Des perspectives d'amélioration futures incluent l'exploration de solutions alternatives pour optimiser les performances sur Raspberry Pi, ainsi que l'utilisation de Raspberry pi plus performante (Raspberry pi 4) et caméra Raspberry pi IR pour une meilleure vision de jour comme de nuit, et améliorer la précision de la détection de

somnolence. Par la suite, on peut dire que le cahier des charges proposé a été largement respecté et même qu'il a été enrichit par une implémentation d'une autre méthode de détection de somnolence ainsi qu'une par une étude comparative entres les deux méthodes développées.

Lors de la réalisation de ce projet, nous avons rencontré des difficultés en utilisant des outils de développement. En effet, l'appareillage dont nous disposions était insuffisant pour supporter les versions de bibliothèque tensorflow et dlib leurs installations a dû nous prendre beaucoup de temps, parce qu'il fallait les installer et les réinstaller plusieurs fois. Ces difficultés nous ont réellement freinés dans notre travail de conception, le futur étudiant doit donc prendre en compte ce côté pour éviter ce type de tempsmort.

Enfin, ce travail a permis l'enrichissement personnel de nos connaissances. En effet, il nous a permis d'explorer le domaine de la vision par ordinateur, et nous avons pu compléter notre formation dans les systèmes embarqués, puisque nous avons approfondi nos connaissances en touchant au Deep Learning et outils de programmation (python,Raspbian) les, un autre domaine aussi intéressant que celui de l'électronique. Nous espérons du plus profond de nous-mêmes que notre vie scientifique et professionnelle sera encore plus enrichissante et remplie de succès.

BIBLIOGRAPHIE

[1] C. Solomon and T. Breckon, Fundamentals of digital Image Processing, Hoboken: wileyBlackwell, 2011.consulter le 12/03/2023

[2].L. Diane, cours de traitement d'images, Nice: Laboratoire I3S, 2004, p. 10. Consulter le 15/03/2023

[3] G. Peyré., "le traitement numérique des images," Archive ouverte HAL, 2011. Consulter le 20/03/2023

[4] M. Fatiha, "Détection et Suivi d'Objets en Mouvement Dans Une Séquence d'Images," Université Mohamed Boudiaf, Oran, 2011. Consulter le 20/03/2023

[5]H. KHAMELI and H. KALLAL, "Segmentation d'images satellitaires par Cuckoo," UNIVERSITE ABDELHAMID IBN BADIS, MOSTAGANEM, 2012. Consulter le 20/03/2023

[6]"ROAD SAFETY FACTS," ASIRT, 2022. [Online]. Available: https://www.asirt.org/safe-travel/road-safety-facts

[7] "Accidents de la circulation, Plus de 32.200 accidents de la route, 1.105 décès et 40.000 blessés en 2022 : Carnage sur les routes", elmoudjahid, 2022. [Online].Available:https://www.elmoudjahid.dz/fr/dossier . Consulter le 21/03/2023

[8] S. Hu and G. Zheng, "Driver Drowsiness Detection with Eyelid related Parameters by Support Vector Machine," *Expert Systems with Applications*, vol. 36, pp. 7651-7658, 2009. Consulter le 20/03/2023

[9]W. Zhang, B. Cheng and Y. Lin, "Driver drowsiness recognition based on computer vision technology," Tsinghua Science and Technology, vol. 17, pp. 354-362, 2012. Consulter le 01/04/2023

[10]A. Rosebrock, "Drowsiness detection with OpenCV," 08 May 2018. [Online]. Available: https://www.pyimagesearch.com/2017/05/08/drowsiness-detection-opencv/. Consulter le 04/04/2023

[11] Boukraa, Mohamed, et Abdelkader Boukraa. « L'interprétabilité en apprentissage machine :Un regard sur les réseaux de neurones artificiels profonds ». ResearchGate, août 2020, [Online].Available :

https://www.researchgate.net/publication/343135807_L%27interpretabilite_en_apprentissage_machine_Un_regard_sur_les_reseaux_de_neurones_artificiels_profonds consulter le 12/04/2023

[12]Namane, « cours IA3 master 2 électronique des systèmes embarqués », Université Saad Dahled, 2021. . Consulter le 13/04/2023

[13] Cristina Heghedus1 , Anton Shchipanov2 , Chunming Rong1 1Department of Computer Science and Electrical Engineering, University of Stavanger, Norway 2NORCE- Norwegian Research Center, Stavanger, Norway. . Consulter le 13/04/2023

[14]G. Reza, G. Marnim and A. Vassilis, "A Realistic Dataset and Baseline Temporal Model for Early Drowsiness," CoRR, vol. abs/1904.0 (7312, 2019. consulter le 12/04/2023

[15] Goodfellow I, Bengio Y, Courville A, Bengio Y. Deep learning, vol. 1. Cambridge: MIT press; 2016. Consulter le 13/04/2023

[16] K. Dahmane, « Analyse d'images par méthode de Deep Learning appliquée au contexte routier en conditions météorologiques dégradées», thèse de doctorat, École doctorale : Sciences Pour l'Ingénieur, 2020. Consulter le 16/04/2023

[17] Serena Raju, « yawn-eye-dataset-new », Kaggle,2020,[online],disponible :

/ https://www.kaggle.com/datasets/serenaraju/yawn-eye-dataset-new consulter le : 21/03/2023

[18] Rukshan Pramoditha, « L'architecture du réseau neuronal convolutif (CNN) expliquée en langage simple à l'aide de schémas simples », towards data science, juin20 2023

[Online].Available :https://towardsdatascience.com/convolutional-neural-network-cnn-architecture-explained-in-plain-english-using-simple-diagrams-e5de17eacc8f

[19] Claire Giot, Marion Hay, Jacques Taillard, Damien Davenne, Nicolas Bessot,«Vers une nouvelle approche de la détection de la somnolence : validation de l'Objective Sleepiness Scale en situation de conduite simulée».Médecine du Sommeil,Volume 19, Issue 1,2022,P 59-60,ISSN 1769-4493,https://doi.org/10.1016/j.msom.2022.01.023.

[20]ABDAT, Faiza. Reconnaissance automatique des émotions par données multimodales : expressions faciales et signaux physiologiques. Université de Metz, France, 2010.

[21] *Manitra Tsilavina RAZAFIMANDIMBY. «Détection et reconnaissance de visages», 2016.* Récupéré de

http://biblio.univantananarivo.mg/pdfs/razafimandimbyManitraT_MP_MAST2_16.pdF

[22] kdnuggets Dlib: Library for Machine Learning, Disponible sur: https://www.kdnuggets.com/2014/06/dlib-library-machine-learning.html

[23] Davis E. King, Dlib-ml: A Machine Learning Toolkit, maching learning, Northrop Grumman ES, ATR and Image Exploitation Gsroup, Journal of Machine Learning Research 10 (2009)

[24] T. Soukupová, «Real-Time Eye Blink Detection using Facial Landmarks».2016

[29] Debian GNU/Linux Installation Guide,June 11, 2023.[Online].Available:

https://www.debian.org/releases/stable/s390x/install.en.pdf

[25] Christine Dewi, Rung-Ching Chen, Xiaoyi Jiang,Hui Yu4, «Adjusting eye aspect ratio for strong eye blink detection based on facial landmarks»,Pub Med Central,[Online].Available : https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9044337/ consulter le 01/06/2023.