الجمهورية الجزائرية الديمقراطيةالشعبية
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

كليةالتكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

Mémoire de Master

Filière : Électronique
Spécialité : Électronique des systèmes embarqués

Présenté par

Hadj Moussa Abdelhamid

&

Benkherouf Mohamed Samy

Application Java mobile Embarquée pour la détection de la somnolence par intelligence artificielle

Proposé par : Bougherira Hamida

Année Universitaire 2022-2023

Remerciements

Tout d'abord, on aimerait remercier Dieu de nous avoir donné le courage et la volonté de faire ce travail.

On est profondément reconnaissant envers notre mentor Mme.Bougherira Hamida, pour ses conseils inlassables et Tous ses conseils, orientations et corrections.

On remercie chaleureusement nos enseignants.

On tient également à remercier les membres du jury qui ont eu la gentillesse de nous décerner leur honneur.

On tient à remercier nos familles de nous avoir soutenu durant nos études universitaires.

Enfin un grand merci à tous nos amis.

ملخص: لقد بحثنا في طرق مختلفة للكشف عن النعاس (معدل ضربات القلب وتخطيط القلب وتخطيط كهربية القلب وغيرها)، وفي النهاية اخترنا طريقة العين التي تهدف إلى اكتشاف العينين والإبلاغ عند إغلاق العينين. لقد طورنا تطبيقًا للهاتف المحمول لـ ANDROID باستخدام بيئة Android Studio ولغة برمجة Java. بعد إنشاء وتنظيم المجلدات المطلوبة بواسطة استوديو Android ثم إنشاء قاعدة تعليمنا ومعالجتها مسبقًا، قمنا بتطبيق خوارزمية Deep Learning SSD. لقد اختبرنا نظامنا بنجاح على الكمبيوتر الشخصي. ثم قمنا بدمج تطبيقنا على هاتف ذكي يعمل بنظام Android، وحصلنا على نتائج اكتشاف النعاس في الوقت الفعلي، وهو أمر حاسم للغاية. تطبيق الهاتف المحمول الخاص بنا فعال وفعال.

الكلمات الرئيسية: السائقون النائمون؛ تطبيق الهاتف المحمول؛ Java؛ الذكاء الاصطناعي؛ Android ووضع العين.

Résumé : Nous avons fait des recherches sur différentes méthodes de détection de somnolence (rythme cardiaque, ECG, EEG et autres), et au final nous avons choisi la méthode oculaire qui a pour but de détecter les yeux et de signaler quand les yeux se ferment. Nous avons développé une application mobile, pour ANDROID en utilisant l'environnement Android Studio, et le langage de programmation Java. Après avoir créé et organisé les dossiers requis par Android studio, puis créé et prétraité notre base d'apprentissage, nous avons implémenté l'algorithme de Deep Learning SSD. Nous avons testé, avec succès, notre système sur PC. Nous avons ensuite embarqué notre application sur un smartphone Android, et obtenu des résultats de détection de somnolence en temps réel, très concluants. Notre application mobile est disponible fonctionnelle et efficace.

Mots clés : Conducteurs somnolents ; application mobile ; Java ; Intelligence artificielle ; Android et positionnement des yeux.

Abstract: We researched different methods of detecting drowsiness (heart rate, ECG, EEG and others), and ultimately, we chose the ocular method that aims to detect the eyes and report when the eyes close. We have developed a mobile application for ANDROID using the Android Studio environment and the Java programming language. After creating and organizing the folders required by Android studio and then creating and preprocessing our learning base, we implemented the Deep Learning SSD algorithm. We have successfully tested our system on PC. We then integrated our application on an Android smartphone, and obtained results of detection of sleepiness in real time, very conclusive. Our mobile app is functional and efficient.

Keywords: Sleepy drivers; mobile app; Java; Artificial intelligence; Android and eye positioning.

Listes des acronymes et abréviations

API: Application Programming Interface

APK: Android Package Kit

APK: Android Package Kit

ECG : Électrocardiographie

EEG : Électroencéphalographie

HDD: Hard Disk Drive

HDD: Hard Disk Drive

HRV: Heart Rate Variability

IA : Intelligence Artificielle

JDK: Java Development Kit

NSC: Noyau suprachiasmatique

PC: Personal Computer

RAM: Random Access Memory

RAM: Random Access Memory

SDK: Software Development Kit

SSD: Single Shot MultiBox Detector

USB : Universal Serial Bus

USB: Universal Serial Bus

XML: Extensible Markup Language

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

Dans un monde en constante évolution, où la technologie joue un rôle de plus en plus important dans notre vie quotidienne, la sécurité et le bien-être des individus sont des préoccupations majeures. La somnolence au volant, par exemple, est un problème qui peut entraîner des accidents graves et même mortels. Afin de lutter contre ce fléau, notre projet a pour objectif la création d'une application mobile innovante qui vise à détecter, en temps réel, la somnolence chez les conducteurs en tirant parti des technologies de pointe telles que le langage de programmation Java et l'intelligence artificielle

Notre application mobile fournit un outil efficace et convivial permettant de surveiller et d'alerter les utilisateurs en cas de signes de somnolence. Pour ce faire, nous avons choisi d'utiliser le langage Java en raison de sa portabilité, de sa performance et de sa compatibilité avec les systèmes d'exploitation mobiles les plus populaires, tels qu'Android [1] De plus, Java offre un large éventail de bibliothèques et de frameworks qui facilitent le développement d'applications mobiles complexes et robustes.

L'intelligence artificielle (IA) est également un élément clé de notre application. Nous avons choisi le réseau de neurones profond SSD car il permet d'analyser et d'interpréter les données collectées pour détecter les signes de somnolence de manière précise et en temps réel [2], En combinant des techniques d'apprentissage profond (deep learning) et de traitement d'images, notre application est capable de reconnaître les comportements et les expressions faciales associés à la somnolence, tels les clignements d'yeux et les mouvements de la tête.

Ce mémoire présente en détail les différentes étapes du développement de l'application, depuis la conception initiale jusqu'à la mise en œuvre et les tests. Pour cela, nous l'avons organisé comme suit :

Dans le chapitre un, nous présentons un état de l'art des méthodes de détection.

Le chapitre deux décrit d'une part, la conception de notre application, sa structure ; ses algorithmes…d'autre part, nous détaillons la création de la base d'apprentissage (et son prétraitement), que nous avons utilisée pour l'apprentissage profond du réseau de neurones à convolution SSD.

L'implémentation, la mise en œuvre, et les résultats de notre application sont exposés dans le chapitre trois.

Notre conclusion aborde les défis rencontrés lors de la création de cette application, ainsi que les solutions adoptées pour les surmonter.

Chapitre 1 Les généralités des expressions faciales

1.1 INTRODUCTION

Notre projet a pour but le développement d'une appli Mobile en langage Java sur Android studio de détection de la somnolence chez un conducteur à partir de ses caractéristiques faciales relever sur des images en temps réel. Nous présentons, dans ce chapitre un état de l'art de méthode et algorithmes de détection chez un conducteur. Ainsi que le principe de développement d'une application mobile sous l'environnement de développement Android studio

1.2 Historique

L'expression faciale est un aspect important du comportement et de la communication non verbale, et les modifications du visage sont visuellement apparentes. L'expression faciale a été étudiée par Darwin et Duchenne de Boulogne dès le XVIIIe siècle, et elle a joué un rôle important dans l'étude des émotions depuis les travaux de Sylvain Tomkins dans les années 1960. Ses étudiants Paul Ekman et Carol Izzard ont défendu un nombre limité d'idées sur les émotions de base associées aux expressions faciales automatiques, universelles et innées [3][4].

Dans la seconde moitié du XVIIIe siècle, le neuroscientifique Duchenne de Boulogne a mené une série d'expériences sur les expressions faciales émotionnelles. Il utilise la photographie et la stimulation électrique des muscles faciaux pour mettre en évidence les mouvements associés à l'expression émotionnelle. En particulier, il a remarqué des contractions de l'orbicularis oculi, les muscles autour des yeux peuvent faire la distinction entre un sourire sincèrement heureux et un sourire volontaire. Ces résultats ont été confirmés et complétés par des études menées par Paul Ekman et son équipe dans les années 80. Ekman précise que la plupart d'entre nous ne peuvent pas contracter volontairement l'orbicularis oculi, et ceux qui peuvent volontairement contracter l'orbicularis ne peuvent généralement pas contracter les deux muscles en même temps. De plus, les sourires de Duchen ne sont généralement associés qu'à une activité symétrique dans le lobe frontal et sont considérés comme un signal affectif positif [5].

Aujourd'hui, la technologie de reconnaissance faciale est utilisée dans les espaces publics pour soutenir les forces de l'ordre en identifiant les personnes susceptibles d'avoir le hoquet, en luttant contre la fraude au passeport, en identifiant les personnes disparues.

1.3 Méthode de détection de la somnolence

1.3.1 Méthode physiologique

La détection de la somnolence implique plusieurs processus physiologiques qui se produisent dans le corps et le cerveau. Les principaux mécanismes impliqués dans la détection de la somnolence sont la régulation du sommeil et de l'éveil, la régulation de l'activité cérébrale, et la modulation de la vigilance.

La régulation du sommeil et de l'éveil est assurée par le système circadien, une horloge biologique interne qui contrôle le rythme veille-sommeil. Cette horloge est située dans une région du cerveau appelée le noyau suprachiasmatique (NSC) Comme le montre la figure 1, qui reçoit des signaux de la lumière et de l'obscurité pour ajuster le rythme circadien. Les signaux lumineux sont captés par les cellules ganglionnaires de la rétine, qui projettent l'information au NSC via le nerf optique [6].

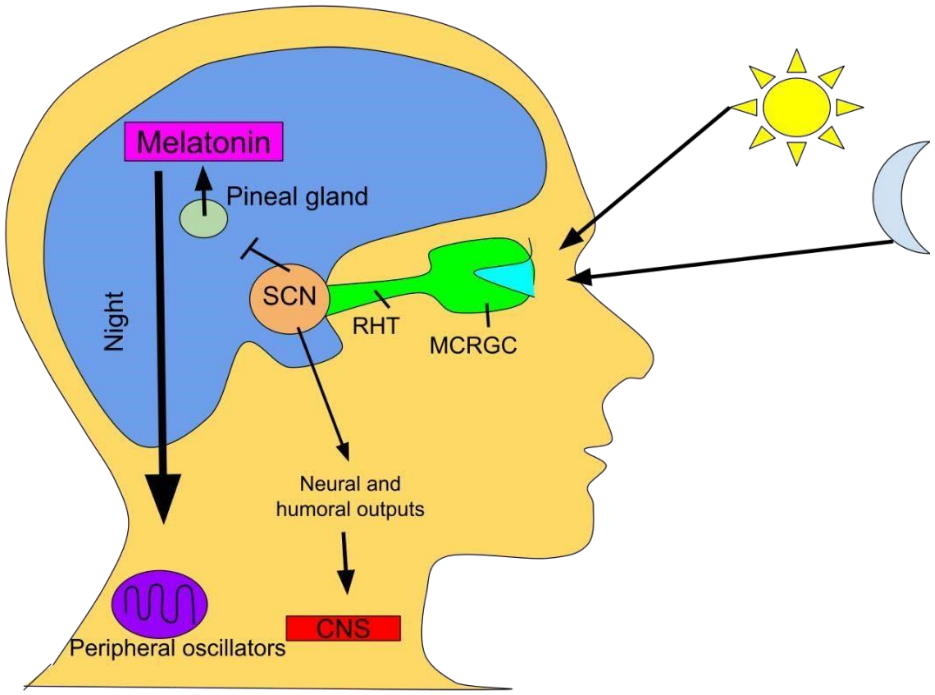

Figure 1 : La méthode de réception (SCN) des signaux de la lumière et de l'obscurité pour ajuster le rythme circadien

1.3.2 Méthode EEG

L'électroencéphalographie (EEG) est généralement une technique non invasive convertit l'activité du cortex cérébral en signaux électriques utilisables. Un électroencéphalographe est un appareil électronique utilisé pour effectuer des opérations ce signal est enregistré par des électrodes placées à la surface du cuir chevelu.

Transmettre ce signal à la carte électronique sans modifier ses propriétés. Enregistrer Plusieurs dérivations peut être en cours en même temps, chaque dérivation utilise une chaîne des acquisitions comme le montre la figure 2.

Le terme EEG fait référence à l'enregistrement par sous le titre Scalp Leads and électrocorticogram (ECOG), enregistrez via des électrodes placées sur de l'écorce nue. Pour pouvoir mesurer l'EEG, il est nécessaire de connaître l'anatomie et la fonction du système nerveux et sa physiologie, puisque l'EEG est basé sur cette connaissance.

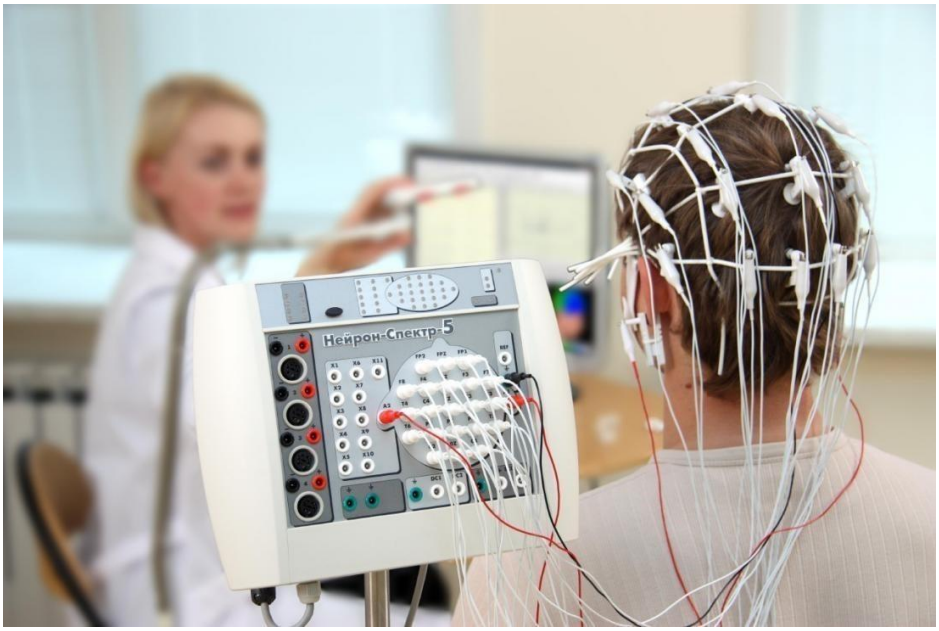

Figure 2 : Machine EEG.

1.3.3 Méthode ECG

La méthode ECG (électrocardiographie) filière pour la détection de la somnolence est basée sur l'analyse des changements dans les signaux ECG qui se produisent lorsque le corps est en état de somnolence ou de fatigue comme le montre la figure 3. Plusieurs études ont montré que les signaux ECG peuvent fournir des informations précieuses sur l'état de somnolence d'un individu.

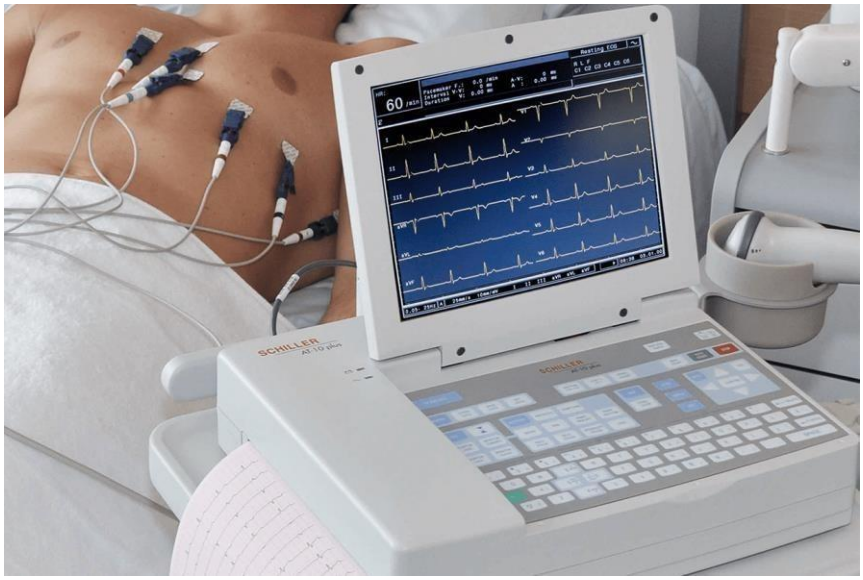

Figure 3 : Machine ECG.

La détection de la somnolence à l'aide de l'ECG se produit en analysant les caractéristiques des signaux ECG, telles que l'intervalle de temps entre les battements cardiaques, la variabilité de la fréquence cardiaque et les changements dans la forme des ondes du signal ECG. Ces caractéristiques sont ensuite utilisées pour déterminer le niveau de somnolence ou de fatigue d'un individu [7].

Des techniques telles que l'analyse spectrale de la fréquence cardiaque ont été utilisées pour identifier les caractéristiques des signaux ECG associées à la somnolence comme le montre la figure 4. En particulier, la baisse de la fréquence cardiaque et de la variabilité de la fréquence cardiaque a été liées à un état de somnolence. Il est important de noter que la détection de la somnolence à l'aide de l'ECG ne peut pas être utilisée à elle seule pour diagnostiquer un trouble du sommeil ou un problème de somnolence. Cependant, elle peut fournir des informations supplémentaires pour aider à évaluer l'état de somnolence d'un individu [8].

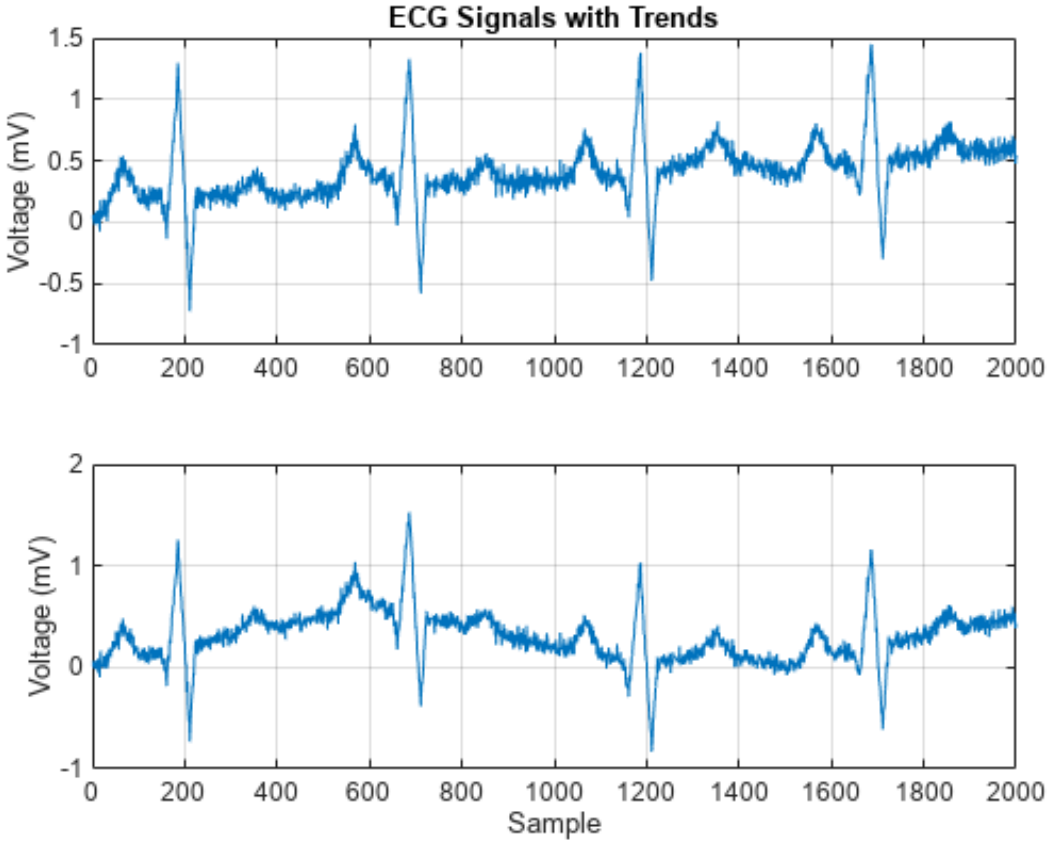

Figure 4 : Exemple de signal d'électrocardiogramme (ECG)

1.3.4 Approche basée sur le véhicule

La deuxième méthode pour mesurer la somnolence du conducteur, et l'utilisation de capteurs placés sur divers composants du véhicule.

Dans ce contexte, les capteurs peuvent être utilisés pour mesurer différents paramètres tels que la vitesse, la direction du volant, la position de la pédale d'accélérateur et de frein, la pression de l'air dans les pneus, les changements de voie, etc. Ces données peuvent ensuite être analysées pour détecter des comportements potentiellement dangereux associés à la somnolence du conducteur, tels que des mouvements erratiques ou une baisse de la fréquence cardiaque comme le montre la figure 6 [9].

Une étude récente publiée dans la revue IEEE Access a proposé une méthode de détection de la somnolence du conducteur à l'aide de capteurs de pression installés sur le siège du conducteur. Cette méthode a été testée sur un groupe de conducteurs qui ont effectué une tâche de conduite sur un simulateur de conduite, et les résultats ont montré une corrélation significative entre les données de pression et la somnolence du conducteur [10].

Figure 5 : un détecteur de somnolence au volant.

1.4 Méthode basée sur traitement d'image

1.4.1 Filtre de Gabor

Un filtre Gabor est un type de filtre utilisé pour l'analyse de textures dans les images. Il est basé sur les fonctions de Gabor, qui sont des fonctions mathématiques utilisées pour décrire les signaux oscillants locaux dans une image. Les filtres Gabor sont utilisés pour extraire les caractéristiques d'une image en capturant les changements de phase et d'amplitude des signaux oscillants dans différentes directions et à différentes fréquences spatiales. Ils sont largement utilisés dans la vision par ordinateur pour des tâches telles que la détection d'objets, la reconnaissance de formes et la segmentation d'image [11].

1.4.2 Extraction des caractéristiques avec Filtre de Gabor

L'extraction de caractéristiques avec des filtres Gabor peut être utilisée pour la détection de visages en analysant les textures de l'image. Les filtres Gabor peuvent être appliqués à des images de visages pour extraire des caractéristiques telles que la texture de la peau, les contours et les formes du visage. Cette analyse permet d'identifier les régions de l'image qui sont les plus susceptibles de contenir un visage [12].

1.4.3 Détection des repères

L'idée d'utiliser des points de repère pour détecter les régions a été inspirée par plusieurs chercheurs. Pour cela, nous utilisons un algorithme basé sur un ensemble d'arbres de régression (détection de repères faciaux. Cette technique utilise une fonction (la différence d'intensité des pixels) pour estimer L'emplacement du point de repère direct. L'algorithme peut détecter rapidement un ensemble de 68 repères sur un visage donné. Après détection de points, certains de ces points permettent de définir trois régions (œil droit et son sourcil, œil gauche et son sourcil, bouche) qui se présentent sous la forme de rectangles [13].
Rectangles, nous avons donc calculé la taille moyenne de chaque rectangle Chaque type de région, après avoir redimensionné toutes les régions taille régulière. Cette opération fait que la taille des vecteurs propres soit Chaque visage dans la base de données est le même comme le montre la figure 6.

a b

Figure 6 : Exemple des régions détecté dans un visage.

1.4.4 Utilisation de filtres de Gabor pour l'extraction de caractéristiques faciales

Dans cette partie, nous discutons du filtre de Gabor et de la façon dont il extrait fonctionnalités et obtenir des fonctionnalités de la région que nous utilisons Un ensemble de filtres de Gabor avec un certain nombre de directions et de fréquences. Dans l'étape suivante, nous verrons comment déterminer cette valeur Le nombre de directions et de fréquences selon l'algorithme génétique Par exemple, le nombre de directions est égal à 5 et la fréquence est égale à deux. Les paramètres détaillés du filtre Gabor sont : 1) Taille du filtre = 3px * 3px. 2) Longueur d'onde = 0,4, 0,8. 3) sens = 5, 2 5, 5, 5, 4) écart type = 3. 5) Rapport d'aspect spatial = 1. 6) Déphasage = /2.

1.5 La Méthode (LBP)

Le modèle binaire local (LBP) est un descripteur d'image de texture qui peut Définit la structure spatiale et le contraste local d'une image ou d'une partie d'image. LBP devient Un descripteur de texture standard. Ce descripteur est largement utilisé en raison de sa simplicité de mise en œuvre Extraction de vecteurs de caractéristiques avec une grande précision de classification.

L'opérateur LBP est robuste aux changements d'échelle de gris uniformes. Déjà considéré C'est également l'un des descripteurs les plus standard et les plus efficaces dans la description de la texture d'image.

L'effet LBP est considéré comme l'une des méthodes d'analyse de texture d'image les plus appropriées [14].

Ont introduit pour la première fois le descripteur LBP, qui est indépendant de la rotation. À ceci méthode, considérez d'abord le voisinage de chaque point de l'image. Intensité des pixels Le centre est alors comparé à l'intensité des pixels voisins. Si l'intensité lumineuse des pixels adjacents est supérieure au pixel central, considérez la valeur de ce voisin dans le motif binaire extrait Égal à un, sinon zéro.

Enfin, une somme binaire pondérée est effectuée sur les valeurs en Modèle de minage binaire, nous obtenons des valeurs de base dix. Cette somme s'appelle Valeur de la lombalgie [15]. Le calcul final de LBP est défini dans l'équation suivante :

$$LLLLP(IIII) = \sum SS(IIpp - IIII) * 2\ pp-1\ pp\ bb=1$$

Où:

- P : le nombre de voisins pour le pixel central.
- $IIII$: intensité lumineuse du pixel central.
- $IIpp$: intensité lumineuse du pixel voisin.

$S(xx)$: fonction de signe pour générer des valeurs binaires est définie comme suit:

S(x) = { 1 si x >= 0; 0 sinon }

Le descripteur LBP est appliqué à tous les pixels de l'image, comme le montre la figure 7. Les histogrammes LBP peuvent être utilisés pour extraire les caractéristiques de texture des images Pour éviter la sensibilité de l'opérateur LBP à la rotation, le voisinage est considéré comme un cercle, Intensité lumineuse d'un point dont les coordonnées ne correspondent pas exactement au centre pixels, mais ils sont déterminés par interpolation.

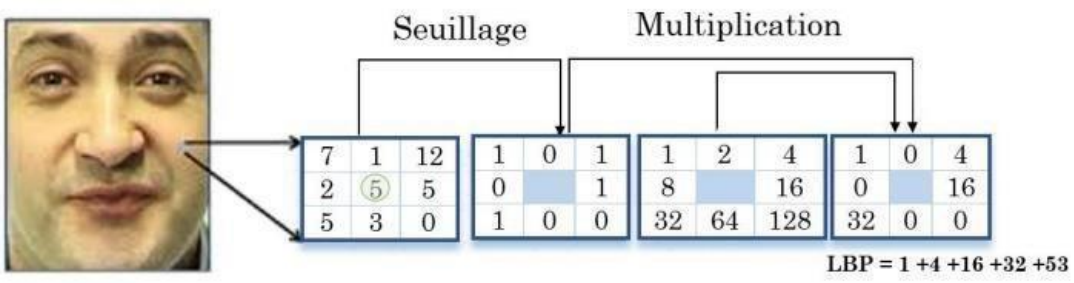

Figure 7 : Exemple de calcule d'un code LBP.

La figure 8 montre un voisinage circulaire avec différents rayons RR et nombre de voisins À partir du point PP. La propriété la plus importante de LBP est son invariance au changement. Monotonie de l'éclairage causée par les différences d'éclairage dans les applications du monde réel. Une autre caractéristique tout aussi importante est la simplicité de ses calculs en temps réel.

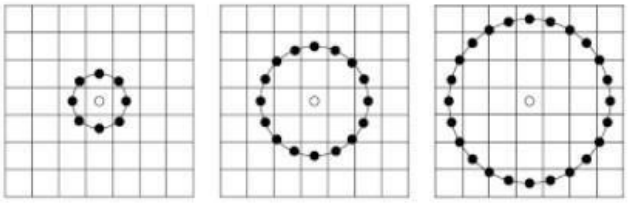

Figure 8 : Voisins symétriques circulaires pour différentes valeurs de PPet RR.

1.5.1 Le descripteur LBP

Différentes valeurs [14]. Lorsque nous n'avons pas d'image pivot, tous les voisins tournent autour du pixel central $IIII$ direction, résultant en des valeurs différentes pour $LLLLPPpp$, rr. Ça signifie $LLLLPPpp$,r dépend des indices des pixels voisins. L'opérateur de rotation droite au niveau du bit est Introduit pour attribuer à chaque mode une valeur unique pour éliminer les destructeurs En raison de l'effet de rotation comme le montre la figure 9. Cet opérateur de rotation indépendant est calculé comme suit :

$$LLPPPP,R\ rrbb\ (xx, yy) = \min(RRRRRR(LLLLPPPP,RR\ rrbb\ (xx, yy),bb), bb \in [0, PP - 1])$$

Où: $RRRRRR$est la fonction de rotation d'un $PP\_bbbbbbbb$ vers la droite et $rrbb$ est l'insensibilité de l'opérateur à la rotation.

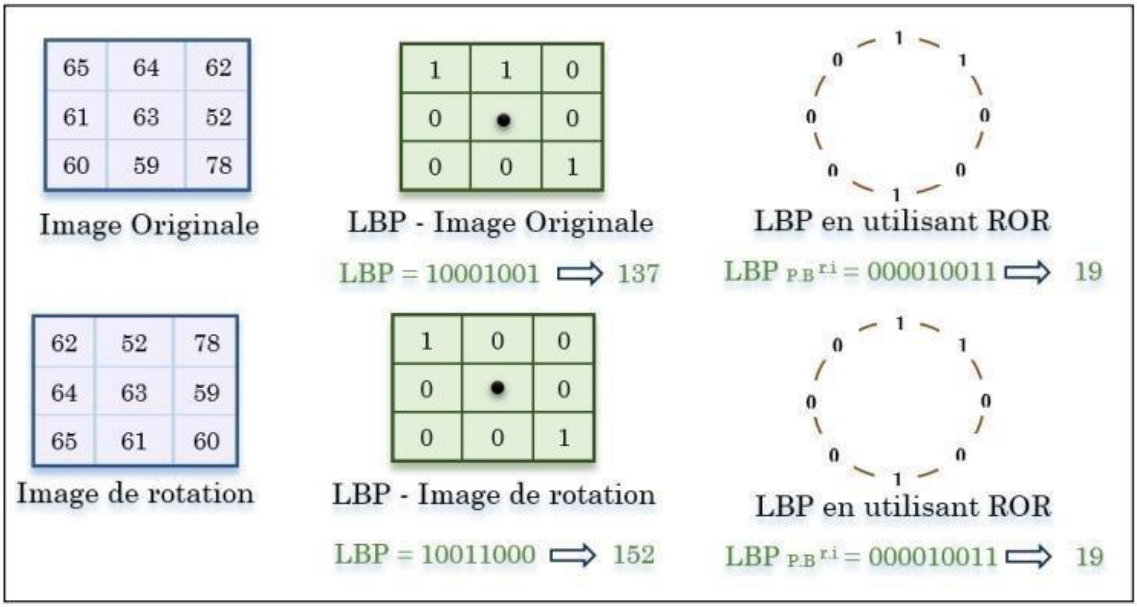

Figure 9 : Un processus LBP utilisant la rotation de bits vers la droite.

1.6 Méthode de deep learning

1.6.1 Protocole SSD

Le protocole SSD (Single Shot Multibox Detection) est un modèle de réseau de neurones profonds conçu pour effectuer la détection d'objets dans des images en temps réel. Il a été développé par Wei Liu, Dragomir Anguelov, et d'autres chercheurs chez Google en 2016.

Le modèle SSD utilise un réseau de neurones convolutifs pour extraire des caractéristiques d'une image, puis utilise des prédicteurs de boîtes englobantes pour détecter des objets à partir de ces caractéristiques. La méthode SSD diffère des autres approches de détection d'objets, car elle effectue la détection et la localisation des objets en une seule passe (single shot), plutôt que d'utiliser plusieurs étapes comme le montre la figure 10 et 11.

Le modèle SSD est souvent utilisé dans des applications d'intelligence artificielle, telles que la reconnaissance de visages et peut être utilisé pour détecter des caractéristiques de la somnolence, telles que les yeux qui se ferment lentement ou la tête qui commence à tomber en avant. En utilisant un modèle SSD entraîné sur des données d'apprentissage incluant des images de personnes somnolentes, le modèle est capable de détecter ces caractéristiques en temps réel à partir d'une source vidéo, comme une caméra de surveillance ou une caméra. Il est également utilisé dans certaines bibliothèques d'apprentissage en profondeur, telles que TensorFlow et PyTorch rotation [17].

Figure 10 : Détection d'objets à l'aide d'un protocole SSD.

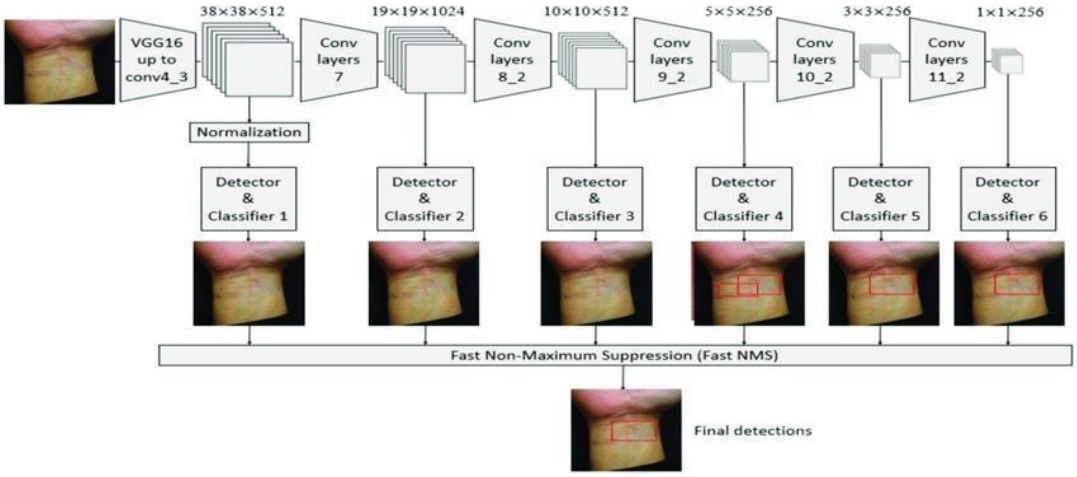

Figure 11 : un exemple schématique du fonctionnement du protocole SSD.

1.7 Android studio

Android Studio est l'environnement de développement intégré (IDE) officiel des applications Android. Basé sur le puissant outil de développement et d'édition de code d'IntelliJ IDEA, Android Studio offre encore plus de fonctionnalités qui améliorent votre productivité lorsque vous créez des applications Android [17].

Figure 12 : logo Android

Il présente de nombreux avantages pour les développeurs, notamment : [18]

- **Éditeur de code avancé**:

Android Studio dispose d'un éditeur de code riche en fonctionnalités, qui offre une mise en évidence de la syntaxe, une complétion automatique, un re factoring intelligent et des outils de navigation.

- **Émulateur d'appareil Android:**

Il est possible de tester les applications Android sur une variété d'appareils virtuels en utilisant l'émulateur d'appareil Android intégré dans Android Studio. Cela permet aux développeurs de tester leurs applications sur différents appareils et configurations, sans avoir besoin de posséder physiquement tous ces appareils.

- **Système de compilation Gradle:**

Android Studio utilise Gradle pour automatiser le processus de création et de génération de l'APK final.

Ce système permet de compiler automatiquement les dépendances et les bibliothèques tierces, de gérer les ressources et de générer le code nécessaire pour construire l'application.

- **Intégration avec d'autres outils Google :**

Android Studio est étroitement intégré avec d'autres outils Google, tels que Firebase, Google Cloud Platform et Google Play Store. Cela facilite le développement, le déploiement et la gestion des applications Android.

- **Prise en charge de plusieurs langages :**

Android Studio prend en charge plusieurs langages de programmation, notamment Java, Kotlin et C++. Il offre également la possibilité de créer des applications Android pour les téléphones, les tablettes, les montres connectées, les téléviseurs et d'autres appareils formés comme le montre la figure 13.

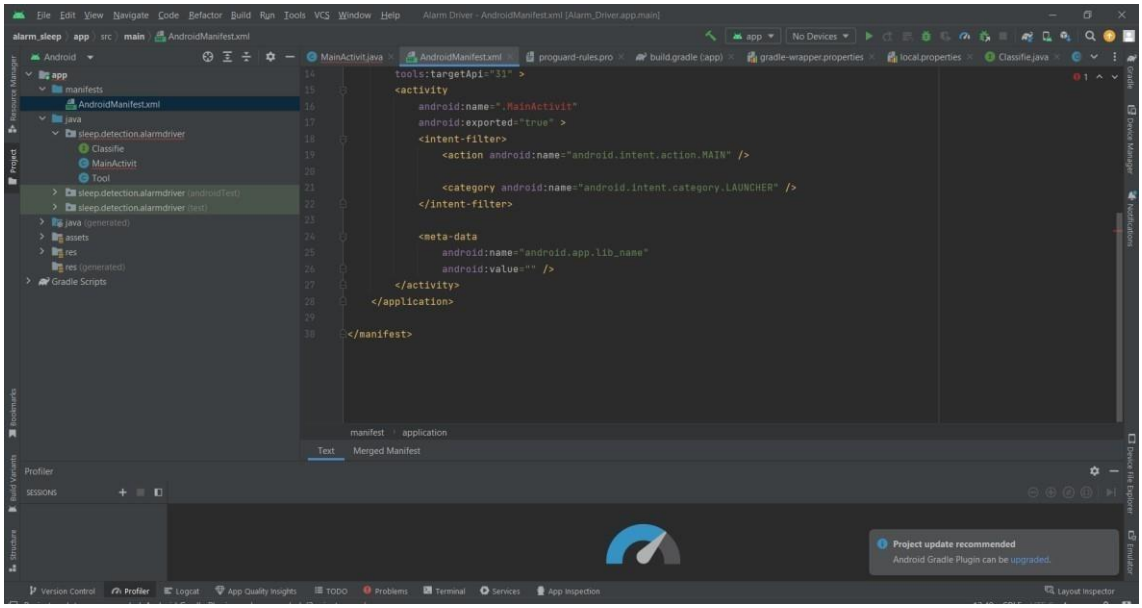

Figure 13 : Interface de logicielle "Android studio".

1.8 Conclusion

Dans ce chapitre, nous présentons les principaux aspects de la reconnaissance faciale et leur importance dans l'évolution et l'architecture globale des systèmes de reconnaissance faciale, et nous donnons un aperçu de certaines méthodes de reconnaissance des expressions faciales, des méthodes globales, des méthodes locales et des méthodes hybrides.

Dans la section suivante, nous présenterons la reconnaissance des expressions faciales et les différentes méthodes qui permettent leur extraction.

Chapitre 2 Analyse et conception de l’application

2.1 Introduction

Ce chapitre se concentre sur les étapes de la création de l'application mobile. Nous allons décrire dans un premier temps la création et le prétraitement de la base de données utilisée pour l'apprentissage de l'architecture de Deep Learning SSD que nous avons implémenté. Nous présenterons l'architecture générale de notre système, l'environnement de développement et une description des outils utilisés. De plus, nous allons passer en revue les langages de développement et de programmation utilisés pour la création de l'application mobile.

2.1.1 L’objectif de l’application

L'objectif de notre application est de détecter quand une personne devient somnolente ou endormie, et de l'alerter pour qu'elle prenne une pause ou arrête de conduire. Cela aide à prévenir les accidents causés par la conduite en état de somnolence.

2.1.2 Le fonctionnement de l'application Android

Le diagramme de fonctionnement de notre application est la surveillance des signes de somnolence, l'analyse des données et la notification de l'utilisateur. Pour mieux comprendre le fonctionnement de cette application, la figure 14 en illustre les étapes.

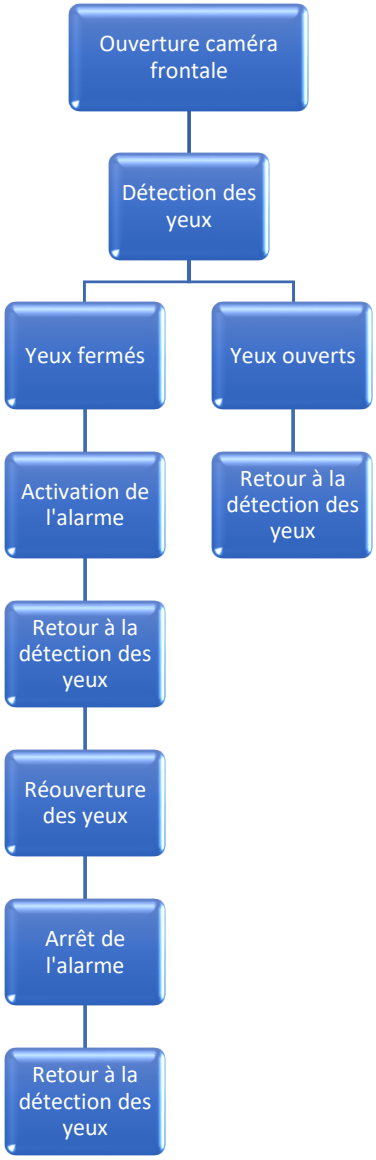

Figure 14 : diagramme de système propose

2.1.3 La création d'une application mobile

Lorsqu'il s'agit de créer une application mobile, il est essentiel d'avoir une vision claire de chaque étape du processus, C'est pourquoi un diagramme peut aider à suivre le progrès du projet et à identifier les problèmes potentiels plus rapidement, ce qui peut aider à réduire les délais, Comme le montre la figure 15.

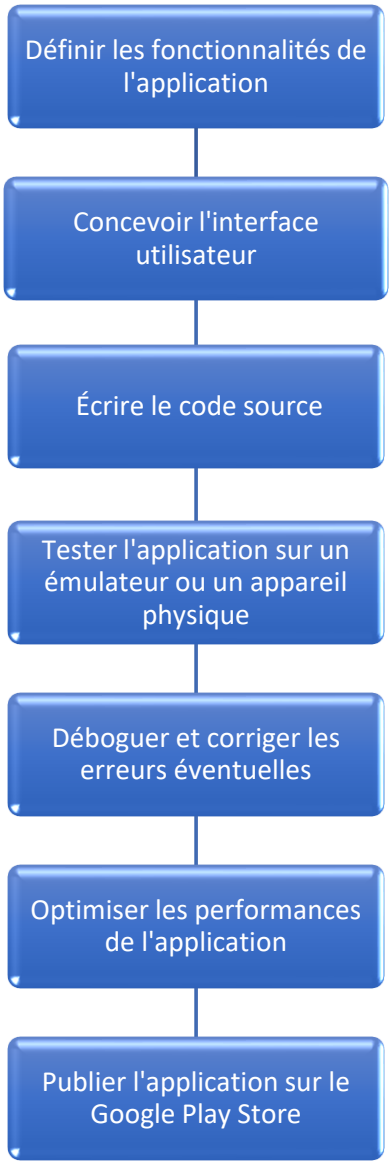

Figure 15 : diagramme de La création d'une application mobile

2.2 Partie pc

La partie "PC" de notre application de détection de la somnolence fait référence à notre utilisation d'un ordinateur personnel pour exécuter les tâches de traitement d'images, d'analyse et d'interprétation des données afin de détecter les signes de somnolence chez les utilisateurs. Cela implique notre utilisation de matériels informatiques appropriés, de logiciels de traitement d'images et d'une interface utilisateur pour faciliter l'interaction avec notre système

2.2.1 Création de la base de données

La base de données joue un rôle essentiel dans votre application de détection de la somnolence en permettant le stockage, l'accès, l'analyse des données, ainsi que l'évolutivité et l'extensibilité de l'application.

2.2.2 Acquisition d'images

Notre acquisition d'images est la première étape dans laquelle nous collectons les images nécessaires pour alimenter notre application mobile. Il peut s'agir de photos prises à l'aide de notre caméra de téléphone, de captures d'écran, d'images téléchargées depuis Internet, ou de tout autre moyen nous permettant d'obtenir des données visuelles. Cette étape est cruciale car la qualité et la pertinence des images que nous collectons auront un impact direct sur nos résultats et notre expérience utilisateur de notre application. Ce diagramme représente les étapes d'Acquisition d'images Comme le montre la figure 16 :

Figure 16 : diagramme des étapes d'acquisition d'images

2.2.3 Annotation des images

Une fois que nous avons collecté les images, l'étape suivante consiste à les annoter. L'annotation d'images consiste à ajouter des métadonnées, des balises ou des étiquettes à chaque image pour décrire et identifier les éléments ou les caractéristiques spécifiques qu'elles contiennent. Cela peut inclure des tâches telles que la détection d'objets, la segmentation sémantique, l'identification de visages, la reconnaissance de texte, etc. L'annotation d'images est essentielle pour entraîner nos modèles d'apprentissage automatique et pour fournir une compréhension contextuelle des images dans notre application mobile. Cet organigramme représente les étapes de l'annotation d'image comme le montre la figure 17 :

:

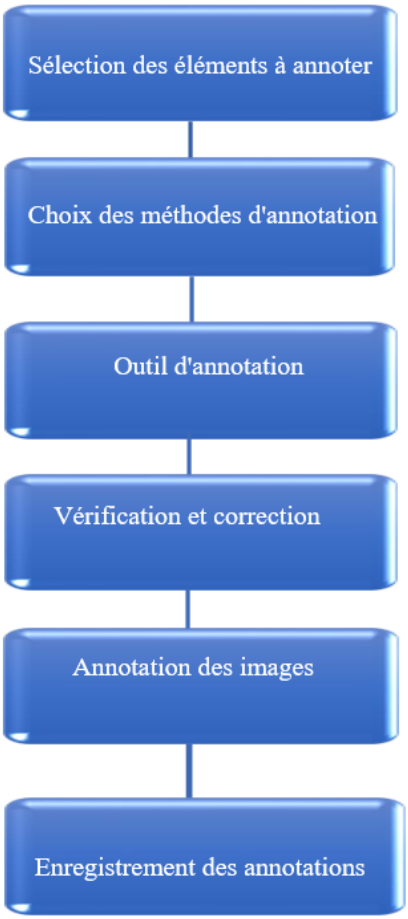

Figure 17 : diagramme des étapes Annotation des images

Dans notre cas nous avons annoté les images en encadrant la partie du visage contenant les deux yeux. Nous avons associé à chacune des 230 images annotées, la propriété de somnolence, ou pas, selon que les yeux étaient fermés ou ouverts respectivement

2.2.4 Traitement d'images

Une fois que nous avons acquis et annoté les images, nous pouvons passer à l'étape de traitement des images. Le traitement d'images englobe un large éventail de techniques et d'opérations visant à modifier, améliorer ou analyser les images de manière algorithmique. Cela peut inclure des opérations telles que le redimensionnement, le recadrage, la correction des couleurs, la suppression du bruit, la reconnaissance de formes, la classification d'images, la détection de contours, etc. Le traitement d'images est souvent utilisé pour préparer les données d'images avant leur utilisation dans notre application mobile, afin de les rendre plus adaptées à l'objectif spécifique de notre application. Ce diagramme représente les étapes de traitement d'image :

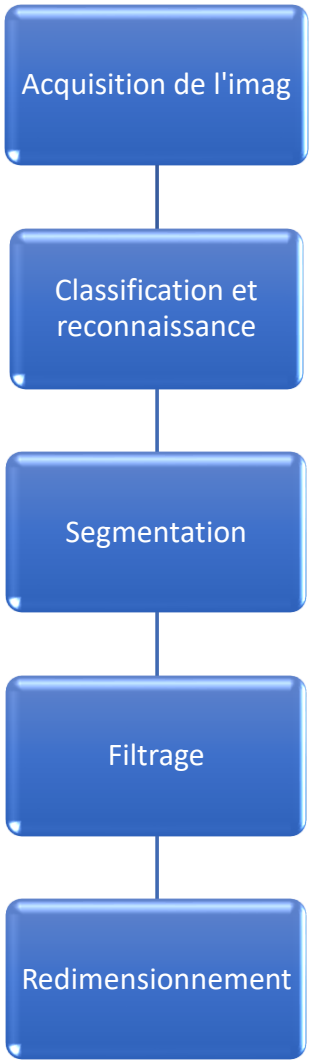

Figure 18 : diagramme des étapes traitement d'image

Dans notre cas le prétraitement a consisté à redimensionner les images.

2.2.5 La structure du réseau SSD

Le réseau SSD (Single Shot MultiBox Detector) est un modèle de réseau de neurones profonds utilisé pour la détection d'objets dans des images. Il se compose de plusieurs couches qui travaillent ensemble pour détecter et localiser des objets de différentes tailles et formes comme le montre la figure 19. Voici une brève explication des différentes couches utilisées dans SSD [22] :

Couche de base (base layer) : Cette couche est généralement un réseau de neurones convolutif pré-entraîné, comme VGG ou ResNet, qui extrait des caractéristiques de bas niveau à partir de l'image d'entrée.

Couches de détection (detection layers) : Ces couches sont ajoutées au-dessus de la couche de base pour détecter des objets à différentes échelles et niveaux de résolution. Elles sont constituées de plusieurs blocs de convolution qui réduisent progressivement la taille spatiale des cartes d'activation.

Couches de prédiction (prediction layers) : Pour chaque couche de détection, il y a des couches de prédiction qui sont utilisées pour prédire les coordonnées des boîtes englobantes et les étiquettes de classe associées à chaque région d'intérêt.

Couches de fusion (fusion layers) : Ces couches combinent les prédictions provenant de différentes échelles et niveaux de résolution pour former une sortie cohérente. Elles aident à gérer les objets de différentes tailles présents dans l'image.

Le nombre de couches dans SSD peut varier en fonction de la configuration spécifique utilisée, mais en général, le modèle comporte plusieurs couches de détection et de prédiction, généralement réparties sur différentes échelles spatiales pour capturer les objets de différentes tailles. Le nombre exact de couches peut être ajusté en fonction des besoins spécifiques de détection d'objets dans un scénario donné.

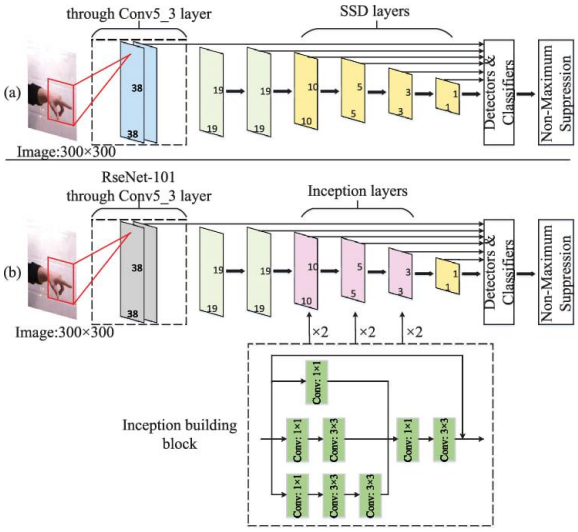

Figure 19 : Architecture en couches du réseau SSD pour la détection d'objets

2.2.6 Fonctionnement du réseau neuronal

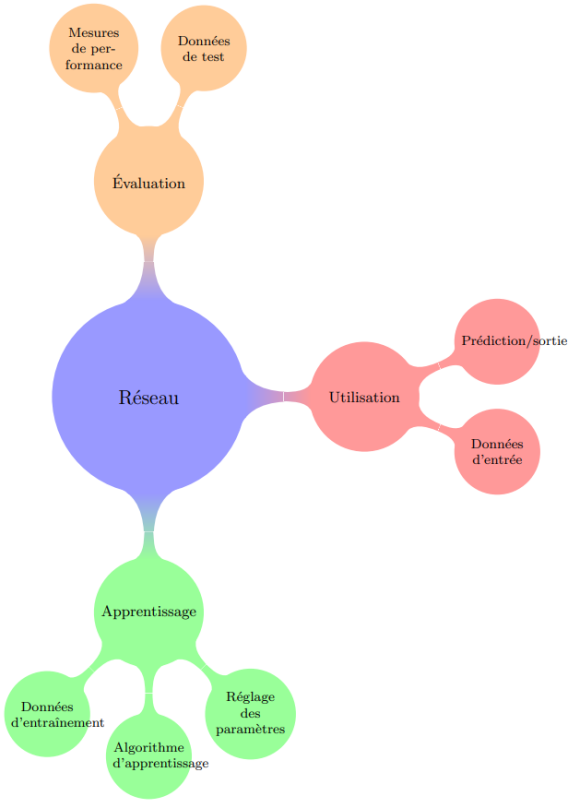

Figure 20 : Schéma des composantes d'un réseau - Apprentissage, Utilisation et Évaluation

Le schéma représente les différentes étapes impliquées dans le fonctionnement d'un réseau comme le montre la figure 20, Voici une explication détaillée de chaque étape [23] :

1. **Apprentissage :**

Données d'entraînement : Cette étape consiste à collecter un ensemble de données spécifiques qui seront utilisées pour entraîner le réseau. Ces données peuvent inclure des exemples étiquetés ou non étiquetés.

Algorithme d'apprentissage : Un algorithme d'apprentissage est utilisé pour ajuster les paramètres du réseau en fonction des données d'entraînement. Il existe différents types d'algorithmes d'apprentissage tels que les réseaux de neurones, les arbres de décision, etc.

Réglage des paramètres : Pendant l'apprentissage, les paramètres du réseau sont ajustés afin d'optimiser sa performance. Cela peut impliquer le choix des poids des connexions entre les neurones ou la sélection de valeurs optimales pour d'autres paramètres spécifiques du modèle.

2. **Utilisation :**

Données d'entrée : Une fois le réseau entraîné, il peut être utilisé pour effectuer des prédictions ou générer des sorties en utilisant de nouvelles données d'entrée. Ces données peuvent être des exemples sur lesquels le réseau n'a pas été entraîné.

Prédiction/sortie : Le réseau utilise les données d'entrée pour générer des prédictions ou des sorties correspondantes. Cela peut inclure des classifications, des estimations, des reconnaissances de motifs, etc.

3. **Évaluation :**

Données de test : Des données de test indépendantes sont utilisées pour évaluer les performances du réseau après son entraînement. Ces données n'ont pas été utilisées lors de l'apprentissage et servent à vérifier l'aptitude du réseau à généraliser et à produire des résultats précis sur de nouvelles données.

Mesures de performance : Différentes mesures sont utilisées pour évaluer la performance du réseau, telles que la précision, le rappel, la précision moyenne, l'erreur quadratique moyenne, etc. Ces mesures permettent de quantifier l'efficacité et la précision du réseau dans la réalisation de la tâche spécifique pour laquelle il a été entraîné.

2.2.7 La répartition des dossiers principaux sur Android Studio

Android Studio est un environnement de développement intégré populaire pour les développeurs d'applications Android. L'un des aspects clés de la gestion des projets Android Studio est la répartition des dossiers, Comme le montre la figure 21.

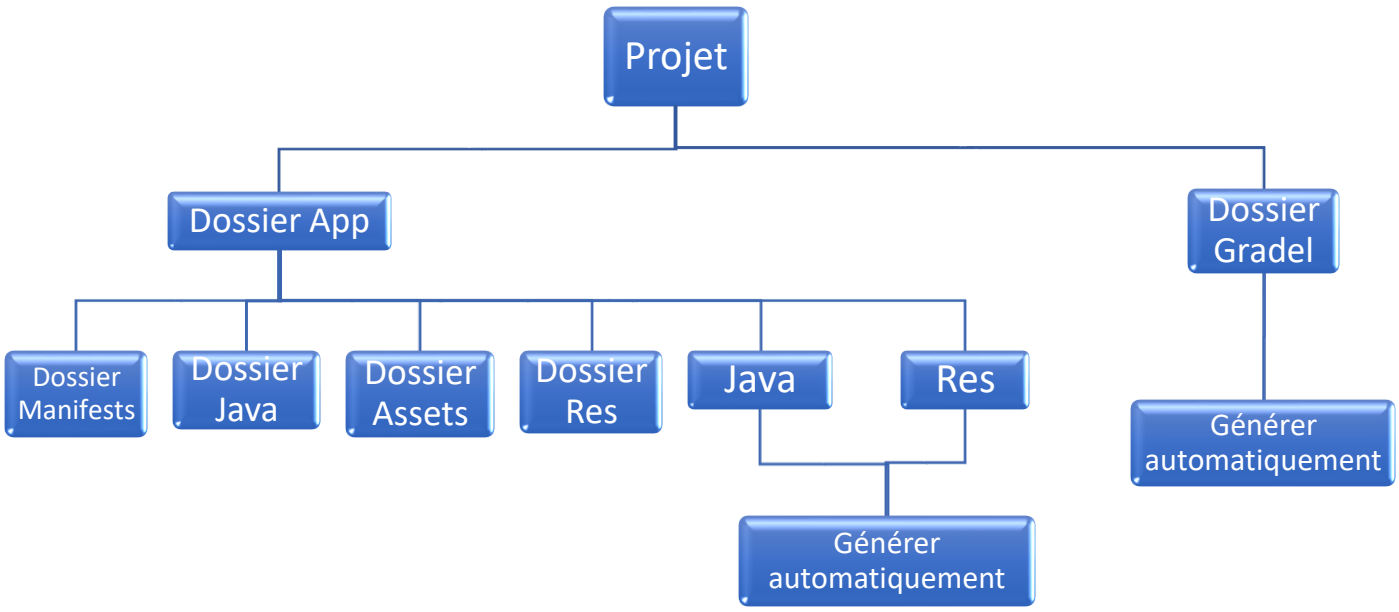

Figure 21 : Diagramme de la répartition du dossier principal sur Android Studio

2.2.8 La création de l'interface sur Android Studio

La création d'interfaces utilisateur est une partie essentielle du développement d'applications mobiles Android. Android Studio offre une variété d'outils pour créer des interfaces conviviales et interactives pour les utilisateurs, La figure 22 montre la création de l'interface sur Android Studio.

Figure 22 : Diagramme de la création de l'interface sur Android Studio

2.2.9 L’implantation de l’icône

L'icône d'une application est une représentation visuelle importante de son identité. Elle permet aux utilisateurs d'identifier facilement l'application et de la distinguer des autres applications installées sur leur appareil. Android Studio offre un processus simple pour implémenter l'icône de l'application, La figure 22 montre l’implantation de l’icône sur Android Studio.

.

Figure 23 : Diagramme de L’implantation de l’icône sur Android Studio

2.2.10 L’implantation de l’audio

L'implémentation de l'audio est une partie importante du développement d'applications mobiles sur Android Studio. Les développeurs peuvent ajouter des fonctionnalités audios à leur application pour améliorer l'expérience utilisateur, tels que des effets sonores, des musiques de fond ou des enregistrements audio. Nous avons créé le fichier son de l'alarme à déclencher en cas de détection de somnolence, comme le montre la figure 24.

Figure 24 : Diagramme de l'implantation de l’audio sur Android Studio

2.2.11 La création du fichier APK

La création d'un fichier APK est une étape cruciale du processus de développement d'applications Android. Un APK (Android Package Kit) est le fichier final qui est installé sur un appareil Android et contient tous les composants nécessaires pour exécuter l'application, comme le montre la figure 25. Android Studio propose un processus simple pour générer un fichier APK.

Figure 25 : Diagramme de La création du fichier APK

2.3 Partie smartphone

Ces étapes fournissent un aperçu général du processus de création d'une application mobile dans Android Studio

2.3.1 L'implantation du fichier APK

L'implantation fichier APK sur un appareil mobile Android peut se faire via une connexion USB ou sans fil. Android Studio propose des fonctionnalités intégrées pour faciliter ce processus, qui peut être représenté par un diagramme en plusieurs étapes comme le montre la figure 26.

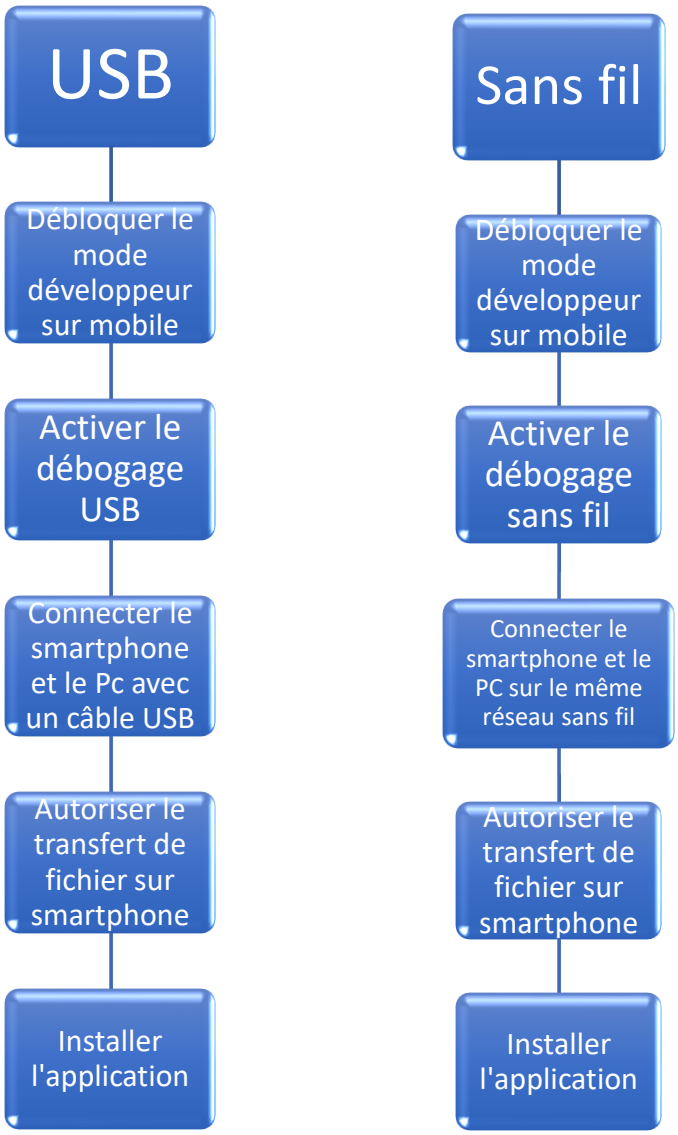

Figure 26 : Diagramme de L'implantation du fichier APK

2.3.2 Les Étapes d'acquisition et d'installation de l'application

Pour acquérir et installer une application sur un appareil Android, il existe plusieurs méthodes comme le montre la figure 27. Les utilisateurs peuvent installer des applications directement depuis leur appareil via le Google Play Store, ou ils peuvent également le faire depuis un PC ou un autre smartphone

Figure 27 : Diagramme pour les Étapes d'installation de l'application

2.3.3 Utilisation de notre application

L'utilisation d'une application peut varier selon son but et sa complexité. Cependant, il est possible de décrire les étapes générales pour utiliser une application mobile en créant un diagramme simple en quelques étapes clés comme le montre la figure 28.

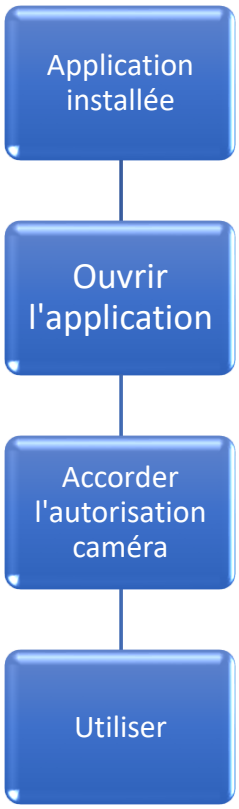

Figure 28 : Diagramme d'utilisation de notre application mobile

2.4 Conclusion

En conclusion, ce chapitre a fourni une vue d'ensemble des aspects pratiques de la création de Notre application mobile. Nous avons examiné la mise en œuvre de la base de données, présenté une architecture générale du système, effectué une démonstration de l'environnement de développement et décrit les outils utilisés. Nous avons également passé en revue les langages de développement et de programmation qui ont été utilisés pour la création de notre application mobile.

Chapitre 3 Implémentation et Résultat

3.1 Introduction

Le but de ce chapitre est de présenter les étapes de mise en œuvre de la méthode proposée pour notre application mobile de détection de la somnolence. Nous commençons par présenter les ressources, le langage et l'environnement de développement que nous utilisons. Ensuite, nous détaillons la phase de mise en place du modèle, et enfin, nous abordons les tests effectués. Ce chapitre est composé de deux parties : la réalisation du système et les résultats des tests expérimentaux.

3.2 Environnement de travail

Les différents outils matériels et logiciels utilisés pour la réalisation de notre application est dans ce qui suit :

3.2.1 Environnement matériel

Notre application est réalisée sur un pc portable avec un smartphone pour l'exécution dont les caractéristiques sont résumées dans le tableau suivant :

	Micro-ordinateur (Développement)	Smartphone (Exécution)
Marque	ACER ASPIRE	ONEPLUS 8T.
Processeur	Intel core i7	Qualcomm Snapdragon 865 Octa-core
Disque dur	HDD de 1 TO	128 Go
RAM	16 Go	8 Go
Système	Windows 10, 64 bits	Android 13

Table 1 : Présentation de l'environnement de travail utilisé

3.2.2 JDK

Le Java Development Kit (JDK) est l'environnement de développement officiel pour la plateforme Java. Il est fourni par Oracle Corporation et contient tous les outils nécessaires pour développer, déboguer et déployer des applications Java. Le JDK inclut notamment le compilateur Java, les bibliothèques Java standard, les outils de développement Java, les fichiers de documentation Java, ainsi que d'autres outils et utilitaires [16] La figure 29 montre logo du jdk.

.

Figure 29 : Logo du jdk

3.2.3 SDK Manager

SDK signifie Software Development Kit, c'est un ensemble d'outils d'aide à la programmation pour concevoir des logiciels comme le montre la figure 30, jeux, applications mobiles, etc. pour un terminal et/ou un système d'exploitation spécifique. Depuis l'arrivée de Android, le système d'exploitation que Google a développé pour se lancer dans le secteur mobile, sont un kit de développement est désormais disponible. Le kit de développement Android inclut les utilitaires nécessaires pour aider les développeurs Android avec les premières étapes : les différents API développés par Google à la fois pour contrôler les fonctions de l'appareil et l'intégration des services, un émulateur complet pour tester des applications, et tout le matériel de lecture nécessaire pour vous aider à effectuer les premières étapes de la programmation pour Android [19].

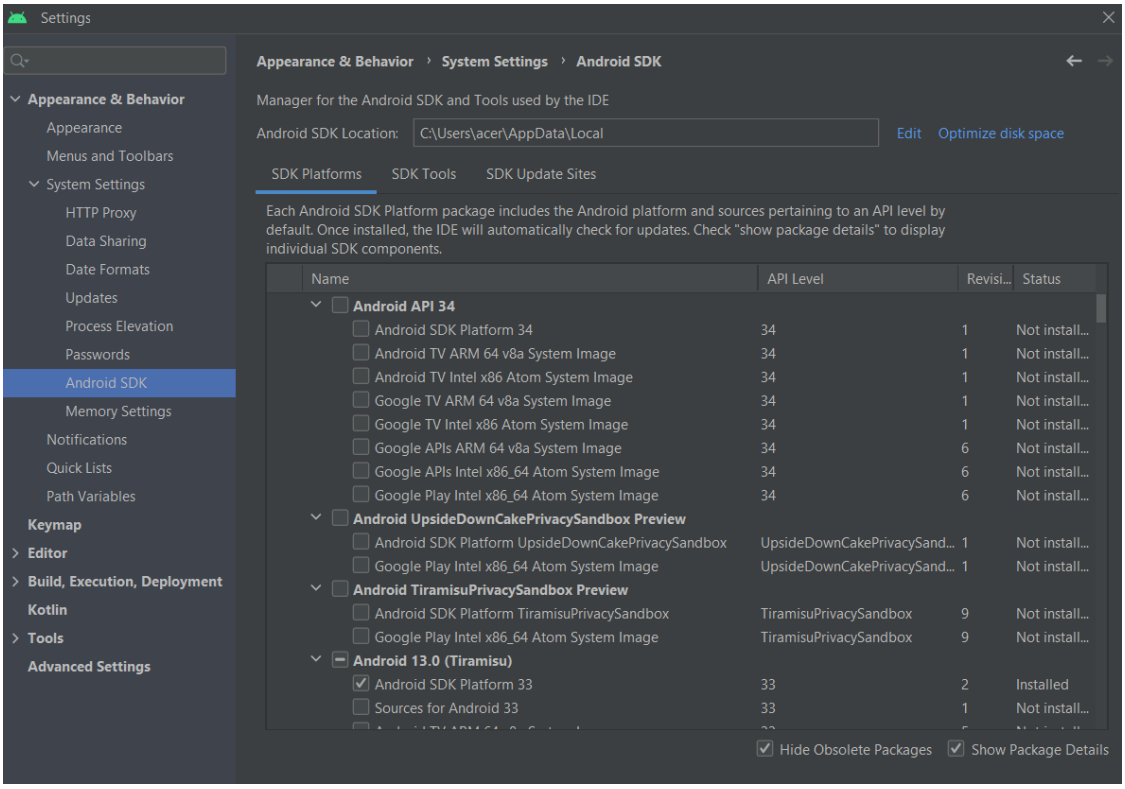

Figure 30 : Android SDK Manager.

3.2.4 Dossier SDK

Le gestionnaire télécharge un fichier d'environ 800 Mo :

Outils SDK : Obligatoire, contient Manager.

Outils de plate-forme SDK : essentiel, inclut ADB.

Plate-forme SDK : Obligatoire, contient des bibliothèques.

Image système : créer AVD.

Prise en charge d'Android : divers outils pour créer des applications.

3.3 Le langage de développement

Le choix du langage de développement est une étape cruciale dans le processus de création d'une application. Le langage choisi détermine non seulement la syntaxe et les fonctionnalités disponibles, mais il influence également la productivité du développeur, les performances de l'application et la compatibilité avec les plateformes cibles.

3.3.1 Java

Java est un langage de programmation gratuit et populaire, en particulier pour
Le développement du système d'exploitation Android. Il a été créé à l'origine par Sun Microsystems, mais appartient actuellement à Oracle. Développé bien après C et C++, Java combine de nombreuses fonctionnalités puissantes de ces langages tout en fixant certains de leurs inconvénients [20].

Figure 31 : Logo du langage de programmation java.

3.3.2 Le langage de balisage XML

Le nom complet de XML est Extensible Markup Language, qui a été défini par le World Wide Web Consortium en 1998.

L'interface utilisateur Android est définie à l'aide du langage XML. Cette approche offre au développeur la possibilité d'écrire son interface, d'utiliser un outil de création d'interface ou de développer lui-même l'interface. Ce format est ouvert compréhensible par l'homme, il est facile à gérer dans les scripts ou les applications, par exemple le remplacement par lots éléments qui existent dans plus d'une interface.

Le langage XML [21] est également utilisé pour définir des valeurs simples telles que des tableaux (arrays.xml), des chaînes (string.xml) et des animations. La figure 15 représente

Figure 32 : logo de xml

3.4 Bibliothèque utilisée

Les bibliothèques Android sont des outils essentiels pour les développeurs d'applications, offrant une multitude de fonctionnalités et de composants pré-construits qui accélèrent le développement, améliorent la qualité et facilitent la mise en œuvre de fonctionnalités avancées. L'exploration et l'utilisation judicieuse de ces bibliothèques peuvent considérablement améliorer notre efficacité et nos performances dans le développement d'applications Android. Toutes les bibliothèques que nous avons utilisées sont présentées dans la figure 33.

```
import android.Manifest;
import android.annotation.SuppressLint;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.os.SystemClock;
import android.view.OrientationEventListener;
import android.view.Surface;
import android.view.View;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.camera.core.AspectRatio;
import androidx.camera.core.CameraSelector;
import androidx.camera.core.ImageAnalysis;
import androidx.camera.core.ImageProxy;
import androidx.camera.core.Preview;
import androidx.camera.lifecycle.ProcessCameraProvider;
import androidx.camera.view.PreviewView;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

import com.google.common.util.concurrent.ListenableFuture;

import org.tensorflow.lite.Interpreter;

import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.nio.MappedByteBuffer;
import java.util.HashMap;
import java.util.Map;
import java.util.Objects;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
```

Figure 33 : Bibliothèques Android utilisées dans le projet

Parmi les bibliothèques importantes pour une application de détection de la somnolence, voici trois exemples :

AndroidX.Camera.Core : Cette bibliothèque fournit des fonctionnalités essentielles pour l'utilisation de la caméra dans votre application. Elle inclut des classes telles que CameraSelector pour sélectionner une caméra, ImageAnalysis pour analyser les images capturées, ImageProxy pour représenter une image capturée, et Preview pour prévisualiser l'image capturée. Ces fonctionnalités pourraient être utiles pour la capture d'images nécessaires à l'analyse la somnolence.

Android.Media.MediaPlayer : Cette bibliothèque permet de lire des fichiers audio et vidéo dans votre application. Si votre application inclut des fonctionnalités audio ou vidéo liées à la détection de la somnolence, cette bibliothèque pourrait être utilisée pour la lecture des fichiers audio liés aux alarmes ou aux stimuli sonores.

Android.View.OrientationEventListener : Cette bibliothèque permet de détecter les changements d'orientation de l'appareil. Dans le contexte de la détection du sommeil, cette fonctionnalité pourrait être utilisée pour ajuster l'interface utilisateur ou le comportement de l'application en fonction de l'orientation de l'appareil.

3.5 L'importance des images de test et d'entraînement

Les images de test et d'entraînement sont des éléments essentiels lorsque nous formons un modèle de détection de la somnolence. Les images de test sont soigneusement sélectionnées pour évaluer les performances de notre modèle une fois qu'il a été entraîné. Elles représentent des situations réelles dans lesquelles la somnolence peut se manifester, et elles nous permettent de mesurer l'exactitude et l'efficacité de notre modèle en termes de détection des signes de somnolence.

D'un autre côté, les images d'entraînement comme le montre la figure 35, sont utilisées pour former notre modèle lui-même. Nous choisissons ces images de manière à représenter un large éventail de situations de somnolence, allant des yeux fermés aux mouvements lents, en passant par d'autres signes distinctifs. En exposant notre modèle à ces différentes situations, nous l'aidons à apprendre à reconnaître les caractéristiques communes associées à la somnolence et à les généraliser pour de nouvelles images.

La création de ces images d'entraînement est cruciale pour nous assurer que notre modèle est capable de détecter la somnolence dans des situations réelles et variées.

Cela améliore sa capacité à généraliser et à prendre des décisions précises dans des scénarios réels. En combinant les images de test et d'entraînement, nous pouvons créer un modèle de détection de la somnolence robuste et fiable, prêt à être utilisé dans des applications de sécurité routière ou d'évaluation de la vigilance humaine.

Name	Size	Packed	Type	Modified	CRC32
..			File folder		
test	12,041,118	12,034,743	File folder	2/19/2023 10:0...	
train	57,809,103	57,775,915	File folder	2/19/2023 10:0...	

Figure 34 : Dossier d'images pour l'entraînement et les tests

Figure 35 : exemples d'images d'éveil et de sommeil

3.6 Annotation des contours des yeux

L'image annotée présente un visage humain avec une mise en évidence des yeux encadrés. L'annotation précise vise à identifier et à délimiter les contours des yeux dans l'image comme le montre la figure 36, permettant ainsi une détection précise de cette caractéristique faciale cruciale. Les yeux sont des éléments clés pour diverses applications de vision par ordinateur, tels que la reconnaissance faciale, le suivi du regard et l'analyse des émotions. Cette annotation aide à fournir une référence visuelle pour l'entraînement des modèles d'apprentissage automatique afin de détecter et de localiser les yeux dans d'autres images. En mettant en évidence les yeux avec des contours précis, cette annotation facilite l'extraction et l'analyse ultérieure des caractéristiques liées aux yeux. En fournissant des données d'entraînement de haute qualité, cette image annotée contribue à améliorer les performances des systèmes de vision par ordinateur dans une variété d'applications pertinentes.

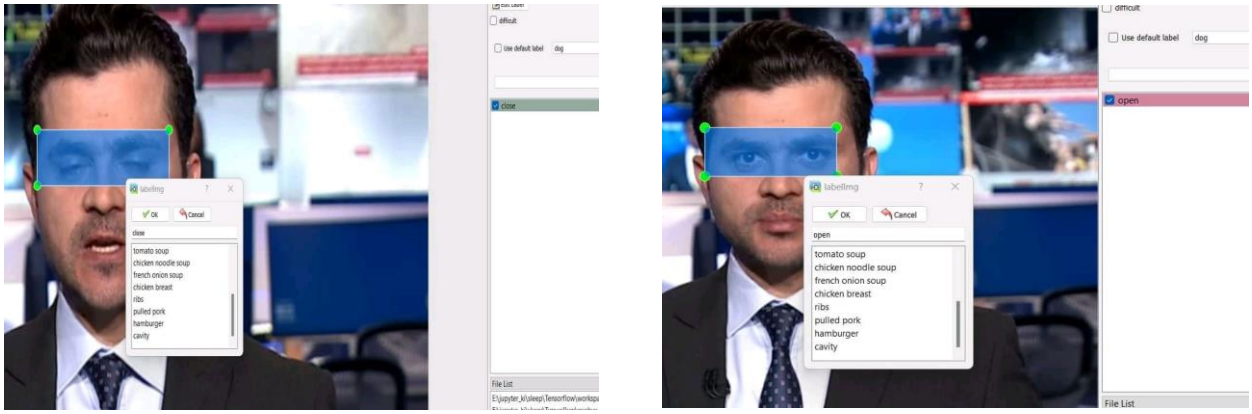

Figure 36 : Exemple d'annotation d'image avec mise en évidence des yeux encadrés

3.7 Annotations d'images pour la détection de la somnolence

Les deux extraits de code que nous avons fournis sont des exemples d'annotations au format XML utilisées pour étiqueter des images dans le contexte de la détection de la somnolence comme le montre la figure 38. Chaque extrait décrit les informations associées à une image spécifique, telles que le chemin d'accès à l'image, les dimensions de l'image, les objets présents dans l'image et leurs coordonnées de délimitation.

Le premier extrait de code concerne une image de test. Il spécifie le dossier dans lequel se trouve l'image, son nom de fichier, son chemin d'accès complet et d'autres informations générales sur l'image, telles que la source, la taille et le fait qu'elle n'est pas segmentée. Il contient également une annotation d'objet qui décrit un objet nommé "open" (ouvert) avec des coordonnées de délimitation (xmin, ymin, xmax, ymax) indiquant sa position dans l'image.

Le deuxième extrait de code concerne une image d'entraînement. Il est structuré de manière similaire au premier extrait, mais il spécifie les informations spécifiques à l'image d'entraînement. Il indique le dossier, le nom de fichier, le chemin d'accès complet, la source, la taille et le fait que l'image n'est pas segmentée. Il contient également une annotation d'objet qui décrit un objet nommé "close" (fermé) avec des coordonnées de délimitation pour indiquer sa position dans l'image.

En résumé, ces deux extraits de code représentent des annotations d'images dans le cadre de la détection de la somnolence. Ils fournissent des informations essentielles pour chaque image, notamment les détails sur l'image elle-même, les objets d'intérêt et leurs positions. Ces annotations sont utilisées pour entraîner des modèles de détection de la somnolence afin de reconnaître les états de somnolence dans les images ultérieures.

```xml
<?xml version="1.0"?>
<annotation>
    <folder>train</folder>
    <filename>Screenshot 2023-02-19 074720.png</filename>
    <path>E:\jupyter_ki\sleep\Tensorflow\workspace\images\train\Screenshot 2023-02-19 074720.png</path>
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>857</width>
        <height>552</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>close</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>414</xmin>
            <ymin>88</ymin>
            <xmax>582</xmax>
            <ymax>173</ymax>
        </bndbox>
    </object>
</annotation>
```

Figure 37 : Annotation d'image de test pour la détection de la somnolence

```xml
<?xml version="1.0"?>
<annotation>
    <folder>test</folder>
    <filename>Screenshot 2023-02-19 092100.png</filename>
    <path>E:\jupyter_ki\sleep\Tensorflow\workspace\images\test\Screenshot 2023-02-19 092100.png</path>
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>862</width>
        <height>532</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>open</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>88</xmin>
            <ymin>124</ymin>
            <xmax>197</xmax>
            <ymax>172</ymax>
        </bndbox>
    </object>
</annotation>
```

Figure 38 : Annotation d'image d'entraînement pour la détection de la somnolence

3.8 Codes utilisés dans notre application Android

L'application Android que nous avons développée repose sur l'utilisation de différents codes pour mettre en œuvre ses fonctionnalités. Ces codes jouent un rôle crucial dans la création et le bon fonctionnement de l'application. Ils permettent d'implémenter des fonctionnalités spécifiques, de gérer les interactions avec l'utilisateur, de traiter les données, et bien plus encore. Dans cette introduction, nous explorerons les différents codes utilisés dans notre application, en mettant en évidence leur importance et leur contribution à l'expérience utilisateur.

3.8.1 Gestion de la permission d'accès à la caméra :

Cette méthode getCameraPermission() comme le montre la figure39. Est utilisée pour vérifier et demander la permission d'accéder à la caméra de l'appareil. Si la permission est déjà accordée, la caméra est démarrée. Sinon, une demande de permission est présentée à l'utilisateur. Cette approche permet de garantir que l'application dispose de la permission nécessaire avant de commencer à utiliser la caméra.

```
private void getCameraPermission() {
    if (ContextCompat.checkSelfPermission(
            context: MainActivity.this, Manifest.permission.CAMERA) ==
        PackageManager.PERMISSION_GRANTED) {
        startCamera();
    } else {
        ActivityCompat.requestPermissions( activity: MainActivity.this,
                new String[]{Manifest.permission.CAMERA},
                HAS_CAMERA_ACCESS);
    }
```

Figure 39 : Code pour implémenter la demande d'autorisation d'accès à la caméra dans l'application Android

3.8.2 Gestion de l'interface utilisateur

Ce code comme le montre la figure 40, configure l'interface utilisateur, initialise la caméra, récupère les références aux éléments d'affichage et met en place un écouteur de clic pour le bouton "start" afin d'appeler la méthode startStop() .

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
    cameraProviderFuture = ProcessCameraProvider.getInstance( context: this);
    startButton = findViewById(R.id.start);
    showOuput = findViewById(R.id.show_state);
    startButton.setOnClickListener((View view) -> {
        startStop(startButton);
    });
```

Figure 40 : Code pour la configuration initiale de l'activité principale dans l'application Android

3.8.3 Gestion de la lecture audio dans MediaPlayer

Ce code comme le montre la figure 41, permet de contrôler la lecture audio dans le contexte spécifique d'une application mobile de détection de la somnolence. Lorsque la détection de la somnolence est activée et que la valeur de "isPlay" est true , le code démarre la lecture d'une alarme. Cette alarme peut être utilisée pour avertir l'utilisateur lorsque des signes de somnolence sont détectés. Si la détection de la somnolence est désactivée ou que la valeur de "isPlay" est false, le code arrête la lecture de l'alarme. Ainsi, ce mécanisme permet d'ajouter une fonctionnalité sonore à l'application de détection de la somnolence pour alerter l'utilisateur.

```
private void setMediaPlayer(boolean isPlay) {
    if (isPlay) {
        if (!mediaPlayer.isPlaying()) {
            mediaPlayer = MediaPlayer.create(getBaseContext(), R.raw.alarm);
            mediaPlayer.start();
        }
    } else {
        if (mediaPlayer.isPlaying()) {
            mediaPlayer.stop();
        }
```

Figure 41 : Code pour la gestion du lecteur multimédia dans l'application Android

3.8.4 Gestion du chargement du modèle

Cette méthode loading() comme le montre la figure 42, est utilisée pour charger un fichier de modèle. Elle appelle une fonction externe (loadModelFile()) pour effectuer cette tâche et gère les exceptions qui pourraient se produire lors du chargement.

```
private void loading() {
    try {
        modelFile = Tools.loadModelFile( context: MainActivity.this);
    } catch (Exception ignored) {
    }
}
```

Figure 42 : Code pour le chargement du fichier modèle dans l'application Android

3.8.5 Préparation des données d'image

Ce code comme le montre la figure 43, prend un objet Bitmap en entrée et convertit ses pixels en un tableau de bytes normalisés, stockés dans un ByteBuffer. Cette conversion est effectuée en itérant à travers les dimensions du bitmap, en extrayant les valeurs des pixels, en les normalisant et en les ajoutant au ByteBuffer. Cette méthode est utilisée pour préparer les données d'entrée avant de les alimenter dans un modèle ou un processus de traitement ultérieur.

```
private void convertBitmapToByteBuffer(Bitmap bitmap) {
    if (imgData == null) {
        return;
    }

    imgData.rewind();
    bitmap.getPixels(intValues, offset: 0, bitmap.getWidth(), x: 0, y: 0, bitmap.getWidth(), bitmap.getHeight());
    long startTime = SystemClock.uptimeMillis();
    int pixel = 0;
    for (int i = 0; i < DIM_IMG_SIZE_X; ++i) {
        for (int j = 0; j < DIM_IMG_SIZE_Y; ++j) {
            final int val = intValues[pixel++];
            imgData.putFloat( v: (((val >> 16) & 0xFF) - IMAGE_MEAN) / IMAGE_STD);
        }
    }
    pixel = 0;
    for (int i = 0; i < DIM_IMG_SIZE_X; ++i) {
        for (int j = 0; j < DIM_IMG_SIZE_Y; ++j) {
            final int val = intValues[pixel++];
            imgData.putFloat( v: (((val >> 8) & 0xFF) - IMAGE_MEAN) / IMAGE_STD);
        }
    }

    pixel = 0;
    for (int i = 0; i < DIM_IMG_SIZE_X; ++i) {
        for (int j = 0; j < DIM_IMG_SIZE_Y; ++j) {
            final int val = intValues[pixel++];
            imgData.putFloat( v: (((val) & 0xFF) - IMAGE_MEAN) / IMAGE_STD);
        }
    }
```

Figure 43 : Code pour la conversion d'un Bitmap en ByteBuffer dans l'application Android

3.9 Codes utilisés pour l'entraînement de modèle d'apprentissage

L'entraînement de modèles constitue une étape fondamentale dans le développement d'applications axées sur l'apprentissage automatique et la vision par ordinateur.

Ces codes offrent des instructions détaillées et des lignes directrices permettant de réaliser ces tâches spécifiques dans le contexte de l'entraînement des modèles. Ils servent de point de départ pratique pour personnaliser et adapter le processus d'entraînement en fonction des besoins particuliers de chaque projet.

3.9.1 Initialisation des importations

Ce code initialise les importations nécessaires pour travailler avec des modèles de détection d'objets comme le montre la figure 44, en utilisant l'API de détection d'objets de TensorFlow. Il importe des modules pour interagir avec le système d'exploitation, télécharger des fichiers, TensorFlow pour l'apprentissage automatique, et des modules spécifiques de l'API de détection d'objets pour la configuration et la construction de modèles de détection d'objets.

```
import os
import wget
```

```
import object_detection
```

```
import tensorflow as tf
from object_detection.utils import config_util
from object_detection.protos import pipeline_pb2
from google.protobuf import text_format
```

Figure 44 : Code pour l'importation des bibliothèques et des modules nécessaires à l'entraînement du modèle

3.9.2 Configuration des paramètres

Ce code définit des variables pour le nom du modèle personnalisé, le nom et l'URL du modèle pré-entraîné, le nom du script pour générer les fichiers TFRecord et le nom du fichier de mappage des étiquettes comme le montre la figure 45, Ces variables sont utilisées pour configurer et préparer l'environnement de détection d'objets avec TensorFlow.

```
CUSTOM_MODEL_NAME = 'my_ssd_mobnet'
PRETRAINED_MODEL_NAME = 'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8'
PRETRAINED_MODEL_URL = 'http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_fpnlite_320x320_coco1
TF_RECORD_SCRIPT_NAME = 'generate_tfrecord.py'
LABEL_MAP_NAME = 'label_map.pbtxt'
```

Figure 45 : Code de définition des constantes pour le modèle pré-entraîné et les fichiers associés

3.9.3 Organisation des chemins de fichiers

Ce code définit un dictionnaire de chemins de fichiers et de répertoires utilisés dans le projet de détection d'objets avec TensorFlow. Ces chemins facilitent la gestion des différents fichiers, répertoires et emplacements nécessaires pour l'entraînement, l'exportation et l'inférence des modèles de détection d'objets comme le montre la figure 46.

```
paths = {
    'WORKSPACE_PATH': os.path.join('Tensorflow', 'workspace'),
    'SCRIPTS_PATH': os.path.join('Tensorflow','scripts'),
    'APIMODEL_PATH': os.path.join('Tensorflow','models'),
    'ANNOTATION_PATH': os.path.join('Tensorflow', 'workspace','annotations'),
    'IMAGE_PATH': os.path.join('Tensorflow', 'workspace','images'),
    'MODEL_PATH': os.path.join('Tensorflow', 'workspace','models'),
    'PRETRAINED_MODEL_PATH': os.path.join('Tensorflow', 'workspace','pre-trained-models'),
    'CHECKPOINT_PATH': os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME),
    'OUTPUT_PATH': os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME, 'export'),
    'TFJS_PATH':os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME, 'tfjsexport'),
    'TFLITE_PATH':os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME, 'tfliteexport'),
    'PROTOC_PATH':os.path.join('Tensorflow','protoc')
 }
```

Figure 46 : Code pour définir les chemins de fichiers et de répertoires dans le projet TensorFlow

3.9.4 Paramètres du modèle SSD

Ce code définit un modèle SSD pour la détection d'objets avec deux classes spécifiques et utilise un redimensionnement fixe des images à une taille de 320x320 pixels avant la détection comme le montre la figure 47.

```
{'model': ssd {
   num_classes: 2
   image_resizer {
     fixed_shape_resizer {
       height: 320
       width: 320
     }
```

Figure 47 : Code pour la configuration du modèle SSD pour la détection d'objets

3.9.5 Conversion d'un modèle TensorFlow

Ce code définit des variables pour les chemins des fichiers de modèle TensorFlow Lite et construit une commande pour convertir un modèle TensorFlow sauvegardé en un modèle TensorFlow Lite en utilisant tflite_convert comme le montre la figure 48. La commande spécifie les chemins d'entrée et de sortie, les dimensions de l'entrée, les noms des tenseurs d'entrée et de sortie, ainsi que d'autres paramètres de conversion. La commande est ensuite imprimée à la sortie.

```
FROZEN_TFLITE_PATH = os.path.join(paths['TFLITE_PATH'], 'saved_model')
TFLITE_MODEL = os.path.join(paths['TFLITE_PATH'], 'saved_model', 'detect.tflite')
```

```
command = "tflite_convert \
--saved_model_dir={} \
--output_file={} \
--input_shapes=1,300,300,3 \
--input_arrays=normalized_input_image_tensor \
--output_arrays='TFLite_Detection_PostProcess','TFLite_Detection_PostProcess:1','TFLite_Detection_PostProcess:2','TFLite_Detectio
--inference_type=FLOAT \
--allow_custom_ops".format(FROZEN_TFLITE_PATH, TFLITE_MODEL, )
print(command)
```

Figure 48 : Code pour la conversion du modèle entraîné en modèle TFLite

3.10 Tests et Résultats

Nous avons présenté les tests effectués pour évaluer l'efficacité et la pertinence du système proposé. Les différents types de tests ont été décrits en détail, tant en évidence les objectifs et les méthodes utilisées pour chaque test. Les résultats obtenus ont été analysés et interprétés, permettant de tirer des conclusions sur les performances du système.

3.10.1 Résultats de l'autorisation d'utilisation de la caméra

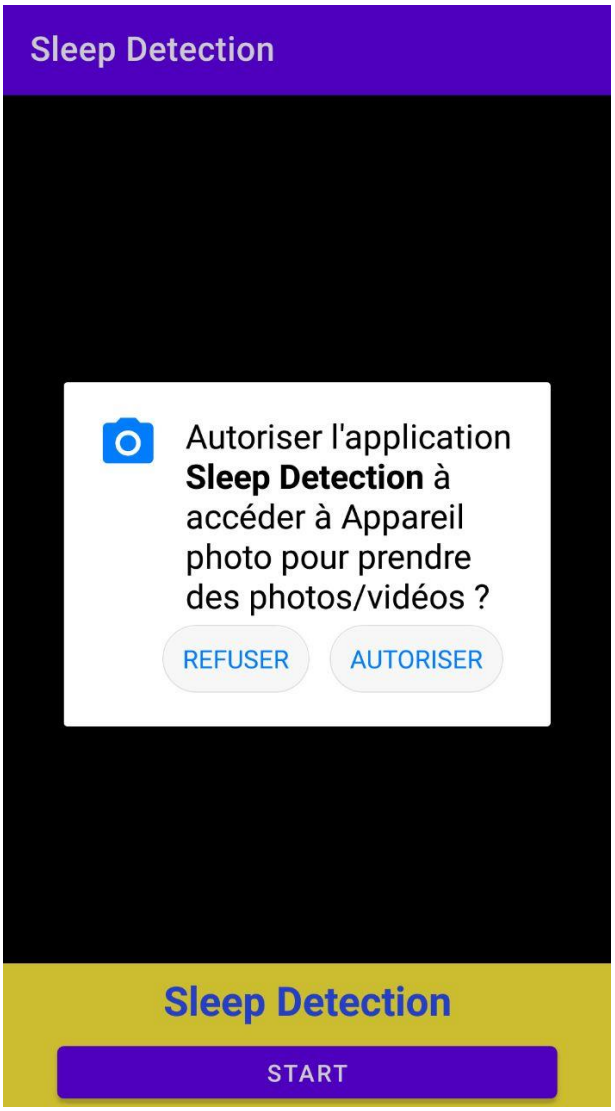

Figure 49 : Capture d'écran de l'autorisation d'utilisation de la caméra

En spécifiant les permissions requises de manière explicite, le système d'exploitation Android peut garantir que notre application ne peut pas accéder à des ressources sensibles sans l'autorisation explicite de l'utilisateur comme le montre la figure 49.

Cela permet de protéger la vie privée et la sécurité des utilisateurs en évitant les abus potentiels d'accès aux données ou aux fonctionnalités de leur appareil.

3.10.2 Résultats de l’interface utilisateur

L'interface de notre application de détection de la somnolence au volant a été conçue de manière intuitive et conviviale comme le montre la figure 50. Elle comprend principalement deux boutons: un bouton "Start" et un bouton "Stop". Ces boutons permettent à l'utilisateur de démarrer et d'arrêter la surveillance de la somnolence.

Lorsque nous lançons l'application et appuyons sur le bouton "Start", celle-ci commence à surveiller les signes de somnolence du conducteur en utilisant caméra frontale.

Pendant la surveillance, notre interface affiche en temps réel le statut du conducteur, indiquant s'il est "éveillé" ou "endormi". Nous utilisons les mots "Awake" (éveillé) ou "Sleep" (endormi) affichés sur l'interface pour permettre à l'utilisateur de visualiser rapidement l'état de vigilance du conducteur.

Lorsque notre application détecte des signes de somnolence et que le conducteur est considéré comme "endormi", nous déclenchons une alerte pour l'avertir de sa condition et l'encourager à prendre des mesures pour éviter les dangers potentiels.

En appuyant sur le bouton "Stop", l'utilisateur peut mettre fin à la surveillance de la somnolence et arrêter alerte. Cela permet de désactiver le mode de détection et de revenir à l'interface principale de l'application.

Notre interface vise à offrir à l'utilisateur une expérience simple et efficace. Les boutons "Start" et "Stop" offrent un contrôle facile du processus de détection de la somnolence, tandis que l'affichage du statut "éveillé" ou "endormi" fournit une indication visuelle claire de l'état de vigilance du conducteur. Cela contribue à améliorer la sécurité au volant en prévenant les situations de conduite dangereuses liées à la somnolence.

Figure 50 : Capture d'écran de l'interface utilisateur de l'application

3.10.3 Résultat de l'exécution sur pc

L'exécution d'un entraînement de modèle SSD sur un ordinateur est cruciale pour évaluer la performance du modèle dans la détection de la somnolence. Les résultats obtenus fournissent des informations précieuses pour évaluer la sensibilité, la spécificité et la performance globale du modèle, ainsi que pour orienter les améliorations nécessaires afin d'obtenir une détection plus précise et fiable de la somnolence.

Ce code affiche des informations essentielles sur l'entraînement d'un modèle de détection de somnolence comme le montre la figure 51, notamment les pertes, le taux d'apprentissage et les statistiques d'étape. Ces informations sont utiles pour évaluer les performances et le progrès de l'entraînement du modèle.

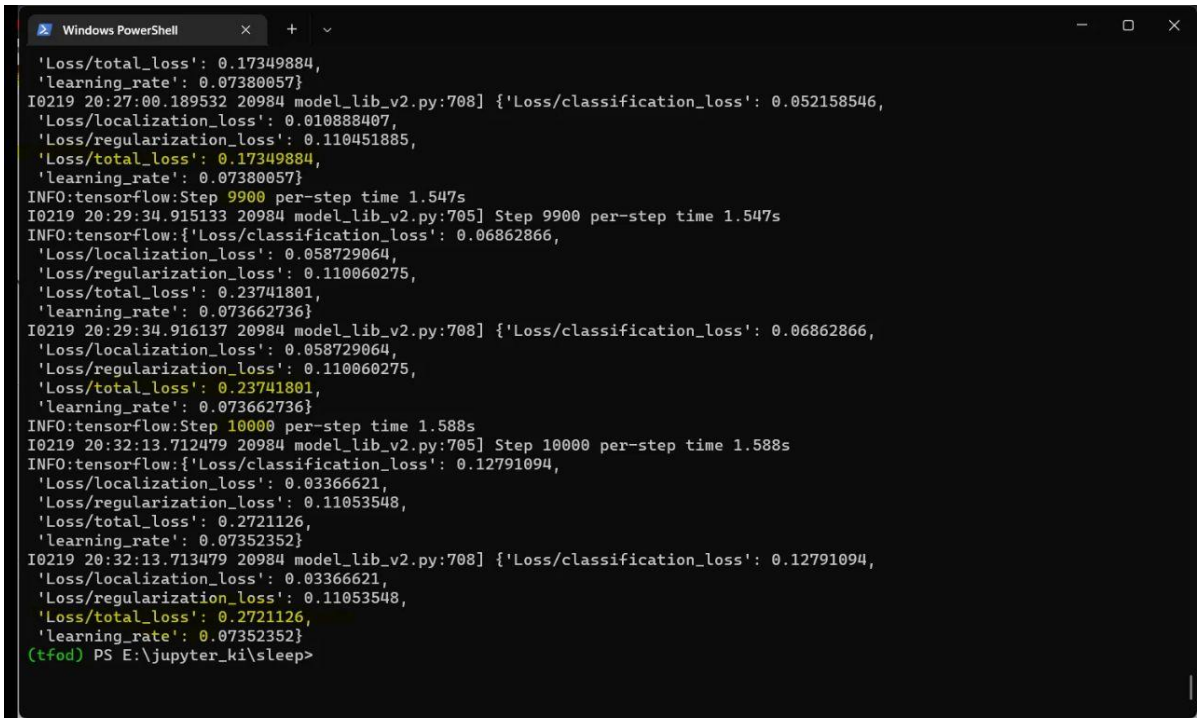

Figure 51 : Résultats d'entraînement du modèle dans le windows powershell

Étapes	Perte de classification	Perte de localisation	Perte de régularisation	Perte totale	Taux d'apprentissage
0	1.0	1.0	1.0	1.0	1.0
100	0.30707	0.310534	0.153638	0.775242	0.03199
200	0.276371	0.141016	0.153528	0.570914	0.037333
4000	0.009127	0.017597	0.135168	0.201892	0.079262
9900	0.052159	0.010888	0.110452	0.173499	0.073801
10000	0.068629	0.058729	0.110064	0.237418	0.073524

Table 2 : Résultats de précision lors du test du modèle d'entraînement

Ce Le tableau illustre les changements des pertes de classification, de localisation, de régularisation et la perte totale, ainsi que du taux d'apprentissage, au cours de l'entraînement du modèle comme le montre la figure 52.

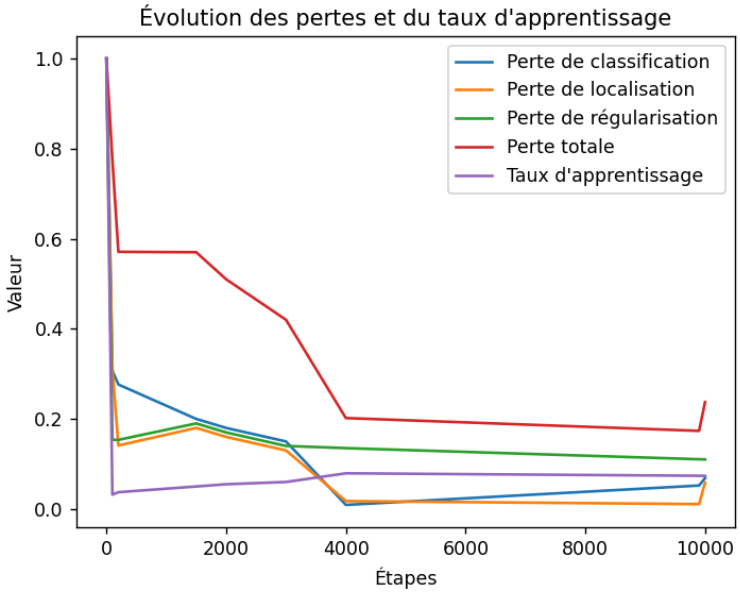

Figure 52 : Courbe de l'évolution des pertes et du taux d'apprentissage

Le graphique représente l'évolution des pertes de classification, de localisation, de régularisation et totale, ainsi que du taux d'apprentissage, en fonction du nombre d'étapes de l'entraînement d'un modèle. Les pertes diminuent progressivement au fil des étapes, ce qui indique une amélioration de la performance du modèle. La perte totale, qui combine les différentes pertes, montre également une tendance à la baisse. Le taux d'apprentissage reste relativement constant au début puis diminue légèrement. Cette visualisation permet d'évaluer l'efficacité de l'entraînement du modèle et de suivre sa convergence vers une meilleure performance.

3.10.4 Résultat de l'exécution sur smartphone

Après l'exécution de l'application sur le smartphone, les résultats obtenus illustrent la performance et la réactivité du logiciel sur l'appareil mobile comme le montre la figure 53.

Figure 53 : Captures d'écran des tests de l'application de détection de somnolence

Les tests de jour ont été effectués sur des personnes entre 23 et 60 de sexe féminin et masculin dans un environnement bien éclairé et dans un véhicule en marche comme le montre le tableau 3 :

Type de test	Nombre de tests	Tests réussis	Fausse alerte	Pas de détection
Visage nu	20	20	0	0
Visage avec lunettes de vue	20	18	2	0
Visage avec lunettes de soleil	20	0	0	20

Table 3 : Résultats des tests de détection de visages pour différentes conditions

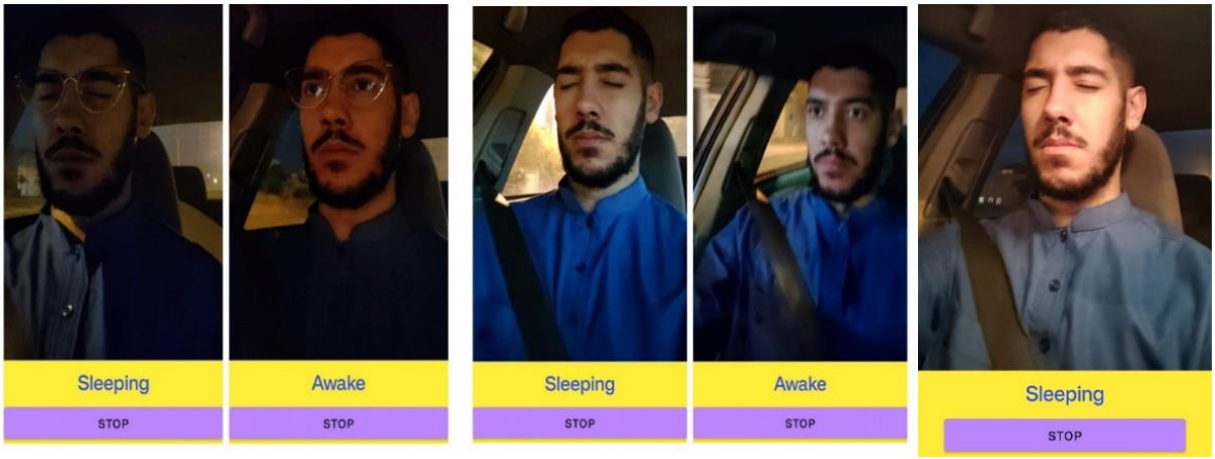

Figure 54 : Captures d'écran des tests de détection de somnolence pendant la nuit dans un environnement sombre

Les tests de nuit ont été effectués sur des personnes entre 23 et 26 dans un environnement sombre et dans un véhicule en marche comme le montre la figure 54 et le tableau 4 :

Type de test	Nombre de tests	Test réussie	Fausse alerte	Pas de détection
Visage nu	10	2	0	8
Visage avec lunettes de vue	10	1	0	9
Analyse du tracé électrocardiograpVisage avec éclairage intérieur du véhicule	10	4	0	6

Table 4 : Résultats des tests de détection de visages pendant la nuit dans un environnement sombre

3.10.5Statistique de réussite finale sur smartphone

Les statistiques de réussite finale sur smartphone mettent en évidence l'efficacité et l'adaptabilité des applications mobiles pour répondre aux besoins des utilisateurs :

- Pour les tests sur les visages nu le taux de réussite est de 73.33%
- Pour les tests sur les visages avec des lunettes de vu le taux de réussite est de 63.33%
- Pour les tests sur les visages avec des lunettes de soleil le taux de réussite est de 0

3.11 Discussion

- Tout d'abord on remarque deux choses essentielles au bon fonctionnement de l'application : l'éclairage et l'emplacement de la caméra c'est-à-dire celui du smartphone.

Plus l'endroit est lumineux et meilleur est l'angle de vue de la caméra, mieux seront le résultat et sa fiabilité

-On remarque que les performances d'un smartphone jouent aussi sur la rapidité de la détection plus un smartphone est puissant plus la détection se fera rapidement et plus précisément

- Les résultats des tests de jour dans un environnement bien éclairé et dans un véhicule en marche sont très satisfaisants on remarque juste que des fois quand on porte des lunettes de vu le reflet fausse un peu la détection

- les résultats des tests de nuit dans un environnement sombre et dans un véhicule en marche sont assez décevants plus il fait sombre plus la détection est mauvaise, mais des fois les phares des voitures qui viennent ou l'éclairage intérieur du véhicule permettent d'avoir une meilleure détection

3.12 Conclusion

Dans ce chapitre, nous avons présenté l'implémentation du système proposé et les résultats obtenus suite à son application. Les différentes étapes d'implémentation ont été décrites, mettant en évidence les choix techniques et les défis rencontrés. Les résultats obtenus durant la journée sont très satisfaisants, démontrant l'efficacité et la pertinence de l'approche adoptée. Cependant, pendant la nuit, où le problème de somnolence est plus présent, les résultats sont en dessous de la moyenne, soulignant la nécessité d'améliorations pour optimiser davantage le système.

Conclusion générale

En conclusion, ce projet a exploré le développement d'une application mobile innovante pour la détection de la somnolence, en utilisant le langage de programmation Java et l'intelligence artificielle. Tout au long du projet, nous avons mis en évidence l'importance de la collaboration, de la communication et de l'apprentissage continu pour surmonter les défis et créer une solution efficace et conviviale

L'application développée a démontré sa capacité à analyser les données en temps réel et à fournir des alertes précises et opportunes aux utilisateurs, et pourrait ainsi contribuer ainsi à la prévention des accidents liés à la somnolence. L'embarquement de l'intelligence artificielle a permis à l'application de s'adapter aux comportements spécifiques de chaque utilisateur, améliorant ainsi la précision et l'efficacité de la détection de la somnolence.

Notre projet a également souligné l'importance de l'attention portée aux détails et de la qualité dans le développement de logiciels. Les tests rigoureux et les processus d'assurance qualité ont été essentiels pour garantir que l'application fonctionne correctement et répond aux besoins des utilisateurs

Enfin, nous avons discuté des améliorations potentielles et des développements futurs qui pourraient être apportés à l'application pour la rendre encore plus efficace et utile. Il est important de continuer à rechercher et à intégrer les dernières avancées technologiques et les meilleures pratiques pour maintenir l'application à jour et répondre aux besoins changeants des utilisateurs.

En somme, ce mémoire vise à démontrer comment la combinaison du langage Java et de l'intelligence artificielle peut être utilisée pour créer une application mobile innovante et efficace dans la détection de la somnolence, contribuant ainsi à améliorer la sécurité et le bien-être des utilisateurs sur les routes.

Bibliographie

[1] Site officiel d'Android Developers : https://developer.android.com/ 2023/5/22

[2] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. In European conference on computer vision (pp. 21-37). Springer, Cham.

[3] https://fr.wikipedia.org/wiki/Expression_faciale.

[4] DANALET, Antonin. Modèles de choix discrets pour la reconnaissance des expressions faciales statiques. 2007.

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

[6] https://upload.wikimedia.org/wikipedia/commons/thumb/e/e6/Circadian_rhythm.svg/ 1280px-Circadian_rhythm.sv 2023/02/1

[7] Halawa, O. I., Hassanein, A. H., Elmogy, M., & Ibrahim, A. (2018). An overview of electrocardiogram-based techniques for the detection of driver drowsiness. IEEE Access, 6, 24451-24466.

[8] Chen, S., Wang, Y., Wang, C., & Wang, M. (2020). A novel ECG-based driver fatigue detection algorithm using time-frequency feature and deep neural network. Sensors, 20(6), 1607.

[9] Benedetto, S., Esposito, A., & Bifulco, P. (2019). A novel approach for drowsiness detection in drivers using machine learning techniques and vehicle sensors. IEEE Sensors Journal, 19(16), 6889-6896.

[10] Fuentes-Pacheco, J., Ramírez-Cortés, J. M., Galván-Tejada, C. E., & EscobedoAguirre, S. (2020). Driver drowsiness detection using pressure sensors installed in the driver's seat. IEEE Access, 8, 144251-144261.

[11] K. S. Gopinath and S. K. Mitra, "Introduction to Digital Signal and System Analysis," John Wiley & Sons, 2010.

[12] "A Survey on Gabor Filters for Image Analysis" par P. Kaur et al., International

[13] V. Kazemi and J. Sullivan, "One Millisecond Face Alignment with an Ensemble of Regression Trees," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1867-1874, 2014.Un facteur très important lors de la détection des régions, la taille moyenne

[14] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., & Berg, A.C. (2016). SSD: Single Shot MultiBox Detector. In European Conference on Computer Vision (pp. 21-37). Springer.

[15] Site officiel d'Android Studio : https://developer.android.com/studio .2023/03/1

[16] Documentation d'Android Studio : https://developer.android.com/docs/ . 2023/03/1

[17] Expected accuracy of the pet example using SSD model in TensorFlow object detection API? - Stack Overflow 2023/03/1

[18] AndroWiiid, Frédéric Espiau et DakuTenshi, « Créez des applications pour Android », le site du zero, Le 10 mai 2012.

[19] Gestionnaire de SDK — RAD Studio (embarcadero.com) 2023/03/1

[20] programmation.developpez.com/actu/291204/Quels-sont-les-meilleurs-langages-deprogrammation-pour-developper-une-application-mobile-Petit-tour-d-horizon-sur-les-plus-populaires

[21] https://developer.mozilla.org/fr/docs/Web/XML/XML_introduction 2023/03/4

[22] Liu, W., & Li, F. F. (2016). Ssd: Single shot multibox detector. In Computer Vision – ECCV 2016 (pp. 21-37). Springer, Cham. Lien vers l'article : https://link.springer.com/chapter/10.1007/978-3-319-46448-0_2

[23] "Deep Learning" (Livre) ,Auteurs : Ian Goodfellow, Yoshua Bengio, Aaron Courville, Éditeur : MIT Press ,Année de publication : 2016