République Algérienne Démocratique Et Populaire
Ministère De l'Enseignement Supérieure Et De La Recherche Scientifique
Université Abderahmane Mira - Bejaia -
Faculté Des Sciences Exactes
Département d'informatique

Mémoire de fin d'études en vue de l'obtention d'un diplôme
Option : Génie logiciel

thème

Développement d'un système de détection du comportement des conducteurs de véhicules en utilisant le deep learning

Réalisé par :

Mr BEKKA Reda **Melle KHERBOUCHE Samia**

Soutenu le 07 Septembre 2020 devant le jury composé de :

Présidente	**Mme BOUKERRAM Samira née AIT KACI AZZOU**	**M.A.A**
Examinateur	**Mr AMROUN Kamal**	**M.C.A**
Encadreur	**Mme El BOUHISSI Houda Épouse BRAHAMI**	**M.C.A**

promotion 2019/2020

- Remerciement -

Au terme de notre travail, nous remercions dieu le plus puissant de nous avoir donné le courage et la patience pour réaliser ce travail.

La réalisation de ce mémoire a été un parcours jalonné de nombreuses rencontres,sans lesquelles ce travail n'aurait pas pu aboutir.Nous n'aurions pas éprouvé autant de plaisir à réaliser ce travail sans ces personnes, qui par leur générosité,disponibilité et leur bonne humeur et l'intérêt manifesté à l'égard de notre recherche, ont grandement contribué à l'amélioration de notre travail.

Nous tenons particulièrement à adresser nos remerciements d'abord à notre promotrice : **Dr El BOUHISSI Houda**, pour nous avoir orienté durant l'élaboration de ce travail.

Nous tenons aussi à remercier chacun des membres des jurys pour nous avoir fait l'honneur d'examiner et d'évaluer notre travail.

Nos reconnaissances et nos estimes sont également portées à l'attention du personnel d'**ImagineParteners**, plus exactement la filiale spécialisé en intelligence artificielle situé à Tunis.

Enfin, nos remerciements s'adressent à tous les enseignants du département informatique de l'Université Abderrahmane MIRA, et à toutes les personnes qui ont contribué de prés ou de loin à la réalisation de ce travail.

- Dedicaces -

Nous dédions ce modeste travail particulièrement à nos chers parents, qui ont consacré leur existence à bâtir la notre, pour leur soutien, patience et soucis de tendresse et d'affection pour tout ce qu'ils ont fait pour que nous puissions arriver a ce stade,que dieu les bénisse et protège.

À nos chères frères et soeurs.

À toutes nos familles.

À nos meilleurs amis.

Table des matières

Table des figures

Liste des tableaux

Liste des abriviations

IA	**I**ntelligence **A**rtificielle
CNPSR	**C**entre **N**ational de **P**révention et de **S**écurité **R**outière
CNN	**C**onvolutional **N**eural **N**etwork
VGG	**V**isual **G**eometry **G**roup
ANN	**A**rtificial **N**eural **N**etwork
IJCV	**I**nternational **J**ournal **C**omputer **V**ision
HOG	**H**istogram of **O**iented **G**radients
YOLO	**Y**ou **O**nly **L**ook **O**nce
FPS	**F**rames **P**er **S**econd
SSD	**S**ingle **S**hot **D**etector
ILSVRC	**I**mageNet **L**arge **S**cale **V**isual **R**ecognition **C**hallenge
SIFT	**S**cale **I**nvariant **F**eature **T**ransform
WRF	**W**eighted **R**andom **F**ores
FER	**F**acial **E**motion **R**ecognition
CK	**C**ohn **K**anade
MMI	**M**an **M**achine **I**nterface
HRNN	**H**ierarchical **R**ecurrent **N**eural **N**etwork
ResNet	**R**esidential **E**nergy **S**ervices **N**etwork
ReLU	**R**ectified **L**inear **U**nit
FC	**F**ully **C**onnected
LRN	**L**ocal **R**esponse **N**ormalization
ROC	**R**eceiver **O**perating **C**haracteristic
PIL	**P**ython **I**maging **L**ibrary
GPIO	**G**eneral **P**urpose **I**nput **O**utput

introduction générale

Les accidents de la route demeurent un véritable fléau au niveau mondial. Chaque année, plus de 1,35 million de personnes perdent la vie dans des accidents de la route. Les chercheurs recensent également 20 à 50 millions de blessés, dont nombre d'entre eux gardent une invalidité à la suite de leurs blessures [1].
L'Algérie occupe actuellement la 98ème place dans le classement mondial des accidents de la circulation routière et la 42ème position au niveau africain. 918 personnes ont perdu la vie et 11.919 autres ont été blessées en Algérie durant le premier trimestre de l'année en cours [2].
L'un des facteurs les plus importants des accidents de la route est la fatigue, la somnolence ainsi que la distraction au volant qui est un événement qui se produit lorsqu'un conducteur détourne son attention principale de la conduite vers une autre tâche ou activité, cette dernière réduit les perceptions du conducteur et sa capacité de prise de décision pour contrôler le véhicule.
Avec notre dépendance accrue à la technologie et aux téléphones portables, la distraction au volant devient une préoccupation croissante. Ainsi, pour prendre des contres mesures efficaces afin d'améliorer la sécurité des conducteurs, nous nous sommes focalisé sur la détection de mouvement en temps réel en utilisant l'intelligence artificielle ou IA, qui croise plusieurs techniques simulant les processus cognitifs humains, existant depuis les années 60, la recherche s'est développée récemment au point de multiplier les applications : voitures autonomes, diagnostics médicaux, assistants personnels, finance algorithmique, robots industriels, jeux vidéo...etc.
Dans notre projet nous allons utiliser l'une des meilleures technologies de l'IA qui est le deep learning, littéralement appelé apprentissage profond, Cette technologie s'inspirant du cerveau humain offre des algorithmes très utiles pour faire face à l'apprentissage à partir de grande quantité de données, les modèles de Deep Learning peuvent atteindre un niveau de précision exceptionnel, parfois supérieur aux performances humaines, en outre, Un des avantages majeurs des réseaux de Deep Learning réside dans leur capacité à continuer à s'améliorer en même temps que le volume de vos données augmente, cependant l'usage des algorithmes conçu afin de réaliser la détection de mouvement des conducteurs reste une tâche délicate et cela est dû à la nouveauté du domaine.
Vu que les interfaces de programmation d'application détectent uniquement la fatigue et quelques expressions faciales tel que la joie, tristesse...etc., elles peuvent pas détecter si le conducteur est distrait ou non et cette distraction n'implique pas seulement l'utilisation de téléphones portables, mais comprend également le fait de manger, de boire et même de parler à un passager pendant la conduite...etc.
L'objectif principal des recherches sur la détection des distractions du conducteur était de proposer

des prototypes qui capturent, suivent et stockent le comportement du conducteur en temps réel, en prédisant automatiquement si les conducteurs sont distraits et quelle action en particulier entraîne la distraction au volant.
Dans ce contexte, et au cours de ce mémoire, nous allons passer à travers différentes étapes pour réaliser le projet. Ainsi, le reste du mémoire est organisé comme suit :
Dans le premier chapitre «présentation du projet», nous présenterons l'organisme d'accueil et notre projet où nous spécifions notre problématique et les solutions que nous avons proposées.
Ensuite, dans le second chapitre «notions fondamentales», nous allons exposer des notions de bases de notre thème tel que l'intelligence artificielle, les algorithmes de détection faciale, objets ainsi la détection de mouvement sur laquelle nous allons focaliser tout au long de notre projet.
Notre état de l'art est présenté dans le troisième chapitre, où nous allons décrire les travaux relatifs et faire une analyse et comparaison entre ces travaux.
Dans le quatrième chapitre «approche proposée», nous mettons en œuvre notre conception du système. Cette partie est accompagnée d'une évaluation permettant de vérifier l'efficacité et la pertinence de notre contribution.
Dans le chapitre 5 «expérimentation», nous allons parler sur implémentations et tests dont nous présentons les langages et les outils que nous avons utilisé dans cette partie et les interfaces de notre logiciel.
Enfin, nous terminons par une conclusion et les travaux futurs.

Chapitre 1

Présentation du projet

1.1 Introduction

Ces dernières années, l'évolution de la technologie a pris un essor important dans le domaine d'industrie automobile, sans prendre en compte la sécurité des usagers.
Tous les usagers empruntant les routes peuvent être victimes des différents dangers inhérents à la conduite. Cependant suite à des comportements inadaptés ils peuvent être victimes ou responsables des accidents. C'est pour cette raison que différents organismes, officiels comme associatifs, réalisent à travers le territoire national des actions de sensibilisation à la prévention routière, afin d'assurer la sécurité des différents usagers. Malgré les moyens mis par l'état ainsi que les associations, le nombre de décès ne cessent d'augmenter. Afin de contribuer à la réduction de nombre d'accidents et de décès, nous avons opté à l'utilisation de l'intelligence artificielle qui couvre tout le spectre des activités humaines, telles que l'environnement et l'énergie, la santé et l'assistance a l'autonomie à domicile, le transport et les villes intelligentes ... etc.
Ce chapitre présentera d'une part l'organisme d'accueil où se déroule notre stage, d'autre part nous aborderons la problématique qui nous a poussée à réaliser ce projet ainsi que la solution proposée et l'utilité du produit à réaliser.

1.2 Présentation de l'entreprise Imagine Partners

Notre projet de fin d'étude intitulé « Développement d'un système de détection du comportement des conducteurs de véhicules de transports publics en utilisant l'IA » s'est déroulé au sein de la filiale tunisienne de l'entreprise « Imagine Partners ». C'est une société de presque 20 ans d'expérience dans l'accompagnement opérationnel fondée dans le cadre de l'essaimage de France Telecom/Orange en 2000 (dispositif d'accompagnement pour la création d'entreprise), née d'une volonté de rassembler des experts de différents métiers afin de proposer à ses clients une approche opérationnelle et personnalisée des problématiques que ce dernier peut rencontrer (organisation, processus, outils, méthodologie... etc.). Spécialisés dans le domaine des services, principalement dans les Nouvelles Technologies de l'Information et de Communication.
Les domaines d'intervention « d'ImaginePartners » sont présentés dans la figure 1.1 :

FIGURE 1.1 – **Domaine d'intervention de la société.**

La société s'attend à assurer que les problématiques clients soient résolus. Pour ce faire, elle accompagne le client avec un vrai transfert de savoir-faire. Le schéma de la figure 1.2 présente les axes importants que la société développe avec ses clients.

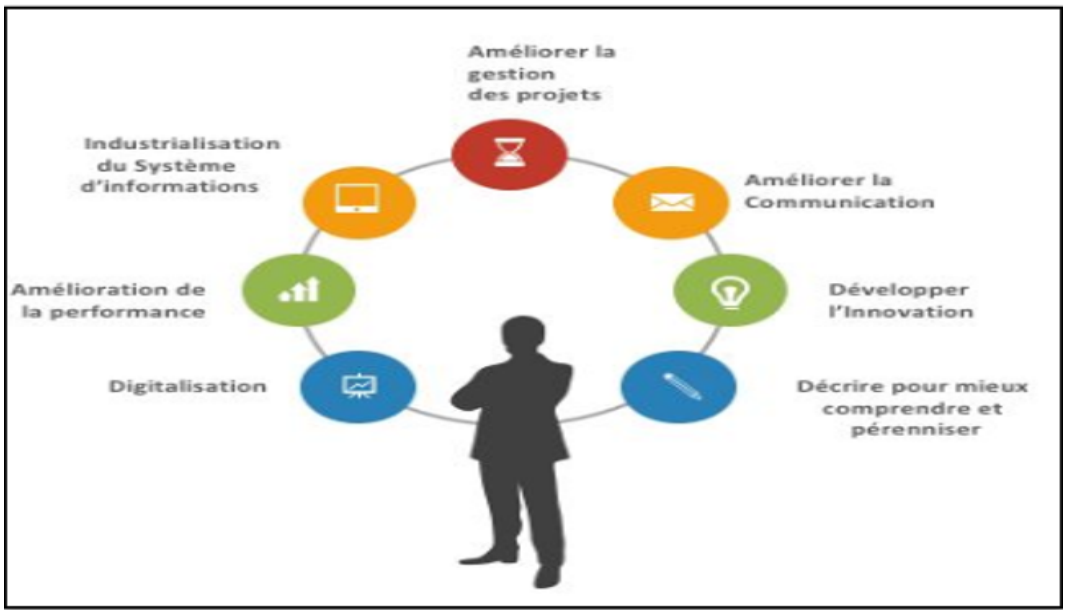


FIGURE 1.2 – **Axe de développement avec les clients.**

1.3 Problématique

L'Algérie mène depuis quelques décennies une lutte contre les accidents et l'insécurité régnante sur la route. Au cours des dernières années des lois plus sévères et des mesures vigoureuses ont été implantées afin d'améliorer le bilan routier.
Cependant, le nombre de décès et d'accidents sur les routes ne cessent d'augmenter, selon le Centre national de Prévention et de Sécurité routière (CNPSR), l'Algérie a enregistré 1.647 décès sur les routes durant le premier semestre de l'année 2019, soit une hausse de 9,7% par rapport à la même période de 2018. Pour le début de l'année 2020, nous constatons que la courbe est toujours ascendante. La question qui se pose vise à déceler les raisons de cette augmentation de décès dans les routes en Algérie et définir, entre autre, les mesures capables de stabiliser et réduire par la suite le nombre de décès dans ce vaste pays.
Le facteur humain est à l'origine de 90% des accidents de la route (figure 1.2), dû principalement à l'excès de vitesse (17,31%), et les nombreuses distractions qui peuvent altérer la conduite entre autre manque de concentration dans les zones urbaines avec un taux de 15,32% d'accidents, parler au téléphone, somnolence ... etc.[3].
A cet effet, notre problématique consiste à apporter une solution qui peut aider à l'atténuation du nombre important d'accidents par conséquent de décès.

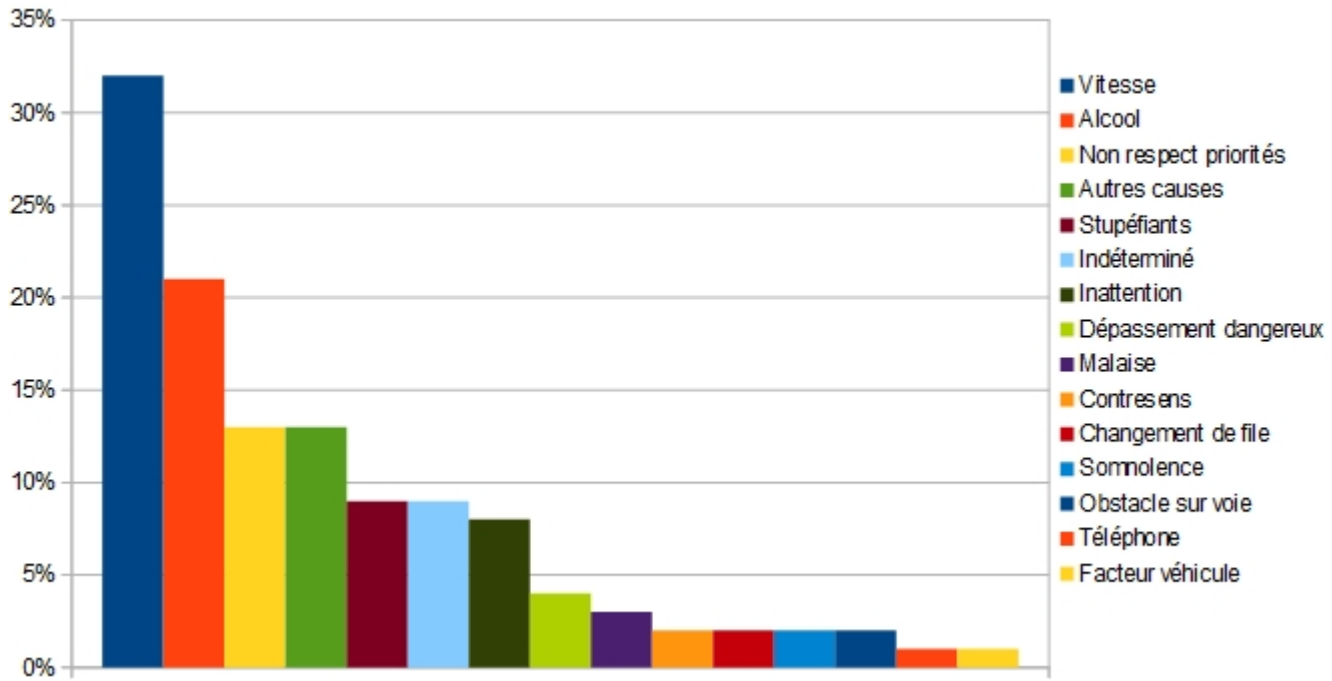


FIGURE 1.3 – **Les causes des accidents de la route [4]**

.

1.4 Solution

Pour réduire cette hécatombe et résoudre certains problèmes liés au trafic routier, nous avons pensé à mettre en place une kyrielle de techniques qui seront implémentées dans un logiciel qui aura comme particularité de capter tous les mouvements du conducteur en utilisant une technologie robuste qui est le deep learning avec ces divers algorithmes qui permettent de faire de la détection de mouvements en temps réel.

1.5 Présentation du projet

Notre projet consiste en l'étude d'un système de détection du comportement des conducteurs de véhicules de transports publics en utilisant l'apprentissage profond qui est l'une des principales technologies de l'intelligence artificielle et qui est dérive de l'apprentissage automatique. Ce dernier donne les résultats les plus optimaux et surtout en ce qui concerne la détection de mouvements en temps réel.
Pour cela, nous avons procédé à la réalisation d'un logiciel qui capte les distractions et mouvements de conducteur en utilisant les algorithmes du deep learning connu sous le nom de l'apprentissage en profondeur.
Le deep learning est un élément essentiel de l'intelligence artificielle qui a déjà contribué a beaucoup de nouveautés dans le domaine de l'industrie automobile comme l'évolution de la sécurité routière, surveillance de l'environnement du véhicule . . . etc.
Nous avons choisi d'utiliser les algorithmes de l'apprentissage profond qui reposent essentiellement sur ce qu'on appelle les réseaux de neurones artificiels permettant de s'en passer d'un expert humain pour faire le tri des données, comme ils permettent de traiter un tas illimité de données en un laps de temps très court.
Notre projet va contribuer non seulement à la diminution du nombre d'accidents et de décès mais aussi à réduire la facture économique et à minimiser les dégâts environnementaux.

1.6 Conclusion

Cette partie nous a permis dans un premier lieu, de citer les problèmes liés à la sécurité routière ainsi qu'à améliorer nos objectifs afin de remédier à ces problèmes.
En deuxième lieu, nous avons proposés nos solutions dédiées à la conception de prochains chapitres.

Chapitre 2

Notions fondamentales

2.1 Introduction

Nous allons présenter dans ce chapitre quelques notions de base de l'intelligence artificielle qui est un sujet de débat passionné pour de nombreux professionnels, elle permet de tirer des conclusions raisonnées, qui peuvent dépasser les capacités de l'esprit humain, à un coût nettement inférieur et avec une rapidité, ensuite nous allons parler sur l'apprentissage automatique (machine Learning) et l'apprentissage profond (deep learning) qui sont les meilleurs technologies de l'intelligence artificielle . Enfin, nous allons passer en revue quelques exemples d'algorithmes de détection faciale d'objets et aussi de mouvements.

2.2 L'Intelligence artificielle

2.2.1 Définition

L'intelligence artificielle, souvent abrégée avec le sigle IA, est définie par l'un de ses créateurs, Marvin Lee Minsky, comme : " La construction de programmes informatiques qui s'adonnent à des tâches qui sont pour l'instant , accomplies de façon plus satisfaisante par des êtres humains car elles demandent des processus mentaux de haut niveau tels que : l'apprentissage perceptuel , l'organisation de la mémoire et le raisonnement critiquée [5].
Donc nous pouvons dire que l'IA permet de réaliser des machines capables de simuler l'intelligence et remplacer l'homme dans certaines mises en œuvre de ses fonctions cognitives.

2.3 Domaine d'applications

Les réalisations actuelles de l'IA peuvent être regroupées en différents domaines, tels que :

— Les systèmes experts,
— L'apprentissage automatique,

— Le traitement automatique des langues,

— La reconnaissance des formes, des visages,

— La reconnaissance des lettres manuscrites,

— Analyse de marché et exploration de données.

2.4 Machine learning

Le Machine Learning consiste à laisser l'ordinateur apprendre quel calcul effectuer, plutôt que de lui donner ce calcul, c'est-à-dire le programmer de façon explicite (Arthur Samuel,1959).
Par la suite un autre chercheur américain, a proposé une définition un peu plus moderne du Machine Learning en énonçant : « une machine apprend quand sa performance à faire une certaine tâche s'améliore avec de nouvelles expérience » (Tom Mitchell,1998).
Pour donner à un ordinateur la capacité d'apprendre, on utilise des méthodes d'apprentissage qui sont fortement inspirées de la façon dont nous, les êtres humains, apprenons à faire des choses [6] . Parmi ces méthodes, on compte :

— **L'apprentissage supervisé** qui analyse les données d'apprentissage et produit une fonction inférée qui peut être utiliser pour mapper de nouveaux exemples,

— **L'apprentissage non supervisé** analyse les données disponible et recherche des modèles et des tendances, il est utilisé pour regrouper des entrées similaires dans des groupes logiques,

— **L'apprentissage par renforcement** laisse l'algorithme apprendre de ses propres erreurs. Afin d'apprendre à prendre les bonnes décisions, l'intelligence artificielle se retrouve directement confrontée à des choix. Si elle se trompe, elle est pénalisée. Au contraire, si elle prend la bonne décision, elle est récompensée. Afin d'obtenir toujours plus de récompenses, l'IA va donc faire de son mieux pour optimiser sa prise de décisions.

2.5 Deep learning

Le deep learning ou apprentissage profond est un type d'intelligence artificielle dérivé du machine learning (figure 2.1) où la machine est capable d'apprendre par elle-même, contrairement à la programmation où elle se contente d'exécuter à la lettre des règles prédéterminées [7].

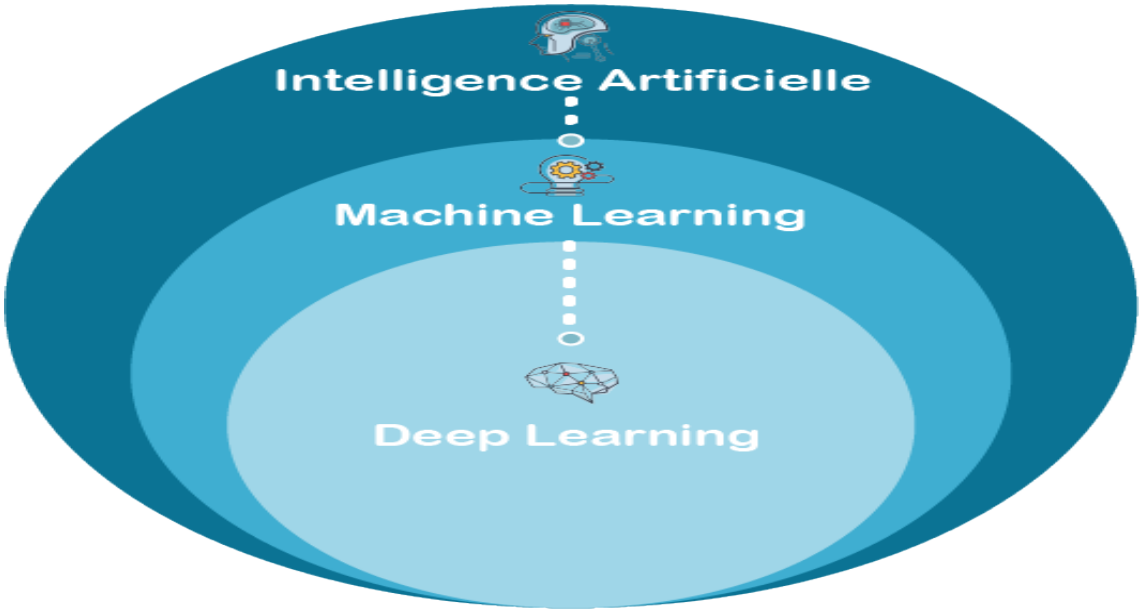


FIGURE 2.1 – **La relation entre l'intelligence artificielle,le machine learning et le deep learning [8].**

2.5.1 Fonctionnement du deep learning

Le deep Learning s'appuie sur un réseau de neurones artificiels s'inspirant du cerveau humain. Ce réseau est composé de dizaines de centaines de «couches» de neurones (figure 2.2), chacune recevant et interprétant les informations de la couche précédente. Le système apprendra par exemple à reconnaître les lettres avant de s'attaquer aux mots dans un texte, ou détermine s'il y a un visage sur une photo avant de découvrir de quelle personne il s'agit. À chaque étape, les «mauvaises» réponses sont éliminées et renvoyées vers les niveaux en amont pour ajuster le modèle mathématique. Au fur et à mesure, le programme réorganise les informations en blocs plus complexes [9].

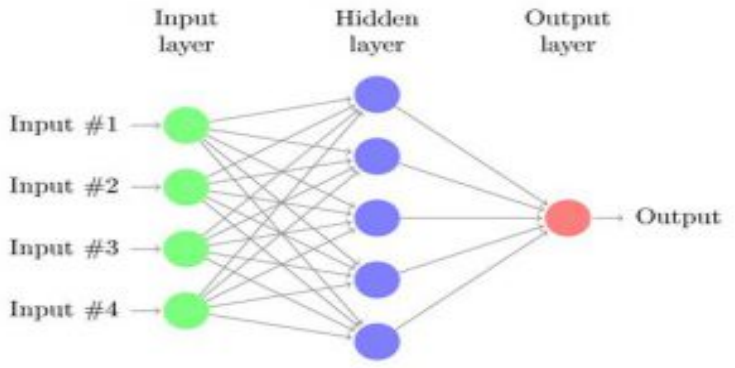


FIGURE 2.2 – **Fonctionnement du deep learning** [10].

2.5.2 Applications du deep learning

Le deep Learning est utilisé dans de nombreux domaines [11] :

— Reconnaissance d'image,

— Traduction automatique,

— Voiture autonome,

— Diagnostic médical,

— Recommandations personnalisées,

— Modération automatique des réseaux sociaux,

— Prédiction financière et trading automatisé,

— Identification de pièces défectueuses,

— Détection de malwares ou de fraudes,

— Chatbots (agents conversationnels),

— Exploration spatiale,

— Robots intelligents.

2.6 Deep learning vs machine learning

— Les modèles de deep learning nécessitent de grandes données,

— Les modèles deep learning nécessite la GPU pour s'entraîner correctement tandis que les modèles d'apprentissage profond s'entraîne sur la CPU,

— Les modèles de deep learning offre une meilleure précision que les modèles élaborés en utilisant le machine learning,

— Le temps d'exécution des modèles d'apprentissage est largement inférieur comparé a ceux de l'apprentissage profond [12].

2.7 Détection faciale

La détection de visage est un domaine de la vision par ordinateur consistant à détecter un visage humain dans une image numérique(figure 2.3). C'est un cas spécifique de détection d'objet, où l'on cherche à détecter la présence et la localisation précise d'un ou plusieurs visages dans une image. C'est l'un des domaines de la vision par ordinateur parmi les plus étudiés [13].

2.7.1 Algorithmes de détection faciale

2.7.1.1 Viola-Jones

La méthode Viola et Jones a été publiée par Paul Viola et Michael Jones dans le journal scientifique International Journal of Computer Vision (IJCV) en 2001 (Viola Jones, 2001) et

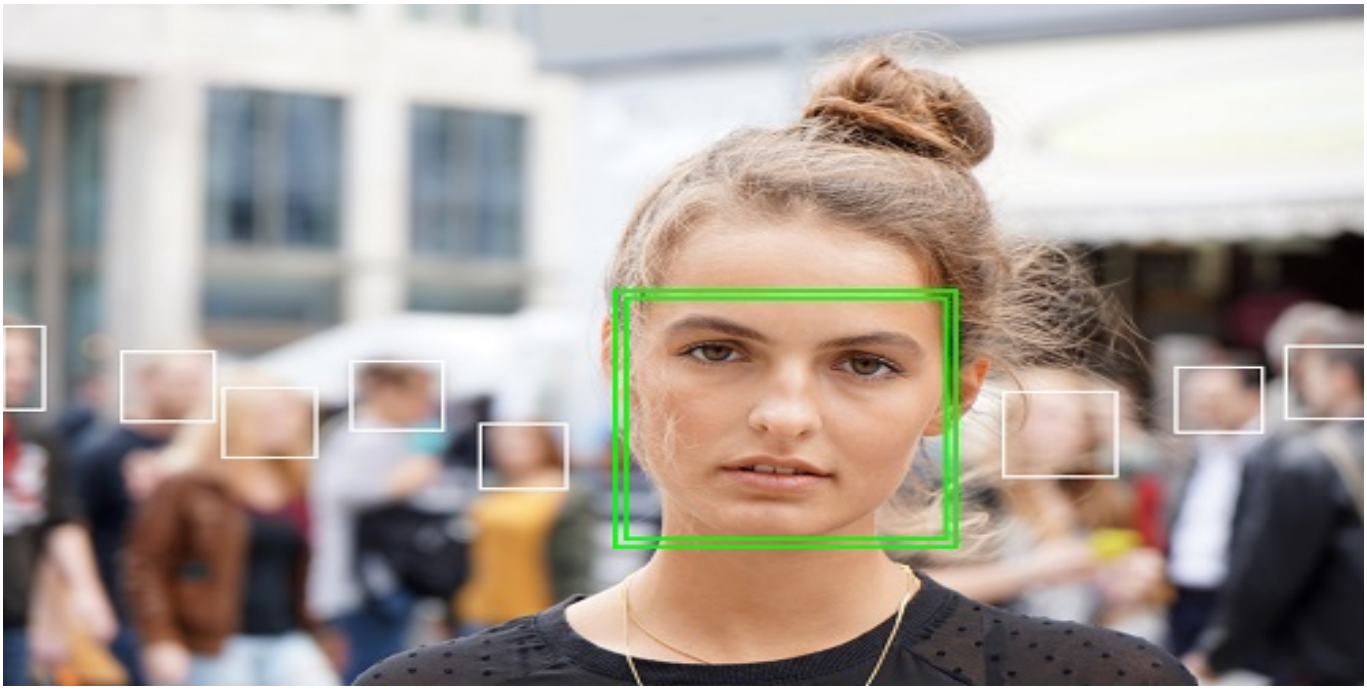


FIGURE 2.3 – **Exemple de détection de visages [14].**

permet de classifier les images. Cette méthode utilise une série de classificateurs faibles pour créer un classificateur fort(figure 2.4). La sortie de l'algorithme est une combinaison pondérée des prédictions faites par chaque classificateur faible [15].
Le Viola - Jones supporte trois (3) techniques pour la détection des parties du visage :

- Les traits de type Haar pour l'extraction de caractéristiques sont de type rectangulaire et sont déterminés par une image intégrale qui est un tableau contenant les sommes des intensités des pixels les valeurs situées directement à gauche d'un pixel et directement au-dessus du pixel à l'emplacement (x, y) inclusive.
- Adaboost est une méthode d'apprentissage automatique pour détecter le visage, il permet de minimiser le nombre de caractéristiques à vérifier.
- Le classificateur en cascade permet de combiner efficacement de nombreuses caractéristiques. Le terme "en cascade" dans un classificateur détermine les différents filtres d'un classificateur résultant [16].

Viola et Jones ont introduit une méthode permettant de détecter avec précision et rapidité les visages dans une image. Cette technique peut être adaptée pour détecter avec précision les traits du visage. Cependant, la zone de l'image analysée pour un trait facial doit être régionalisée à l'endroit où la probabilité de contenir le trait est la plus élevée. En régionalisant la zone de détection, les faux positifs sont éliminés et la vitesse de détection est augmentée grâce à la réduction de la zone examinée [18].

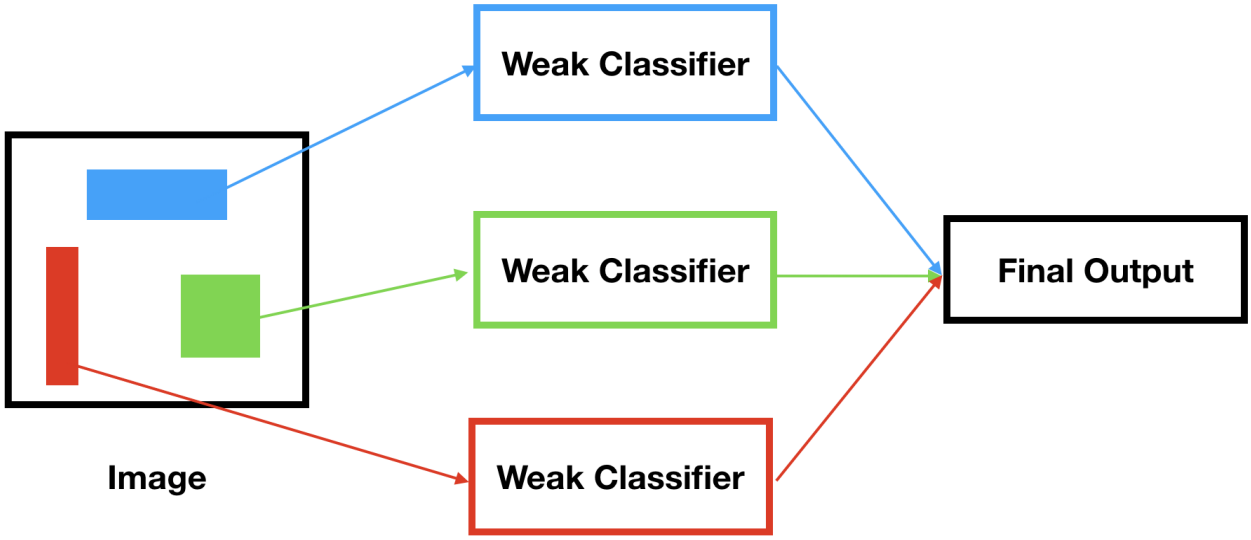


FIGURE 2.4 – **Fonctionnement de l'algorithme de Viola-Jones [17].**

2.7.1.2 Histogram of Oriented Gradients

L' histogramme des gradients orientés (HOG) est un descripteur de caractéristique utilisé dans la vision par ordinateur et un traitement d'image à des fins de détection d'objet En général, il divise l'image en petites cellules et compare les pixels de cette zone les uns aux autres et essaie de mesurer la variation de l'obscurité, puis de trouver la direction où se produit le plus grand changement. Cela montre le mouvement de l'éclairage à cet endroit précis.
Si ce processus est répété pour chaque pixel de l'image, l'image se transforme en une carte des transitions entre les zones claires et les zones sombres. Ces lignes sont appelées des gradients. Chaque gradient montre comment l'image passe d'une zone claire à une zone sombre en ce point. Pour détecter le visage, nous n'avons besoin que des structures globales. L'image sera donc encore simplifiée en la revoyant avec un bloc plus grand cette fois-ci et nous compterons combien de gradients pointent dans chaque direction principale.
Au lieu de garder une trace de tous les gradients séparés dans ce bloc, nous allons simplement stocker un compte du nombre de gradients pointant dans chaque direction. Et la direction qui a le plus de comptes est le facteur le plus fort qui représente cette zone de l'image.
Il y a aussi des gradients pointant dans d'autres directions que nous suivrons. Nous représenterons ici ces autres directions sous forme de lignes moins grasses.
On peut répéter cette opération pour l'ensemble de l'image.
L'image originale est maintenant une simple représentation qui capture la structure de base. Nous pouvons utiliser cette représentation simplifiée pour entraîner facilement un modèle de détection des visages.

Après avoir converti les images en représentations HOG, nous allons commencer à former une machine apprenant le modèle de détection des visages en lui donnant de nombreux exemples de représentations HOG des visages afin qu'elle puisse apprendre à quoi ressemble ce modèle .
Le HOG simplifie l'image de manière à conserver les informations clés nécessaires pour repérer les visages pour permettre au modèle d'apprentissage machine de le résoudre plus facilement. En plus, le HOG présente également d'autres avantages intéressants qui le rendent plus efficace pour les petits ensembles d'apprentissage :
Premièrement, la représentation d'une image par le HOG ne change pas, même si vous éclaircissez ou assombrissez l'image. Comme le HOG ne recherche que les changements de luminosité et non la luminosité absolue, rendre une image un peu plus claire ou un peu plus sombre ne change pas du tout la représentation HOG.
Deuxièmement, la représentation HOG d'une image ne change pas, même si vous modifiez un peu les formes de l'image. Parce qu'il ne s'agit que de changements d'intention importants sur de grandes surfaces de l'image, les petits changements de forme n'ont pas d'importance. C'est très utile pour la détection des visages, car cela signifie que deux visages qui ne se ressemblent pas exactement auront toujours la même représentation HOG [19].

2.8 Détection d'objets

La détection d'objet fait référence à la capacité des systèmes informatiques et logiciels à localiser les objets dans une image/scène et à identifier chaque objet.
La détection d'objets a été largement utilisée pour la détection des visages, la détection des véhicules, le comptage des piétons, les images web, les systèmes de sécurité et les voitures sans conducteur [20].

2.8.1 Algorithme de détection d'objets

2.8.1.1 Algorithme de You Only Look Once(YOLO)

YOLO est un système de détection d'objets ciblé pour un traitement en temps réel. Il a été développé pour créer un processus en une étape impliquant la détection et la classification.
Les prédictions de limites et de classes sont faites après une évaluation de l'image d'entrée [21] Cet algorithme repose sur deux étapes qui sont appliquées sur des images de taille prédéfinie lors de l'apprentissage :

— une détection d'objets opérée par des réseaux de neurones de convolution.

— un quadrillage de l'image où l'on prédit la classe de l'objet s'il existe.

Il existe différents types de YOLO :

— **Tiny Yolo v1** : il est composé de 9 couches de convolution et de 3 couches entièrement connectées. Chaque couche de convolution se compose d'opérations de convolution, de relu qui fuient et de regroupement maximal. Les 9 premières couches de convolution peuvent

être comprises comme l'extracteur de caractéristiques, tandis que les trois dernières couches entièrement connectées peuvent être comprises comme la «tête de régression» (figure 2.5) qui prédit les boîtes englobantes [22].La figure ci-dessous montre la détection d'objets en utilisant l'algorithme de YOLO V1.

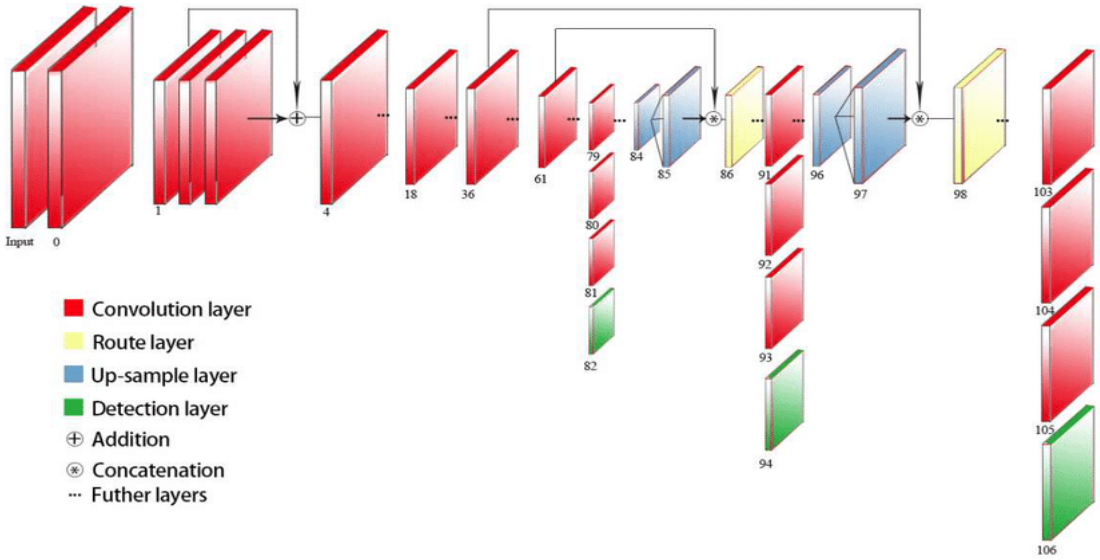


FIGURE 2.5 – **Architecture de Tiny Yolo**
[22].

— **Yolo v1** : Il utilise le Framework Darknet qui est formé sur le jeu de données ImageNet-1000 qui est une base de données de photographies annotées destinées à la recherche en vision par ordinateur, les ensembles de données comprenaient environ 1 million d'images et 1 000 classes d'objets.
La figure ci-dessous montre la détection d'objets en utilisant l'algorithme de YOLO v1.
L'utilisation de YOLO v1 est limitée [23]. Il ne peut pas trouver de petits objets s'ils apparaissent comme un cluster. Cette architecture a rencontré des difficultés dans la généralisation des objets si l'image a d'autres dimensions différentes de l'image entraînée.

— **Yolo v2** : Yolo v2 ou bien Yolo9000 est un système de pointe en temps réel et de détection d'objets pouvant détecter plus de 9000 objets.
La figure ci-dessous montre la detection d'objets en utilisant l'algorithme de YOLO v2.

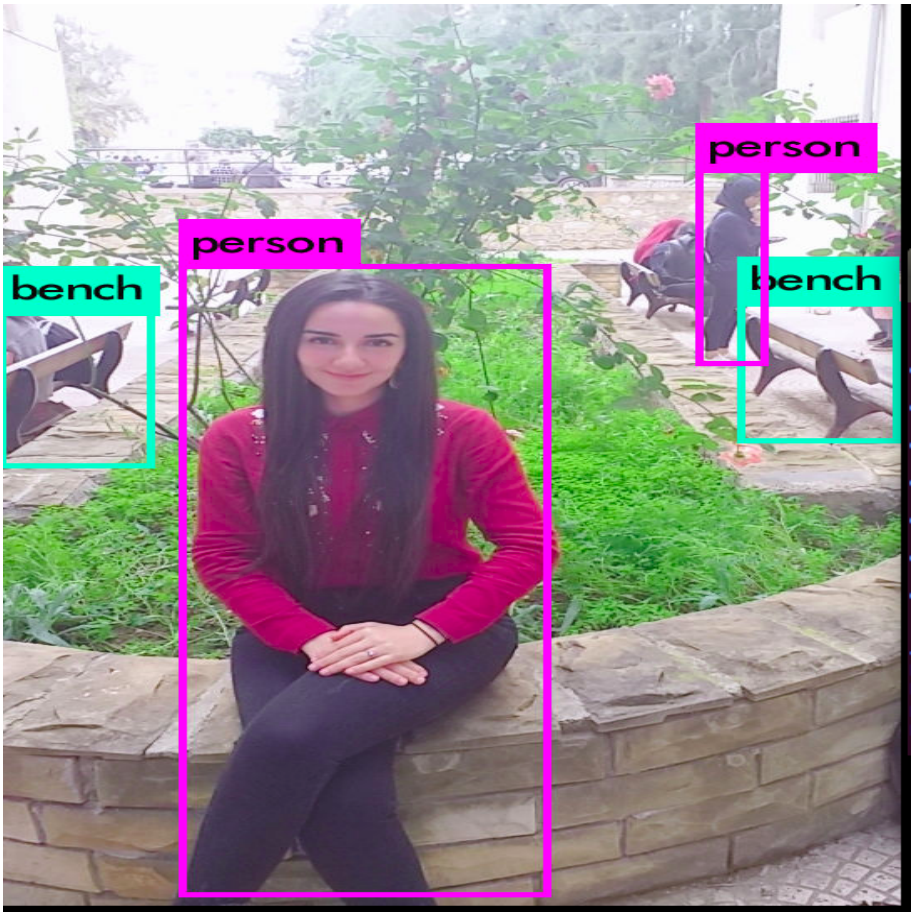


FIGURE 2.6 – **Détection d'objets avec yolo v1**

.

Les améliorations majeures de cette version sont meilleures, plus rapides et plus avancées pour répondre au Faster R-CNN qui est l'une des célèbres architectures de détection d'objets qui utilise des réseaux neuronaux à convolution comme YOLO (You Look Only Once) et SSD (Single Shot Detector) [24].

— **Yolo v3** :Yolo v2 a été amélioré pour une amélioration incrémentielle qui nommée YOLO v3. Comme de nombreux algorithmes de détection d'objets existent depuis un certain temps, maintenant, la concurrence porte sur la précision et la rapidité de détection des objets. YOLO v3 a tout ce dont nous avons besoin pour la détection d'objets en temps réel avec précision et classification des objets. Les auteurs ont appelé cela une amélioration progressive par rapport aux algorithmes précédents [24].

— **Yolo v4** :Le Yolov4 publié par Alexey Bochkovskiy , s'agit d'un modèle de détection d'objets efficace et puissant qui permet à toute personne possédant un GPU 1080 Ti ou 2080 Ti de former un détecteur d'objets ultra rapide et précis [18].
Dans les expériences , Yolov4 a atteint une vitesse en temps réel de 65 FPS(Frames Per Second) sur la Tesla V100, battant les détecteurs les plus rapides et les plus précis en termes de vitesse et précision.

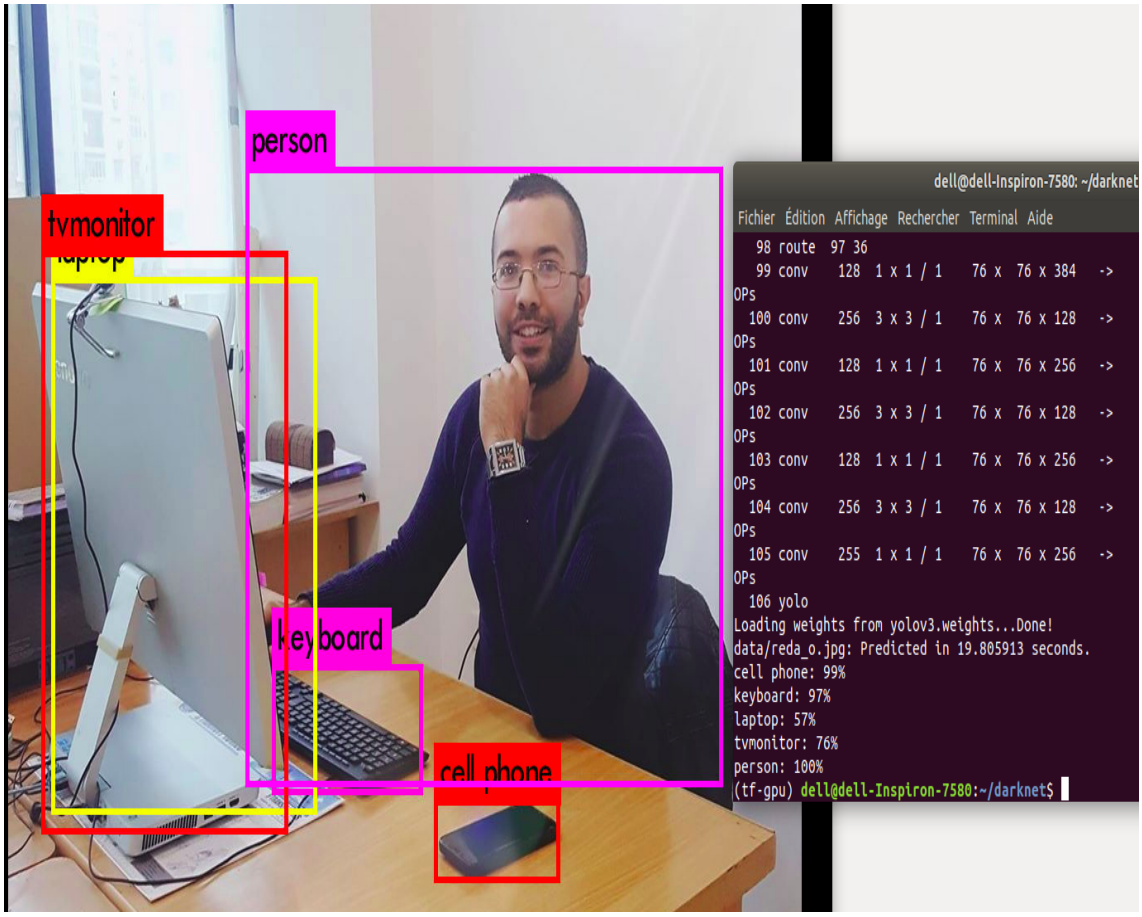


FIGURE 2.7 – **Détection d'objets avec yolo v2**

.

2.9 Détection de mouvements

La détection de mouvement est le processus de détection d'un changement de la position d'un objet par rapport à son environnement ou d'un changement de l'environnement par rapport à un objet [25].
La détection de mouvement concerne plusieurs domaines d'application, nous citons quelques-uns :

— Système de surveillance,

— Caméras de sécurité,

— Contrôle d'éclairage automatisé,

— Reconnaissance de gestes au laser,

— Détecter les comètes et ou les astéroïdes,

— Détecter tout type de mouvement et prendre des photos avec l'appareil photo.

2.9.1 Algorithme VGG16

VGG16 est un modèle de réseau de neurones convolutionnel (figure 2.8) proposé par K. Simonyan et A. Zisserman de l'Université d'Oxford dans l'article «Réseaux de convolution très profonds pour la reconnaissance d'images à grande échelle» [26].
Ce modèle a été utilisé pour remporter le concours ImageNet Large Scale Visual Recognition Challenge (ILSVRC) en 2014, ce défi évalue des algorithmes de détection d'objets et de classification d'images à grande échelle.

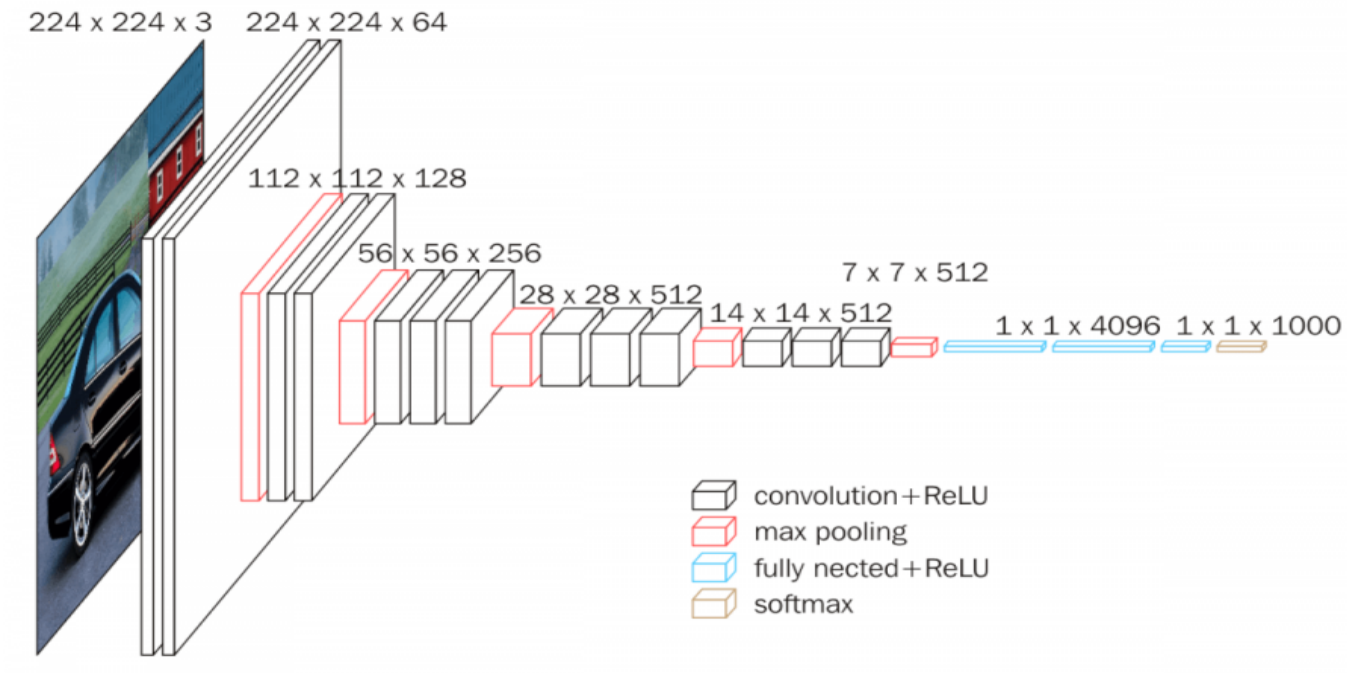


FIGURE 2.8 – **Architecture de VGG16**
[26].

VGG16 est en compétition pour le gagnant de la tâche de classification (GoogLeNet avec une erreur de 6,7 %) et surpasse considérablement la soumission gagnante ILSVRC-2013 Clarifai, qui a atteint 11,2% avec des données de formation externes.
En ce qui concerne les performances du réseau unique, l'architecture VGG16 obtient le meilleur résultat (erreur de test de 7,0%), surpassant de 0,9% un GoogLeNet (réseau neuronal convolutif de 22 couches de profondeur) unique[27].

2.10 Conclusion

Ce chapitre a été consacré à la présentation de l'intelligence artificielle qui correspond à un ensemble de concepts et de technologies plus qu'une discipline autonome . En effet le but des scientifiques dans le domaine de l'IA est de pouvoir simuler l'intelligence humaine et le comportement d'un être humain doté de conscience et de sentiments. Et si ces machines pouvaient nous

dominer, elles deviendraient supérieures à l'homme. De plus elles possèdent un avantage qui n'est pas négligeable qui est l'absence de contraintes physiques. En effet elles n'ont pas besoin de dormir pour se régénérer, de manger pour avoir la forme, elles ne peuvent pas être malades.

Ensuite nous avons présenté et évaluer les algorithmes de détection faciale, d'objets ainsi de mouvements qui sont les algorithmes les plus robustes et utilisés en littérature. Cette étape primordiale nous a bien éclairé à choisir l'algorithme adéquat pour la résolution des problèmes des distractions du conducteur.
De ce fait, nous avons opté pour la détection de mouvements afin de faire face aux problèmes des distractions du conducteur. Ce choix de la détection des mouvements est favorisé par rapport à la détection faciale ou détection d'objets, puisqu'il est d'actualité et capable de détecter tous les mouvements et expressions faciales.
La détection de mouvement se fera grâce à un réseau VGG16 préformé qui est un modèle réalisé à base du réseau neuronal CNN (Convolutional neural network). VGG16 s'est révélé très efficace et plus performant pour détecter les conducteurs distraits suite au nombre de couches que ce dernier comporte. VGG16 est utilisé dans de nombreux problèmes de classification d'images d'apprentissage en profondeur. Ainsi, nous pourrions former un modèle qui a une précision d'entraînement de 97%, ce qui est beaucoup mieux par rapport aux divers modèles comme (Réseaux de neurones à convolution).
Cependant, il demeure un excellent élément de construction utilisé à des fins d'apprentissage car il est facile à mettre en œuvre.
Dans le chapitre suivant, nous allons passer en revue les principaux travaux relatifs à notre domaine d'étude. L'état de l'art sera accompagné d'une analyse et comparaison de ces travaux.

Chapitre 3

État de l'art

3.1 Introduction

A mesure que la densité du trafic augmente, le nombre d'accidents devrait encore augmenter et la majorité des accidents signalés sont causés par des conducteurs distraits. A cet effet, nous pensons que la vision par ordinateur peut accroître les efforts des gouvernements pour prévenir les accidents causés par la distraction au volant.
La vision par ordinateur (ou Computer Vision) a base d'intelligence artificielle permet de créer des applications révolutionnaires surtout avec l'apparition des algorithmes robustes du Deep Learning, qui est une nouveauté dont les progrès importants réalisés ces dernières années lui permettent d'apporter une réelle valeur ajoutée, à tel point qu'il fait aujourd'hui partie intégrante de notre quotidien. En plus des conversations qu'il permet de tenir avec nos machines, le Deep Learning peut être exploité dans divers domaines d'applications.
Les nouveaux outils et techniques avancées de nos jours ont permis une amélioration considérable des algorithmes d'apprentissage en profondeur au point de surpasser les humains pour classer les images ou permettre à un assistant à commande vocale comme « Amazon Echo » et « Google Home » de trouver et télécharger cette nouvelle chanson qu'on aime. . . etc.
L'objectif de notre travail est de développer des méthodes avancées d'apprentissage pour l'analyse et l'interprétation automatique des différentes distractions des conducteurs à partir de sources d'informations diverses, telles que les images, les vidéos, afin de diminuer le nombre d'accidents et de décès ainsi que les dégâts matériels.
Bien qu'il existe de nombreux algorithmes de détection d'expressions faciales et d'objets qui fonctionnent convenablement dans des environnements contraints, divers changements au niveau des images présentent un grand défi face à un système de reconnaissance qui doit être robuste en ce qui concerne les grandes variabilités. Pour faire face à ce problème, il est important de choisir une représentation appropriée des images du visage. Cette représentation doit être compacte et significative.
Le but de ce chapitre est de présenter l'état de l'art des différentes méthodes de détection de mouvements proposées par plusieurs auteurs et de faire une étude comparative entre ces travaux. Une analyse des méthodes proposées est également introduite.

3.2 Travaux connexes

Dans cette section, nous passons en revue les principales approches liées au domaine de la prévision des accidents de la route, qui convergent vers notre étude et se concentrent principalement sur la détection de la fatigue et des distractions des conducteurs.
Dans (Jabon et al, 2010), les auteurs utilisent le classificateur d'apprentissage par machine pour l'analyse de l'expression faciale afin de prédire les comportements de conduite dangereuse et les accidents mineurs et majeurs. Le processus comprend différentes étapes, la première étape concerne l'extraction des traits du visage, ils utilisent les vidéos de visages collectées pour extraire les principaux traits du visage et les mouvements de la tête. L'étape suivante consiste en la synchronisation des données, ils utilisent les sorties vidéos synchronisées avec les sorties du simulateur de conduite et le calcul des statistiques des séries chronologiques, le calcul des statistiques du domaine fréquentiel, la création de l'ensemble de données finales et l'extraction des caractéristiques du chi-square, en termes d'entrées. En général, les vidéos et les données de conduite de Faces sont collectées à partir du simulateur STISIM.

Les auteurs dans (Sigari et al., 2013) utilisent une méthode de correspondance pour la détection de la rotation de la tête et un système expert flou pour estimer l'hypovigilance concernant la vision industrielle pour le système de surveillance du visage du conducteur. L'approche proposée pourrait détecter l'hypovigilance du conducteur (fatigue et distraction) par le traitement des régions des yeux et du visage. L'acquisition d'images de détection des visages est la première étape du processus.
Les symptômes d'hypovigilance sont extraits de l'image du visage. Cependant, une étape de détection explicite de l'œil n'est pas utilisée pour déterminer l'œil du visage, mais certains des symptômes importants liés à la région œil (segment de la moitié supérieure du visage) sont extraits.
En outre, une méthode de correspondance de gabarit est utilisée pour détecter la rotation de la tête. Enfin, ils utilisent un système expert flou pour estimer l'hypovigilance du conducteur. Cette approche est efficace pour différents individus ayant des comportements différents au niveau du visage et des paupières en temps réel, ce qui réduit le risque d'accident.

Dans (Fu et al., 2015), les auteurs proposent un modèle de détection de mouvements inhabituels en temps réel, qui comporte deux étapes : la détection des régions saillantes et la détection des mouvements inhabituels. Ils utilisent le modèle d'attention temporelle, la transformation caractéristique invariante de l'échelle SIFT (scale-invariant feature transform) et la technique du flux optique. L'approche utilise cette technique de détection de mouvements inhabituels pour détecter le risque de collision pour les points d'images considérés et surtout pour détecter de manière efficace et efficiente les zones de mouvement inhabituelles et ils y parviennent.
Dans (Tran et al.,2018), les auteurs utilisent des algorithmes d'apprentissage profond, ils ont donc testé les algorithmes suivants VGG-16, GoogleNet, Alexnet, Resnet qui sont tous basés sur les

réseaux neuronals convolutionnels (CNN).
Les systèmes d'assistance à la conduite utilisent cette technique pour détecter les distractions du conducteur.
Afin d'évaluer et de valider les modèles CNN pour la détection de la distraction, ils ont mené des expériences sur le simulateur de conduite assistée. Ils ont utilisé en entrée les images des conducteurs dans des postures de conduite normale et distraite, les algorithmes sont mis en œuvre et évalués sur une plate-forme GPU intégrée, et plus encore.
Ils ont développé un système d'alerte conversationnelle qui avertit le conducteur en temps réel lorsqu'il ne se concentre pas sur la tâche de conduite. En conséquence, ils créent des expériences de conduite réalistes. Les résultats expérimentaux montrent également que l'approche proposée est plus efficace que l'approche de base, qui ne possède que 256 neurones dans les couches entièrement connectées. En outre, les résultats indiquent que GoogleNet est le meilleur modèle des quatres pour la détection de la distraction sur notre banc d'essai de simulateur de conduite.
Une autre étude proposée par Jeong et Ko (Jeong et Ko, 2018) qui utilise des classificateurs forestiers aléatoires pondérés hiérarchiques (WRF) pour une conduite sûre en temps réel. La technique proposée est utilisée pour la reconnaissance de l'expression faciale des conducteurs. Les classificateurs WRF sont utilisés pour obtenir une classification plus précise. Le premier classificateur WRF permet d'apprendre à distinguer la peur, le bonheur et un autre groupe d'expressions. En effet, les trois émotions sont la colère, le dégoût et la tristesse ont des caractéristiques faciales similaires et peuvent donc être classées plus précisément dans le deuxième niveau.
Dans le deuxième niveau, le deuxième WRF classe la colère, le dégoût et la tristesse de l'autre groupe pour obtenir une classification plus précise. Les deux types de classificateurs WRF sont appris séparément à l'aide de deux vecteurs de caractéristiques, en termes d'entrées ; ils utilisent les bases de données FER (Facial Emotion Recognition), CK (Cohn Kanade) et MMI (Man Machine Interface). Les auteurs affirment que les résultats obtenus sur les expressions faciales à partir de la matrice de confusion sont très satisfaisants.
Récemment, Wilhelm a proposé dans (Wilhelm, 2019) une méthode basée sur les CNNs pour la reconnaissance des expressions faciales des conducteurs pour l'analyse de l'expression faciale dans un système d'assistance à la conduite.
Dans ce contexte, un modèle basé sur le ShuffleNet a un nombre total de 871849 paramètres et la couche de sortie est un log softmax. Le modèle a été formé de bout en bout à l'aide de la NLLLoss (fonction en pytorch) et de l'optimiseur Adam. Dans le cadre d'un autre projet sur l'identification du visage, ils ont formé un réseau basé sur une dorsale MobileNetV2 de taille 128x128 et était sensiblement plus petit avec seulement 84871 paramètres. Ils ont remplacé la dernière couche et ont retravaillé le modèle en utilisant la perte ArcFace et l'optimiseur Adam. La proposition fournit des résultats prometteurs avec le démon ShuffleNet.
Pour ce qui est de l'étude de (M.Alotaibi et B. Alotaibi,2019), les auteurs ont proposé un système de détection des distractions du conducteur en utilisant une combinaison de trois techniques les plus avancées en matière d'apprentissage profond le réseau résiduel (ResNet) et le réseau neuronal récurent hiérarchique (HRNN) et l'architecture inception (architecture de réseau neuronal profond).

Dans ce modèle, un bloc ResNet (Residential Energy Services Network) et 2 couches HRNN sont intégrés au module inception et sont suivis de 2 couches denses et enfin du classificateur soft-max. Afin d'évaluer les résultats de la méthode proposée, ils ont utilisé l'ensemble de données de la state farm Distracted Driver Detection, tel que fournie par Kaggle, l'expérience leur a permis de conclure que leur modèle apprend des représentations plus riches avec un petit nombre de paramètres, ils ont divisé leurs données en ensembles de tests et de formations selon des pourcentages, ils ont pris 10%, 20%, 30%, pour la formation (Training) et le reste pour le test. La méthode proposée a donné des résultats prometteurs, en effet, ils ont réussi à combiner plusieurs techniques pour avoir un modèle plus efficace et plus robuste.

Approche	Catégorie	Source de données	Sortie	Technique utilisée	Outil supporté	Avantages
(Jabon et al, 2010).	Systèmes d'aide à la conduite (DAS).	Vidéos de visages collectées et données de conduite du simulateur STISIM.	Modèles de calcul des prévisions d'accidents mineurs et majeurs.	Classificateur d'apprentissage machine.	Oui.	Prévision des accidents mineurs et majeurs.
(Sigari et al., 2013).	Vision industrielle.	Images de visages. vidéos capturées.	Détection en temps réel de la fatigue et de la distraction du conducteur.	Méthode d'appariement pour la rotation de la tête. . système expert flou pour estimer l'hypovigilance.	Oui.	Utilisé efficacement pour différents individus ayant des comportements différents au niveau du visage et des paupières. Réduire les risques d'accidents.

(Fu et al., 2015).	Systèmes d'aide à la conduite (DAS).	Séquences vidéo. les images vidéo.	la détection de mouvements inhabituels.	Modèle d'attention temporelle. Transformation caractéristique invariante à l'échelle (SIFT). La technique du flux optique.	Oui.	Détecter le risque de collision pour les points d'images Considéré. Détecter de manière efficace et efficiente les zones de mouvement inhabituelles.
(Tran et al.,2018).	Système d'aide à la conduite.	les images des conducteurs dans des postures de conduite normales et distraites.	Détection des mouvements du conducteur et création d'expériences de conduite réalistes.	Réseaux neuronaux convolutionnels profonds (VGG-16 , AlexNet, GoogleNet and ResNet).	Oui.	L'approche proposée est plus performante que l'approche de base.
(Jeong et Ko, 2018).	Systèmes avancé d'aide à la conduite (ADAS).	Bases de données FER,CK (Cohn Kanade) et MMI.	Matrices de confusion pour les bases de données CK et MMI.	Un système hiérarchique de classification aléatoire pondérée des forêts (WRF).	Non.	Bonne détection des expressions faciales.

(Wilhelm, 2019).	Systèmes d'aide à la conduite (DAS) Système FER.	Les images des visages.	Reconnaissance des expressions faciales du conducteur.	Méthode basée sur les réseaux de neurones convolutifs en pytorche : ShuffleNet, MobileNetV2.	Oui.	Des résultats prometteurs avec le démon ShuffleNet.
(M.Alotaibi et B. Alotaibi,2019).	Système de transports intelligents.	Images de Caméra 2D.	Reconnaissance du comportement du conducteur.	Reseau résiduel (RESNET), reseau neuronal hiérarchique (HRNN), Architechture Inception.	Non.	Résultats très précis. Apprendre des représentations plus pérformanantes avec un petit nombre de paramétres.

TABLE 3.1: **Présentation des différentes approches proposées par les auteurs.**

Le tableau 1 résume les principales caractéristiques des approches citées ci-dessus. Le tableau contient 8 colonnes qui indiquent un critère de comparaison comme suit :

— la colonne "approche" désigne l'approche sous-jacente,

— La colonne "catégorie" indique à quelle fin l'approche est utilisée,

— La colonne "source de données" décrit les données et les sources utilisées en entrée,

— La colonne "production" implique le résultat de l'approche,

— La colonne "technique utilisée" indique toutes les techniques utilisées pour arriver au résultat,

— La colonne "outil soutenu" désigne la mise en œuvre de l'approche,

— La colonne "avantages" indique les avantages de l'approche.

3.3 Analyse et critiques

Après analyse du tableau nous avons constaté que la plupart des auteurs ont obtenus des résultats satisfaisant sur la détection et la reconnaissance des visages. En effet, les auteurs avec la détection de la fatigue et de la somnolence du conducteur, ont pu réduire le nombre d'accidents et cela grâce aux algorithmes proposés.
Cependant, malgré les résultats encourageants, nous constatons certains manquements :

- Dans les travaux de (Jabon et al, 2010), l'étude ne tient pas compte des mouvements de véhicules et des effets sur l'environnement. D'autre part, leur technique ne permet pas d'analyser le taux de fausses alertes dans le temps pour les classificateurs (pas les systèmes en temps réel).
- La méthode de suivi de visage proposé par (Sigari et al., 2013) est imprécise et très complexe d'un point de vue informatique, en effet la capture du visage est faite d'une manière inexacte et la complexité de l'algorithme est très élevée.
- Dans le papier de (Fu et al., 2015), nous constatons l'absence de prise en compte des trames successives et la technique proposée n'est pas assez robuste.
- Dans (Tran et al.,2018), le système proposé ne prend pas en compte la détection des expressions faciales tel que la fatigue et somnolence, en outre, le choix des algorithmes s'est porté sur GoogleNet qui n'est pas assez précis pour effectuer la tâche de détection de mouvements.
- Pour ce qui est des travaux de (Jeong et Ko, 2018) et en dépit de leur bonne détection d'expression faciale, ils n'ont pas obtenu de bons résultats lorsque le visage est tourné ou partiellement occulté par des objets.
- Dans les travaux de (Wilhelm, 2019), les performances de MobileNetV2 sont inférieures en raison de la complexité réduite du modèle et de la taille réduite des données d'entrée.
- Enfin, dans les travaux de (M.Alotaibi et B. Alotaibiet,2019), malgré que les résultats de leur méthode prouvent que cette dernière peut maintenir une performance en temps réel , le modèle nécessite beaucoup de temps de calcul.

Afin de remédier à ces lacunes, nous proposons une approche qui consiste à former un modèle permettant la détection des distractions des conducteurs de transport public. A cet effet, nous utiliserons d'une part un ensemble d'algorithmes robustes composé d'un CNN pour améliorer la performance de détection des comportements distraits du conducteur et d'autre part un logiciel qui traite les données et qui envoie un signal d'alerte pour avertir le conducteur lorsque ce dernier est distrait.

3.4 Conclusion

Dans ce chapitre, nous avons présenté une revue de la littérature des principales contributions dans le domaine la détection de mouvement et de somnolence du conducteur, ces travaux ont

été choisis par rapport à notre étude. Une synthèse de ces travaux a été présenté pour cerner la problématique et montrer l'impact de notre contribution.
En général, les chercheurs ont essayé de réaliser différentes tâches pour détecter la distraction et la fatigue du conducteur en utilisant différentes méthodes et techniques mais leurs algorithmes ne sont pas assez robustes et précis pour effectuer la tâche de détection de distractions.
Afin de remédier aux insuffisances des approches citées, nous proposons une approche qui consiste à former un modèle permettant la détection des distractions des conducteurs de transport public. A cet effet, nous allons utiliser d'une part un ensemble d'algorithmes robustes composé d'un CNN pour améliorer la performance de détection des comportements distraits du conducteur et d'autre part un logiciel qui traite les données et envoie un signal d'alerte pour avertir le conducteur lorsque ce dernier est distrait.
Dans le chapitre suivant, nous allons présenter notre approche en détail.

Chapitre 4

Approche proposée

4.1 Introduction

De nos jours, les techniques mises en œuvre pour doter les machines de capacités cognitives similaire à l'homme sont nombreuses, citant par exemple la logique floue, les algorithmes génétiques, l'inférence bayésienne. Mais aujourd'hui c'est l'apprentissage automatique et plus précisément les réseaux de neurones et l'apprentissage profond qui règnent sur cette discipline au point d'être sur les lèvres de toutes les grandes entreprises. Parmi les exemples répandus de ce domaine le Deep Learning utilisé dans les caméras de vidéosurveillance qui sont omniprésentes dans les lieux publics pour détecter les comportements suspects, le secteur bancaire ou bien le secteur de transport public et l'automobile qui souffre de la hausse des accidents et cela en dépit de tous les efforts fournis par le gouvernement.
L'évolution spectaculaire de l'intelligence artificielle dans les systèmes de vidéosurveillance s'explique par les performances sans cesse améliorées des calculateurs et des techniques utilisées entre autres la vision par ordinateur qui est actuellement l'un des domaines de recherche les plus en vogue au sein du Deep Learning. Il se situe à l'intersection de nombreux sujets académiques, tel que l'informatique (graphisme, algorithmes, théorie, systèmes, architecture), les mathématiques, l'ingénierie (robotique, PNL, traitement d'image), la physique (optique), La biologie (neurosciences) et la psychologie (sciences cognitives).
Les domaines d'applications de la vision assistée par l'ordinateur sont immenses, surtout en utilisant le Deep Learning qui construit lui-même les caractéristiques d'analyse, ainsi il peut identifier des caractéristiques non visibles par l'homme, faciliter les études et automatiser les actions, donc gagner plus de temps.
Les algorithmes du Deep Learning utilisés dans le domaine d'industrie automobiles sont immenses. Nous trouvons en tête de liste le réseau neuronal convolutionnel qui était la source de plusieurs autres algorithmes améliorés comme GoogleNet ou encore avec l'apparition de l'apprentissage par transfert avec les deux techniques Vgg16 et Vgg19.
Dans ce chapitre, nous allons présenter en détail l'approche proposée qui consiste en la détection des distractions du conducteur de transport public. Nous allons commencer par définir l'architecture du système proposé tout en détaillant chaque phase du processus accompagnée de quelques

algorithmes.

4.2 Approche proposée

Notre approche vise à réduire le nombre d'accidents et de décès grâce à une technique plus efficace qui consiste à créer un modèle pour détecter les distractions des conducteurs de transport public. Pour ce faire, nous avons utilisé d'une part CNN et d'autre part une technique robuste qui contient plus de couches qu'un simple CNN, basée sur l'apprentissage par transfert nommée Vgg16 permettant de résoudre les distractions des conducteurs de différentes manières telles que : parler aux passagers, au téléphone, se maquiller ...etc.

4.2.1 Architecture du système

Afin de mettre en œuvre notre système de détection des distractions, nous avons proposé une architecture basée sur le Deep Learning (figure 4.1).

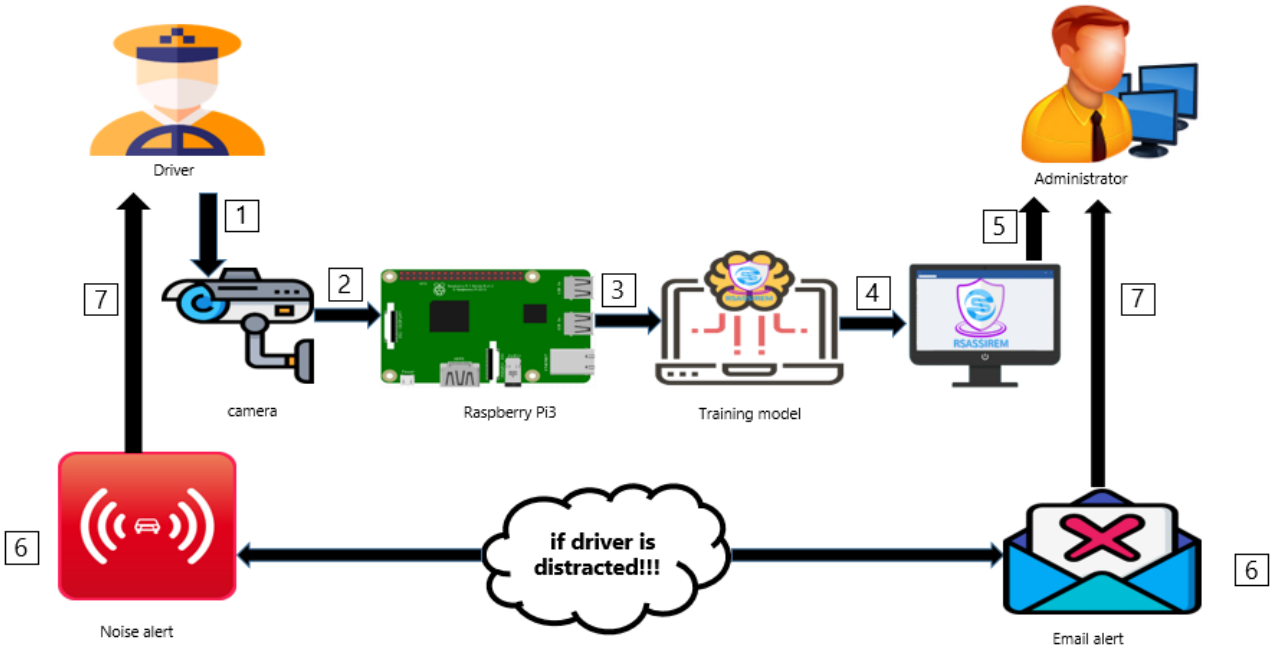


FIGURE 4.1 – **Architecture du système.**

.

Le déroulement de notre processus passe par six étapes :

1. Lors de la conception de notre système, nous avons utilisé une caméra REV 1.3 dont le rôle est de surveiller le conducteur.
2. La camera envoie les données au nano ordinateur 'Raspberry pi3 ' qui sera équipé d'un module WIFI.
3. Le Raspberry transmet les données reçues par la camera vers notre logiciel baptisé « RSassirem » R et S acronyme de nos premières lettres de nos prénoms (Reda, Samia), assirem c'est un terme qui signifie espoir en tamazight. RSassirem va représenter aux yeux de ses clients une source d'espoir, il va contribuer à réduire le nombre de décès et d'accidents due à la distraction des conducteurs.
4. Le logiciel géré par un administrateur transmis les données reçues par le Raspberry à notre modèle d'apprentissage profond.
5. Notre système utilise deux modèles avancés du Deep Learning à savoir CNN qui est un modèle de convolution de base, nous avons entrainé l'algorithme avec le dataset puis nous avons évalué le modèle avec les données captées par la caméra. Nous avons utilisé une seconde fois le dataset avec une autre technique de convolution encore meilleure que CNN qui est VGG16 et nous avons évalué le modèle avec les données captées par la caméra.
6. Après avoir détecté les distractions du conducteur, un email sera transmis à l'administrateur du logiciel afin de sauvegarder cette distraction sur la base de données du système.
   Un message d'alerte sous forme de bruits sonores émis du Raspberry sera envoyé au conducteur pour l'avertir du danger courant lorsque ce dernier est distrait.

Dans ce qui suit, nous allons présenter en détail les modèles deep learning que nous avons utilisé pour détecter les distractions du conducteur, cette partie reflète l'étape 5 de notre processus.

4.2.2 Détection des distractions au volant

4.2.2.1 CNN

L'idée du réseau neuronal convolutif est essentiellement un type de réseau neuronal artificiel (ANN) qui s'inspire du cortex visuel animal, ce réseau a été proposé par LeCun [35] et a fait une percée dans le domaine de la classification des images et de la détection des cibles.
Les CNN profonds introduisent un nombre important de couches cachées, réduisant ainsi la dimensionnalité de l'image et permettant au modèle d'extraire les caractéristiques éparses de l'image dans un espace à faible dimension.
Un CNN est essentiellement formé par l'empilement de ces couches l'une après l'autre. Depuis 2012 les CNNs ont progressé très rapidement en raison de la disponibilité d'une grande quantité de données étiquetées et de la puissance de calculs. Diverses architectures comme AlexNet, VG-GNet, GoogleNet, ResNet ont établi des références en matière de vision par ordinateur.
Pour tenter de résoudre le problème de la distraction du conducteur, nous avons opté à utiliser un simple réseau neuronal convolutionel (Figure 4.2). Techniquement, l'apprentissage en profondeur

des modèles CNN pour former et tester chaque image d'entrée la passera à travers une série de couches de convolution avec filtres (Kernels), Pooling, couches entièrement connectées et appliquera la fonction Softmax.

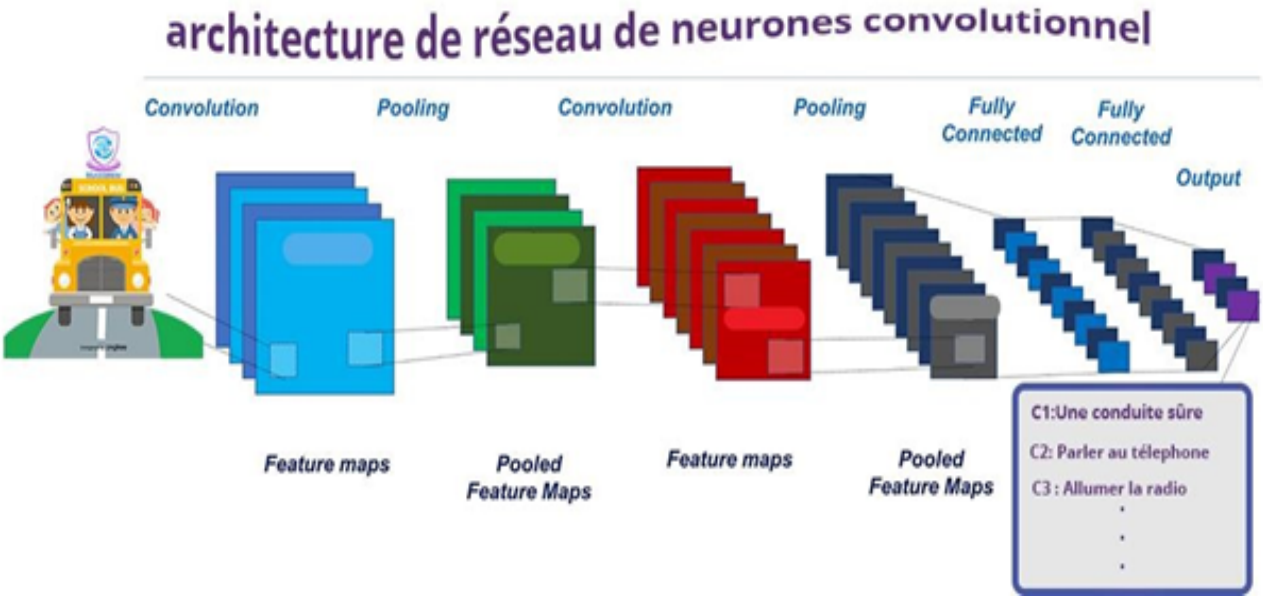


FIGURE 4.2 – **Architecture CNN.**

1. **La convolution** est la première couche à extraire des entités d'une image d'entrée. Cette dernière préserve la relation entre les pixels en apprenant les caractéristiques de l'image à l'aide de petits carrés de données d'entrée. Il s'agit d'une opération mathématique qui prend deux entrées telles qu'une matrice d'images et un filtre ou un noyau.
   Nous utilisons **Feature map** (Figure 4.3) qui est un ensemble des extracteurs / filtres de fonctionnalités appris par la formation. Lorsqu'ils sont liés avec l'entrée et passés par la fonction d'activation, ils génèrent des entrées significatives pour la couche ou la sortie suivante.

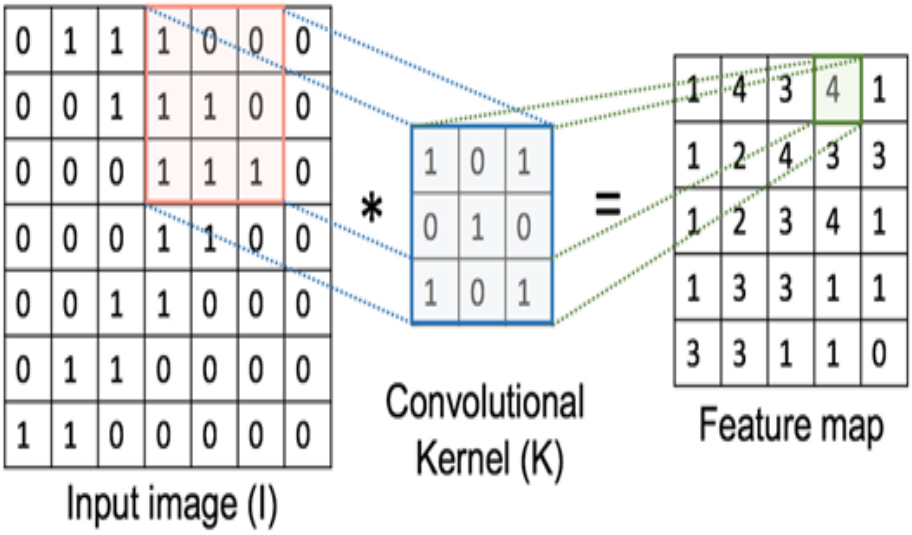


FIGURE 4.3 – **Feature Map [36].**

Stride : est le nombre de pixels décalés sur la matrice d'entrée. Lorsque la foulée est de 1, nous déplaçons les filtres à 1 pixel à la fois. Lorsque la foulée est de 2, nous déplaçons les filtres à 2 pixels à la fois et ainsi de suite.
Padding : Parfois, le filtre ne correspond pas parfaitement à l'image d'entrée. Nous avons deux options :

- Remplir l'image avec des zéros pour qu'elle s'adapte.
- Déposer la partie de l'image où le filtre ne correspondait pas. C'est ce qu'on appelle un remplissage valide qui ne conserve qu'une partie valide de l'image.

2. **Relu ()** :Pour que tout type de réseau neuronal soit puissant, il doit contenir la non-linéarité, réduit la quantité d'informations générées par la couche convolutionnelles pour chaque entité et conserve les informations les plus essentielles (le processus des couches convolutionnelles et de regroupement se répète généralement plusieurs fois) (Figure 4.4).

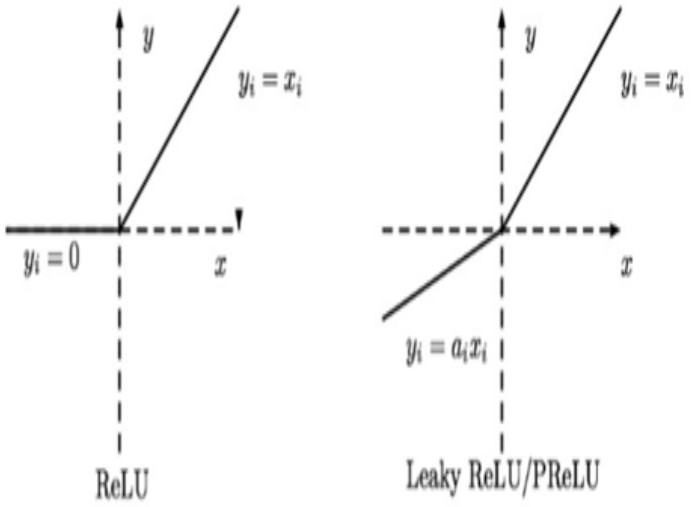


FIGURE 4.4 – **Fonction Relu.**

3. **Couche d'entrée entièrement connectée (Pooling)** :Après une opération de convolution, nous effectuons généralement la mise en commun (Figure 4.5) pour réduire la dimensionnalité. Cela nous permet de réduire le nombre de paramètres, ce qui raccourcit le temps d'entraînement et combat le sur-apprentissage. La mise en commun des couches sous-échantillonne chaque carte d'entités indépendamment, ce qui réduit la hauteur et la largeur, tout en conservant la profondeur intacte.

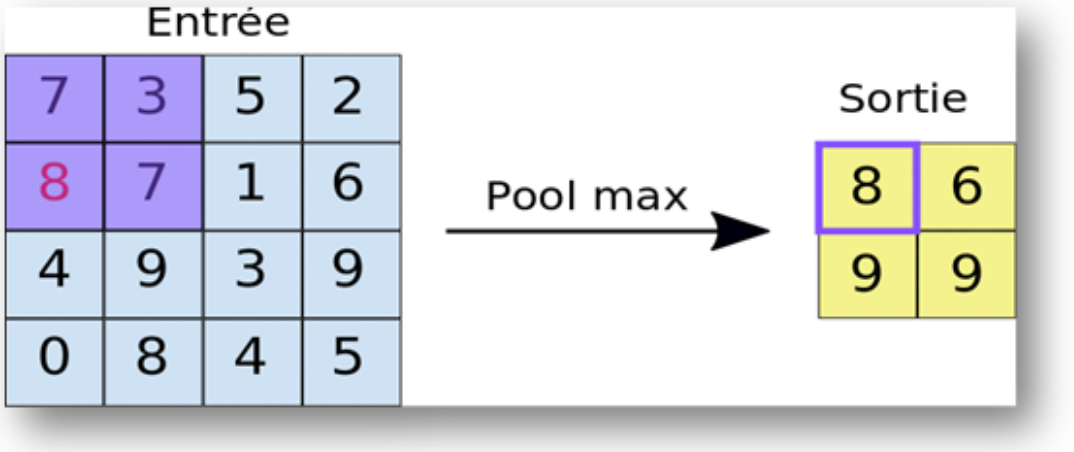


FIGURE 4.5 – **Max pooling [36].**

4. **Couche entièrement connectée (fully-connected)** :applique des pondérations à l'entrée générée par l'analyse d'entités pour prédire une étiquette précise.
   La couche de sortie entièrement connectée : génère les probabilités finales pour déterminer une classe pour l'image.

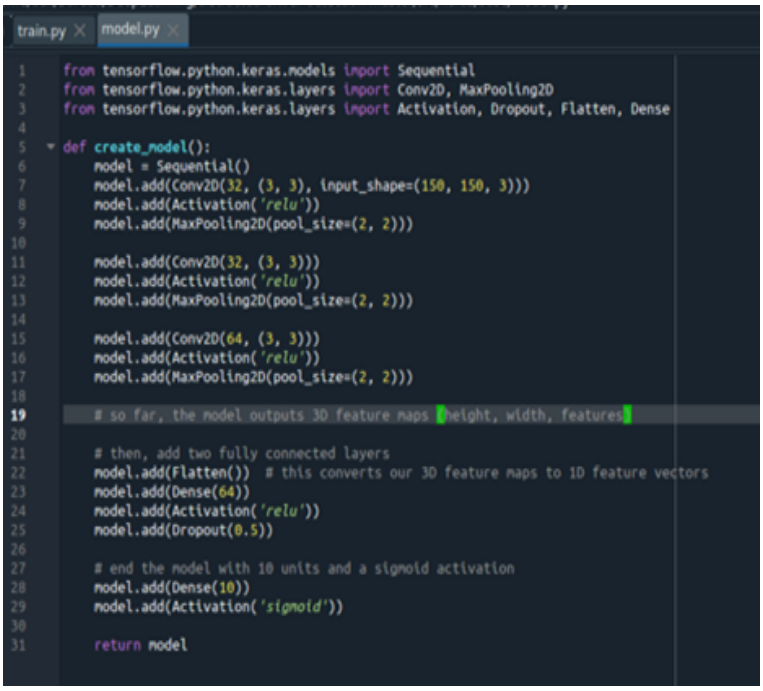


FIGURE 4.6 – **Notre modèle CNN.**

Structurellement, le code ci-dessus (Figure 4.6) ressemble au réseau neuronal artificiel ANN, il se constitue de 5 nouvelles méthodes :

— **Conv2D** : cette méthode crée une couche convolutionnelle. Le premier paramètre est le nombre de filtres et le second est la taille du filtre. Par exemple, dans la première couche de convolution, nous créons 32 filtres de taille 3x3. Nous utilisons la non-linéarité relu comme activation, Nous activons également le rembourrage. Dans Keras, il existe deux options de remplissage : identiques ou valides. Stride est 1 pour les couches de convolution par défaut, donc nous ne changeons pas cela.
  Cette couche peut être personnalisée davantage avec des paramètres supplémentaires.

— **MaxPooling2D** : crée une couche maxpool, le seul argument est la taille de la fenêtre. Nous utilisons une fenêtre 2x2 car c'est la plus courante. Par défaut, la longueur de foulée est égale à la taille de la fenêtre, qui est de 2 dans notre cas, donc nous ne changeons pas cela.

— **Flatten** :aplatit les sorties générées par les couches précédentes pour les transformer en un seul vecteur qui peut être utilisé comme entrée pour la couche suivante.

— **Dense (64)** :est une couche avec 64 unités cachées.

— **Dropout** :C'est la technique de régularisation la plus populaire pour les réseaux de neurones profonds (Figure 4.7). Même les modèles de pointe qui ont une précision de 95% obtiennent une augmentation de précision de 2% simplement en ajoutant un Dropout. Il est utilisé pour éviter le sur-apprentissage, L'hyper-paramètre p est appelé le taux de décrochage et c'est généralement un nombre autour de 0,5 correspondant à 50% des neurones abandonnés.

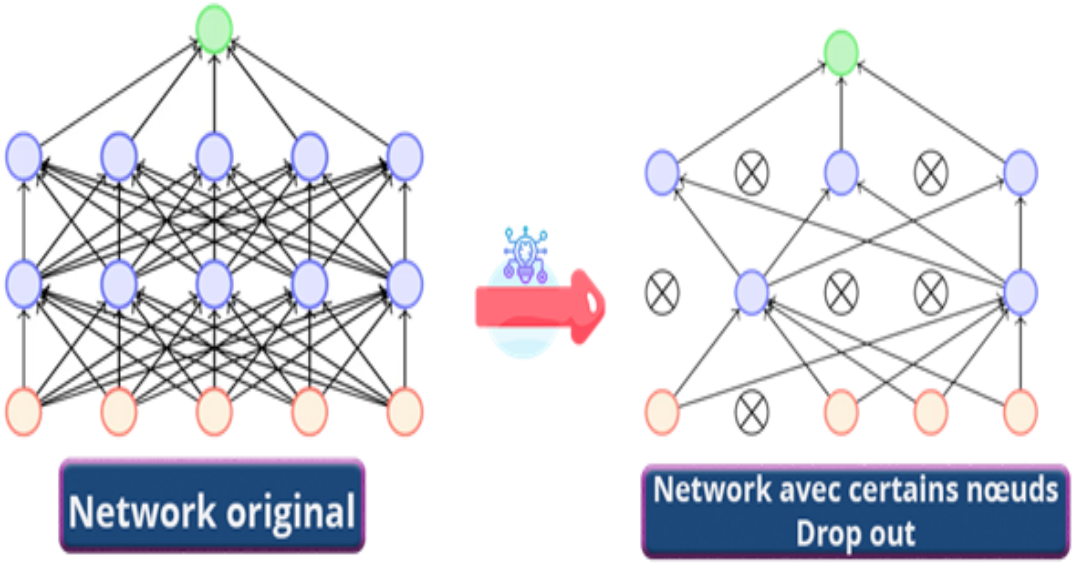


FIGURE 4.7 – **Régularisation Dropout dans le deep learning.**

4.2.2.2 Le sur-apprentissage (over-fitting)

Il se produit lorsque vous réalisez un bon ajustement de votre modèle sur les données d'entraînement, alors qu'il ne se généralise pas bien sur les nouvelles données invisibles. En d'autres termes, le modèle a appris des modèles spécifiques aux données d'apprentissage, qui ne sont pas pertinents dans d'autres données [37].
Il existe plusieurs manières de réduire le sur-apprentissage dans les modèles d'apprentissage en profondeur. La meilleure option est d'obtenir davantage de données d'entraînement. Malheureusement, dans des situations réelles, Il est impossible en raison du temps, du budget ou des contraintes techniques de mettre ça en pratique. Une autre façon de réduire le sur-apprentissage consiste à réduire la capacité du modèle à mémoriser les données d'entraînement. En tant que tel, le modèle devra se concentrer sur les modèles pertinents dans les données de formation, ce qui se traduit par une meilleure généralisation.

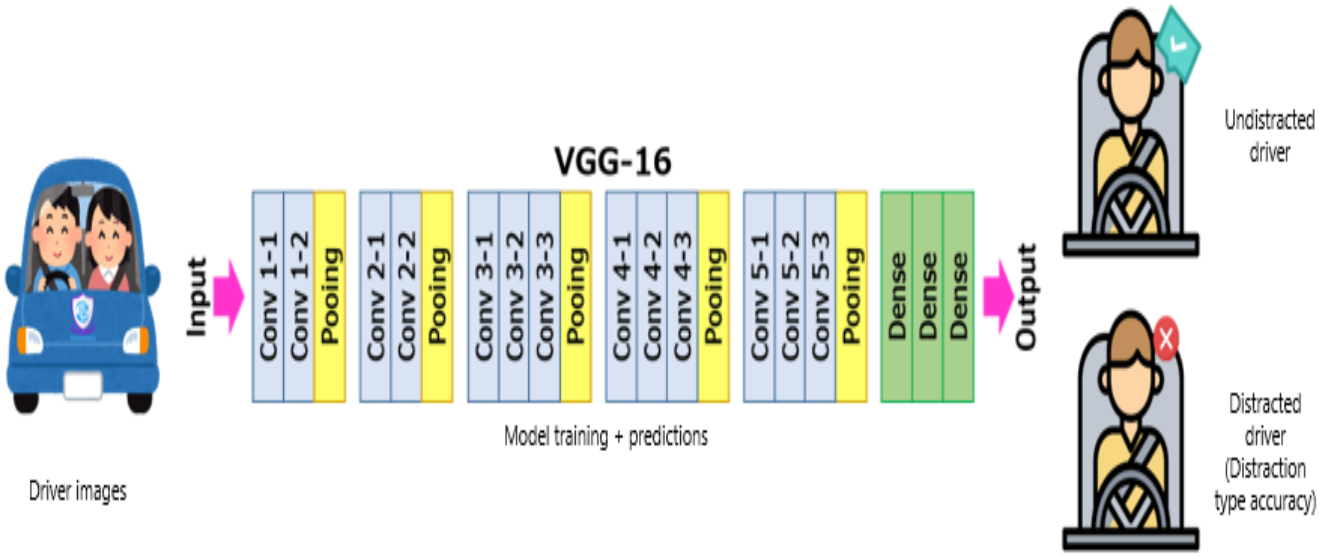


FIGURE 4.8 – **Architecture VGG16.**

4.2.2.3 Architecture du VGG16

L'objet de notre étude est VGG-16, une version du réseau de neurones convolutif très connu appelé VGG-Net.
La figure 4.8 nous montre que la couche cov1 est d'une image RVB de taille fixe 224 x 224. L'image est passée à travers une pile de couches convolutionnelles (conv.).
Dans l'une des configurations, il utilise également des filtres à convolution 1x1, qui peuvent être considérés comme une transformation linéaire des canaux d'entrée (suivie d'une non-linéarité). La foulée de convolution est fixée à 1 pixel ; le remplissage spatial de conv. L'entrée de la couche est telle que la résolution spatiale est préservée après convolution, c'est-à-dire que le remplissage est de 1 pixel pour 3x3 conv. Couches. La mise en commun spatiale est réalisée par cinq couches de mise en commun maximale, qui suivent une partie de la conv. La mise en commun maximale est effectuée sur une fenêtre de 2x2 pixels, Trois couches entièrement connectées (FC) suivent une pile de couches convolutives (qui ont une profondeur différente dans différentes architectures) : les deux premières ont 4096 canaux chacune, la troisième effectue une classification ILSVRC à 1000 voies et contient ainsi 1000 canaux (un pour chaque classe).
La couche finale est la couche soft-max. La configuration des couches entièrement connectées est la même dans tous les réseaux. Toutes les couches cachées sont équipées de la non-linéarité de rectification (ReLU). Il est également noté qu'aucun des réseaux ne contient de normalisation de réponse locale (LRN), une telle normalisation n'améliore pas les performances sur l'ensemble de données ILSVRC, mais entraîne une consommation de mémoire et un temps de calcul accrus.
VGG-16 apprend 138 357 544 paramètres. Pour trouver ce nombre, il suffit de compter les poids

de toutes les couches de convolution et fully-connected, sans oublier les paramètres de biais.
Par exemple, pour la première couche de convolution, le réseau doit apprendre 64 filtres en couleurs (donc de profondeur 3) de taille 3 3, ainsi qu'un paramètre de biais pour chaque filtre. Cela fait un total de (3*3*3) *64 + 64 = 1792 paramètres.
Le nombre de paramètres d'une couche fully-connected s'obtient en multipliant le nombre d'éléments en sortie avec celui en entrée, et en ajoutant le nombre de biais. Ainsi, le réseau doit apprendre 7*7*512*4096 + 4096 = 102 764 544 paramètres pour la première couche fully-connected, 4096*4096 + 4096 = 16 781 312 pour la deuxième, et 4096*1000 + 1000 = 4 097 000 pour la dernière.

4.2.2.4 Transfert d'apprentissage sur le modèle VGG16

La formation d'un modèle prend énormément de ressources et de temps. Plus encore si l'ensemble de données est gros comme dans VGG16, cependant en utilisant la méthode d'apprentissage par transfert, nous pouvons simplement attacher notre ensemble de données à un modèle déjà formé, comme il a un biais et une efficacité parfaite.
L'apprentissage par transfert fait généralement référence à un processus où un modèle formé sur un problème est utilisé d'une manière ou d'une autre pour un deuxième problème connexe.
Dans l'apprentissage en profondeur, l'apprentissage par transfert est une technique par laquelle un modèle de réseau neuronal est d'abord formé sur un problème similaire au problème en cours de résolution. Une ou plusieurs couches du modèle formé sont ensuite utilisées dans un nouveau modèle formé sur le problème d'intérêt[38].
Dans ce qui suit, nous allons appliquer l'apprentissage par transfert sur VGG16.Le modèle sera téléchargé sur Internet et pourrait prendre un certain temps en utilisant github ou bien il sera fourni localement.

4.2.2.5 Optimisation pour l'apprentissage en Deep Learning

La descente de gradient : La descente de gradient est un algorithme d'optimisation souvent utilisé pour trouver les poids ou les coefficients des algorithmes d'apprentissage automatique, tels que les réseaux de neurones artificiels et la régression logistique. Cela fonctionne en permettant au modèle de faire des prédictions sur les données d'apprentissage et en utilisant l'erreur sur les prédictions pour mettre à jour le modèle de manière à réduire l'erreur.
Le but de l'algorithme est de trouver des paramètres de modèle (par exemple, des coefficients ou des poids) qui minimisent l'erreur du modèle sur le jeu de données d'apprentissage. Pour ce faire, il modifie le modèle en le déplaçant le long d'une pente d'erreur vers une valeur d'erreur minimale. Cela donne à l'algorithme le nom de « descente de gradient »[39]. Il existe trois variantes de cette méthode :

— Batch gradient descent,

— Descente de gradient stochastique,

— Mini-batch gradient descent.

4.2.2.6 Algorithmes d'optimisation de la descente de gradient

— **Adagrad** est un algorithme d'optimisation basé sur la descente de gradient qui ne fait qu'adapter le taux d'apprentissage aux paramètres, effectuant de plus grandes mises à jour pour les caractéristiques peu fréquentes et de plus petites pour les caractéristiques plus fréquentes.

— **RMSprop** est une méthode de taux d'apprentissage adaptative qui a été proposée par Geoff Hinton [40]. Au lieu d'accumuler tous les carrés des gradients précédents, nous allons tout simplement restreindre la fenêtre des gradients accumulés à une taille fixée w. Au lieu de stocker les w carrés des gradients précédents, on applique une moyenne mobile exponentielle des carrés des gradients précédents.

— **Adam** est une méthode d'optimisation stochastique. Il hérite quelque caractéristique de Adagrad et RMSprop. Adam est connu pour sa simplicité et le peu de mémoire qu'il demande, il convient bien aux problèmes importants en termes de données et / ou de paramètres.

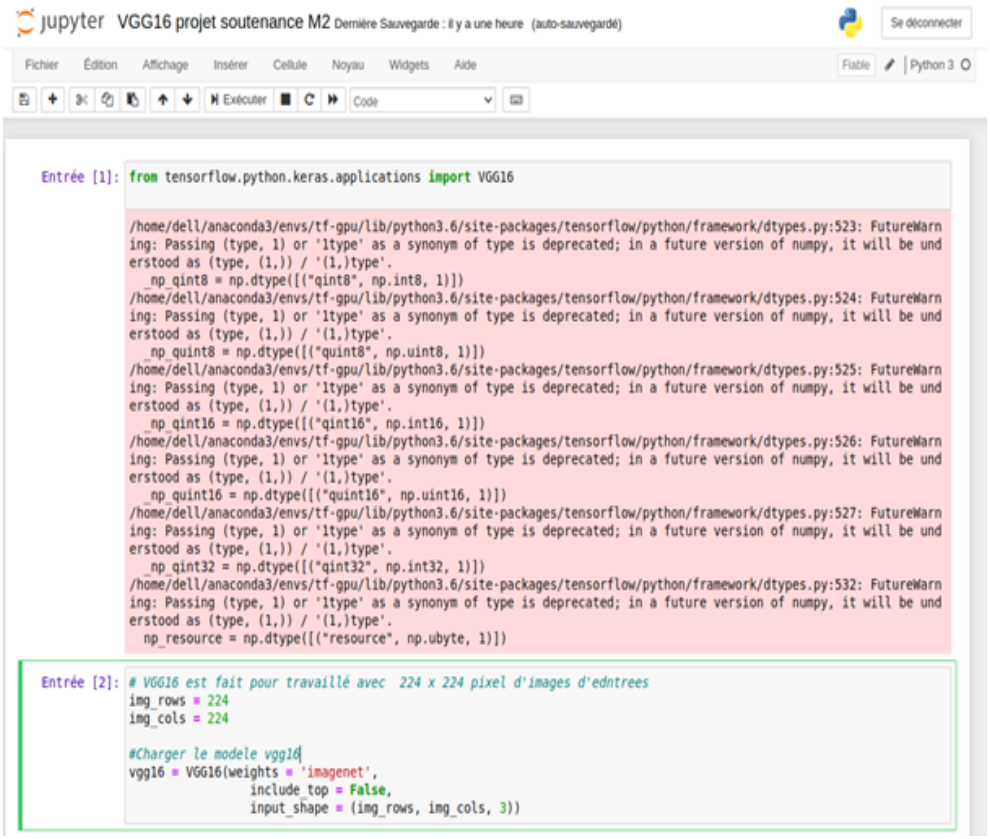


FIGURE 4.9 – **Importation du modèle VGG16 (Localement).**

Ici ,nous importons toutes les bibliothèques dont nous aurons besoin pour implémenter VGG16 (Figure 4.9).
Nous utilisons la méthode séquentielle, le modèle séquentiel signifie que toutes les couches du modèle seront disposées en séquence.
Nous avons importé ImageDataGenerator à partir de keras.preprocessing., son objectif est d'importer facilement des données avec des étiquettes dans le modèle. C'est une classe très utile car elle a de nombreuses fonctions pour redimensionner, faire pivoter, zoomer, retourner... etc. La chose la plus utile à propos de cette classe est qu'elle n'affecte pas les données stockées sur le disque. Cette classe modifie les données en déplacement tout en les transmettant au modèle.
Nous allons maintenant lister tous les calques et voir lesquels sont tous définis pour être recyclés par défaut (Figure 4.10).

Visualiser chaque couche

```
Entrée [5]: for (i,layer) in enumerate(vgg16.layers):
                print(str(i)+ " "+layer.__class__.__name__,layer.trainable)
```

```
0 InputLayer True
1 Conv2D True
2 Conv2D True
3 MaxPooling2D True
4 Conv2D True
5 Conv2D True
6 MaxPooling2D True
7 Conv2D True
8 Conv2D True
9 Conv2D True
10 MaxPooling2D True
11 Conv2D True
12 Conv2D True
13 Conv2D True
14 MaxPooling2D True
15 Conv2D True
16 Conv2D True
17 Conv2D True
18 MaxPooling2D True
```

FIGURE 4.10 – **Visualisation des couches.**

Nous allons définir la forme d'entrée pour le retraining et définir toutes les couches entraînées prédéfinis sur False afin qu'ils ne se recyclent pas (Figure 4.11).

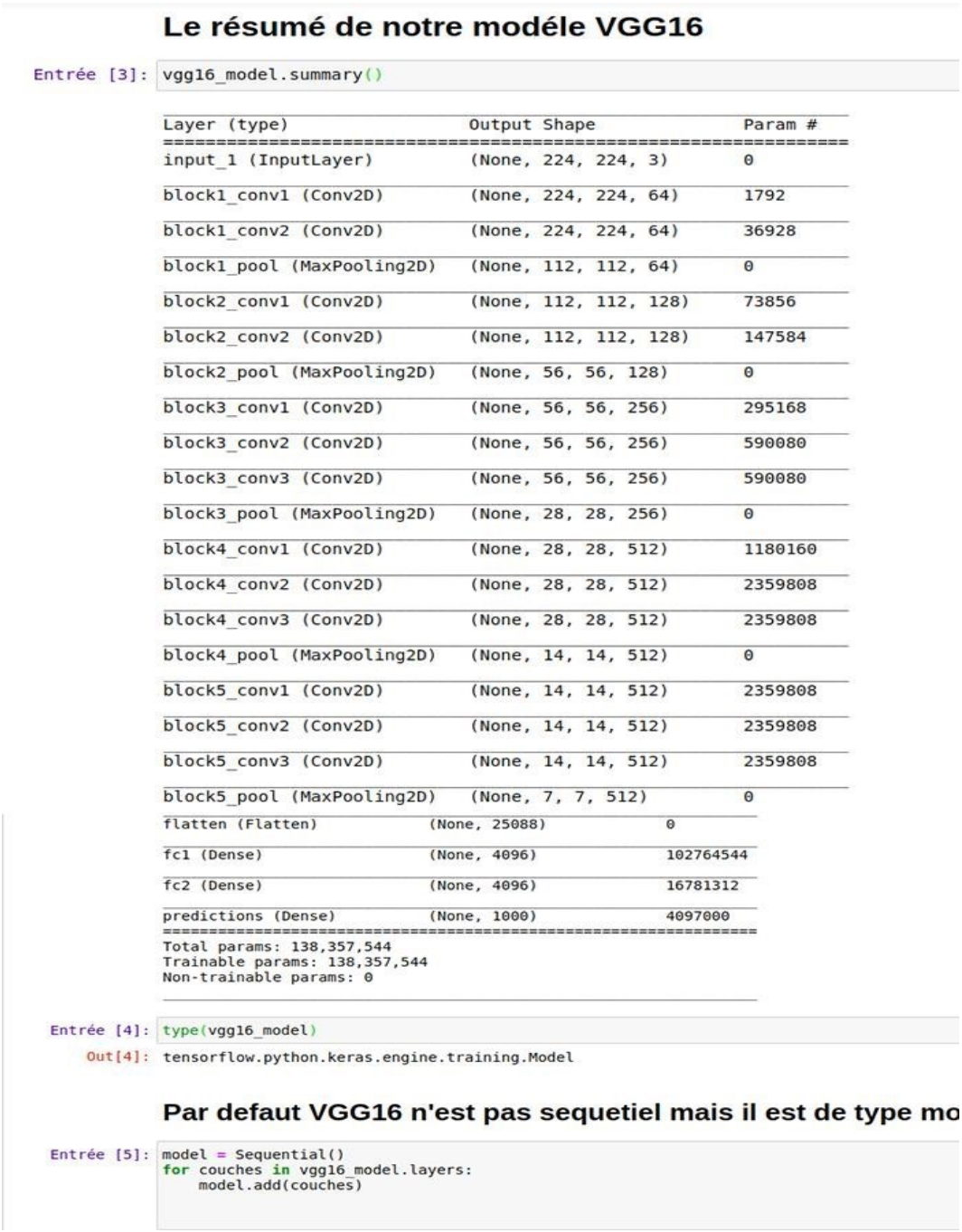
geler toutes les couches expecté le top 4

```
Entrée [3]: from tensorflow.python.keras.applications import VGG16

            # VGG16 was designed to work on 224 x 224 pixel input images sizes
            img_rows = 224
            img_cols = 224

            # Recharger VGG16 model sans the top or FC layers
            vgg16 = VGG16(weights = 'imagenet',
                          include_top = False,
                          input_shape = (img_rows, img_cols, 3))

            # Here we freeze the last 4 layers
            # Layers are set to trainable as True by default
            for layer in vgg16.layers:
                layer.trainable = False

            # Let's print our layers
            for (i,layer) in enumerate(vgg16.layers):
                print(str(i) + " "+ layer.__class__.__name__, layer.trainable)
```

```
0 InputLayer False
1 Conv2D False
2 Conv2D False
3 MaxPooling2D False
4 Conv2D False
5 Conv2D False
6 MaxPooling2D False
7 Conv2D False
8 Conv2D False
9 Conv2D False
10 MaxPooling2D False
11 Conv2D False
12 Conv2D False
13 Conv2D False
14 MaxPooling2D False
15 Conv2D False
16 Conv2D False
17 Conv2D False
18 MaxPooling2D False
```

FIGURE 4.11 – **Modification des couches prédéfini.**

La figure 4.12 illustre le modèle Vgg16 en détails.



Le résumé de notre modéle VGG16

Entrée [3]:
```
vgg16_model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 224, 224, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
predictions (Dense)          (None, 1000)              4097000
=================================================================
Total params: 138,357,544
Trainable params: 138,357,544
Non-trainable params: 0
_________________________________________________________________
```

Entrée [4]:
```
type(vgg16_model)
```

Out[4]:
```
tensorflow.python.keras.engine.training.Model
```

Par defaut VGG16 n'est pas sequetiel mais il est de type mo

Entrée [5]:
```
model = Sequential()
for couches in vgg16_model.layers:
    model.add(couches)
```

FIGURE 4.12 – **Visualisation détaillé du modèle VGG16.**

Nous complétons la construction de notre modèle VGG16 en enlevant la dernière Dense qui contient 1000 catégories comme la figure ci-dessus le montre, nous allons prendre comme exemples 2 catégories, nous utilisons la fonction pop () de python qui a pour rôle de supprimer la dernière ligne (Figure 4.13).

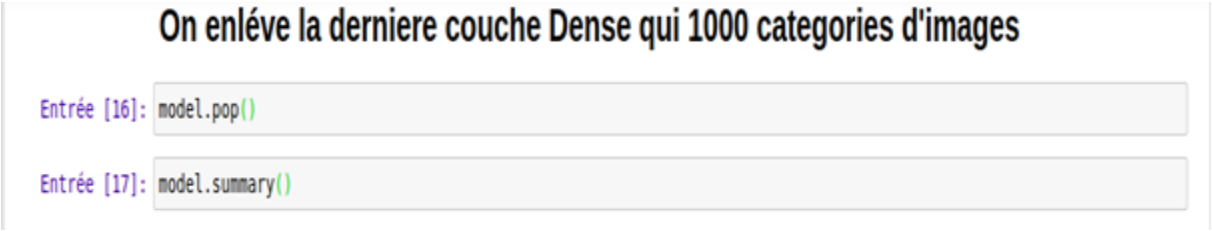


FIGURE 4.13 – **Suppression de la dernière couche Dense.**

Pour finir la réalisation de notre modèle préformé, nous ajoutons la couche Dense avec 2 catégories comme exemple, comme nous définissons la fonction d'activation pour notre modèle qui est la fonction softmax (Figure 4.14).

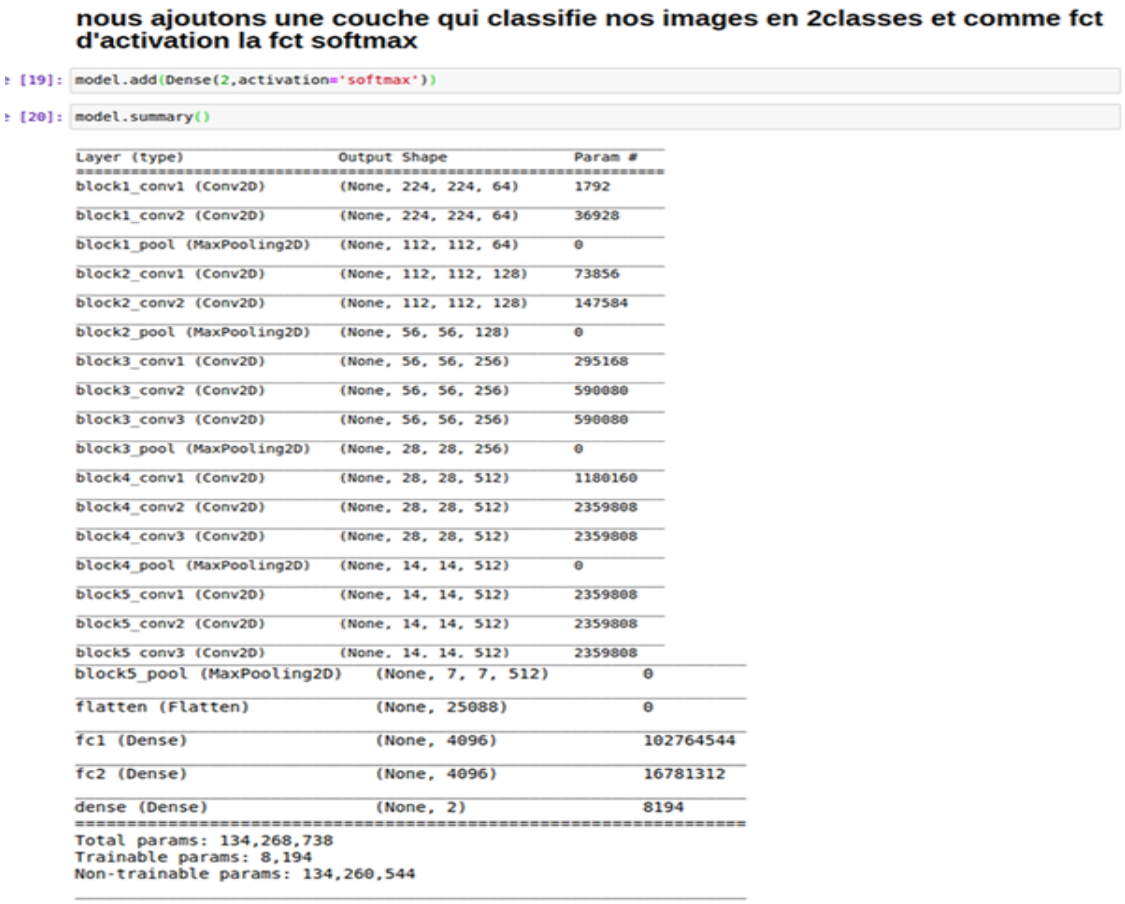


FIGURE 4.14 – **Ajout de la couche Dense et la fonction d'activation.**

4.2.2.7 Préparation de notre modèle de détection des distractions de conducteurs de transport public

La stratégie de cette mise en œuvre de la classification des images des conducteurs distraits consiste à utiliser le VGG16. Ce réseau a été préformé sur l'ensemble de données ImageNet il est facilement accessible via la bibliothèque Keras.
L'idée est que le réseau préformé aurait appris des caractéristiques des données précédentes qui pourraient s'avérer utiles pour prédire les images des conducteurs distraits. Bien entendu, cela ne fonctionnera pas tout de suite, car le réseau a été formé pour prédire des choses comme les chats et les chiens, et ne transfère pas directement les images aux conducteurs. La stratégie consiste à déconnecter les couches entièrement connectées du VGG16, car elles contiennent des poids très spécifiques à l'ensemble de données original d'ImageNet.
Afin d'améliorer le modèle pour qu'il corresponde à l'ensemble de données du conducteur distrait, un nouveau modèle séquentiel a été créé avec un classificateur entièrement connecté similaire à celui du CNN simple.

4.3 Conclusion

L'apprentissage profond a connu un très grand progrès ces dernières années, plus particulièrement des réseaux convolutifs. Une des nouvelles encourageantes est que la plupart de ces progrès ne sont pas seulement le résultat d'un matériel plus puissant, d'ensembles de données plus importantes et de modèles plus grands, mais surtout la conséquence de nouvelles idées, de nouveaux algorithmes et d'architectures de réseau amélioré. Cela nous motive et nous encourage à nous lancer dans ce voyage d'apprentissage approfondi afin de surmonter les difficultés.
Dans ce chapitre, nous avons décrit la démarche suivie dans le but d'obtenir le meilleur modèle de détection de distraction du conducteur de transport public. Pour réaliser notre architecture nous avons commencé en premier lieu par définir le modèle CNN simple et l'analyse de ses défauts en observant son comportement grâce aux captures que nous avons fournies, nous avons changé aussi les paramètres un à un pour comprendre l'impact de chacun sur la performance finale.
Ensuite nous avons expliqué notre modèle basé sur les réseaux convolutionnels qui est le vgg16 préformé , dans cette phase nous avons modifié le modèle vgg16 et nous avons appliqué plusieurs techniques de régularisations pour éviter le sur-ajustement des données d'entraînement et d'avoir les meilleurs précisions .A cet effet nous avons utilisé aussi l'apprentissage par transfert qui a l'avantage de réduire le temps de formation pour notre modèle et peut entraîner une erreur de généralisation plus faible.
Après avoir conçu l'architecture qui convient au modèle proposé, la suite du travail sera consacrée à la mise en œuvre de notre modèle, détailler les différentes phases du logiciel et expliquer la partie électronique qui consiste à réaliser la liaison entre la partie HARD (caméra surveillance) et la partie SOFT (Logiciel).

Chapitre 5

Expérimentation

5.1 Introduction

Le réseau de neurones à convolution (ConvNets ou CNN) est l'une des principales catégories pour effectuer la reconnaissance d'images, les classifications d'images, la détection d'objets, la reconnaissance de visages ... etc. Ces technologies appartiennent à des domaines où les CNN sont largement utilisés.
Dans notre cas nous avons utilisé VGG16 pour détecter les distractions des conducteurs. VGG16 est une architecture de réseau neuronal à convolution (CNN) qui a été utilisée pour remporter le concours ILSVR (Imagenet) en 2014. Elle est considérée comme l'une des excellentes architectures de modèle de vision jusqu'à ce jour, ce réseau est assez grand, il a environ 138 millions de paramètres. Nous avons implémenté le VGG16 complet à partir de zéro dans Keras. Cette implémentation se fera sur l'ensemble de données pour prédire 10 types de distractions.
Dans ce chapitre, nous allons essayer de projeter la lumière sur les grandes étapes de réalisation de notre projet, en commençant par le choix des outils software, hardware et langages de développement. Nous allons expliquer notre modèle tout en détaillant l'étape d'entraînement et de test, nous poursuivons ce chapitre par la présentation des différentes fonctionnalités et interfaces de notre logiciel, et nous terminons par expliquer notre choix de carte d'acquisition qui est le Raspberry pi3 qui permet de faciliter la réalisation de notre prototype en temps réel.

5.2 Présentation des outils

5.2.1 Outils logiciels

5.2.1.1 Environnement

- **Pycharm** : est l'IDE le plus populaire pour Python, et comprend d'excellentes fonctionnalités telles qu'une excellente complétion et inspection du code avec un débogueur avancé et une prise en charge de la programmation Web et de divers cadres. Il est créé par la société tchèque Jet brains qui se concentre sur la création d'un environnement de développement intégré pour divers langages De développement Web [41].
- **Anaconda** : est un gestionnaire de packages, une distribution gratuite et open-source des langages de programmation Python et R pour le calcul scientifique qui vise à simplifier le package gestion et déploiement [42].
- **Spyder** : est un environnement de développement pour Python. Libre et multiplateforme, il intègre de nombreuses bibliothèques d'usage scientifique : Matplotlib, NumPy, SciPy et IPython [42].
- **Jupyter-notebook** : Le bloc-notes Jupyter est un environnement informatique interactif qui permet aux utilisateurs de créer des documents de bloc-notes comprenant : du code en direct, des widgets interactifs, des tracés, du texte narratif, des équations, des images, de la vidéo [42].
- **Qt Designer** : est l'outil Qt pour concevoir et construire des interfaces utilisateur graphiques (GUI) avec Qt Widgets. Vous pouvez composer et personnaliser vos fenêtres ou boîtes de dialogue à la manière de ce que vous voyez et les tester à l'aide de différents styles et résolutions [41].
- **MySQL Workbench** : est un logiciel de gestion de base de données MySQL. Disponible sous Windows, Mac et Linux, il permet de gérer des tables (ajout, modification, suppression) à travers une interface graphique simple d'usage [43].

5.2.1.2 Framework

- **Tensorflow** : Il fut créé par l'équipe Google Brain pour mener des recherches sur le ML et le Deep Learning. Il est considéré comme une version moderne de Theano [42].
- **Keras** : Le Framework le plus haut niveau, le plus convivial de la liste. Il permet aux utilisateurs de choisir si les modèles qu'ils construisent sont exécutés sur Theano ou TensorFlow. Il est écrit et entretenu par Francis Chollet, un autre membre de l'équipe Google Brain [42].

5.2.1.3 Bibliothèques utilisées

- **OpenCV (Open Source Computer Vision Library)** : est une bibliothèque de logiciels open source de vision par ordinateur et d'apprentissage automatique, elle possède plus de

2500 algorithmes optimisés, qui comprennent un ensemble complet d'algorithmes de vision par ordinateur et d'apprentissage machine à la fois classique et à la pointe de la technologie [42].

- **Numpy** : Est une bibliothèque permettant d'effectuer des calculs numériques avec Python. Elle introduit une gestion facilitée des tableaux de nombres, des fonctions sophistiquées (diffusion), on peut aussi l'intégrer le code C / C ++ et Fortran [42].
- **Pandas** : est une bibliothèque open source sous licence BSD fournissant des structures de données hautes performances et faciles à utiliser, ainsi que des outils d'analyse des données pour le langage de programmation Python [42].
- **Matplotlib** : est une bibliothèque de traçage pour le langage de programmation Python et son extension mathématique numérique NumPy. Il fournit une API orientée objet permettant d'incorporer des graphiques dans les applications[42].
- **Scikit-learn** : est une bibliothèque développée en Python, un langage de programmation de haut niveau. Elle est dédiée à l'apprentissage statistique (machine learning) et peut être utilisée comme middleware, notamment pour des tâches de prédiction[42].

5.2.1.4 Langages de programmation

- **Python 3** : est le langage de programmation le plus utilisé dans le domaine du Machine Learning, du Big Data et de la Data Science[41].

5.2.1.5 Langages de manipulation de données

- **SQL (Structured Query Language)** : est un langage informatique utilisé pour exploiter des bases de données. Il permet de façon générale la définition, la manipulation et le contrôle de sécurité de données[43].

5.2.2 Outils matériels

- **Raspberry pi3** : Le Raspberry pi est un nano ordinateur de la taille d'une carte de crédit que l'on peut brancher à un écran et utilisé comme un ordinateur standard. Sa petite taille, et son prix intéressant fait du Raspberry pi un produit idéal pour tester différentes choses, et notamment la création d'un serveur Web chez soi. Évidemment, pour sa taille il ne faut pas s'attendre à des performances incroyables, mais pour mettre en ligne des projets à montrer au client ou expérimenter avec linux c'est largement suffisant [44].
- **Caméra surveillance** : est un système de caméras et de transmission d'images, disposé dans un espace public ou privé pour le surveiller à distance ; il s'agit donc d'un type de télésurveillance, et en ce qui concerne nos tests nous utiliserons une caméra rev1.3 [44].

Le DeepLearning est un domaine avec des exigences en calculs intenses et la disponibilité des ressources (surtout en GPU et TPU) dédiés à cette tache vont fondamentalement influencer sur l'expérience de l'utilisateur car sans ses ressources, il faudra trop de temps pour apprendre des

erreurs ce qui peut être décourageant.

Les expérimentations ont tous été effectuées sur une machine qui offre des performances acceptables dont voici les caractéristiques :

CPU	Intel® Core™ i7-8565U CPU 1.80GHz 1.99GHz
RAM	8 GO
GPU	NVIDIA Geforce MX150 4GB

Nous avons utilisé pendant la réalisation de notre système de détection des distractions du conducteur le système d'exploitation ubuntu 18.04 parce qu'il n'appelle pas d'autre processus gourmands en dehors de l'exécution de notre jeu de données, par conséquent notre ordinateur aura assez de mémoire cache pour exécuter notre code et enregistrer les calculs nécessaires.

5.3 Modèle vgg16 pour la détection des distractions

FIGURE 5.1 – **Exemple de classification de quelques images du dataset.**

5.3.1 Fractionnement des données

Pour les besoins de notre travail, nous avons utilisé l'ensemble des données fournies par State Farm contient près de 100 000 images qui relèvent des 10 classes Plus précisément (Figure 5.1), l'ensemble de données a été divisé à l'origine en données de formation et de test, avec respectivement 22 424 et 79 726 images dans chacune de ces catégories (Figure 5.2). La principale différence entre les deux est que les données de formation contenaient les étiquettes de chaque image, alors que les

images de test ne le faisaient pas. Cela a été fait délibérément puisqu'il s'agissait d'un concours et, comme tel, cela a permis une certaine réussite entre les différentes soumissions. Cependant, dans ce travail, afin de former et d'évaluer la performance des modèles, les 22 424 images étiquetées pour la formation ont été divisées en 70% de données réelles de formation, 10% de validation et 20% de données de test. Cela nous permet d'évaluer le modèle hors ligne sans avoir à le soumettre au Kaggle mais aussi de tester le modèle par rapport à des données invisibles et d'appliquer nos propres mesures de test. La crainte initiale est que les données ne soient pas suffisantes pour entraîner le modèle, mais elle sera complétée par des techniques d'apprentissage par augmentation et transfert de données.

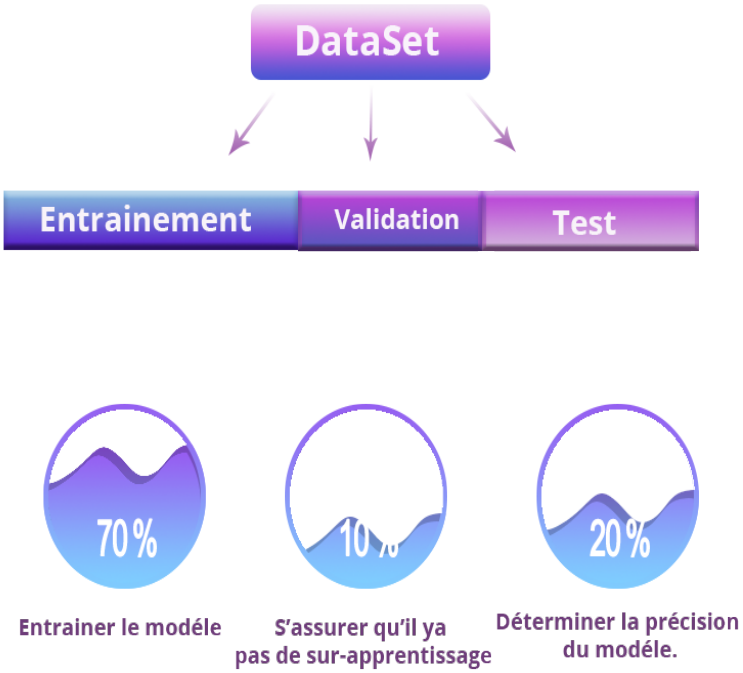


FIGURE 5.2 – **Fractionnement du dataset.**

5.3.2 Formation du model avec Keras

L'apprentissage par transfert est le concept de l'apprentissage en profondeur dans lequel nous prenons un modèle existant qui est formé sur beaucoup plus de données et utilisons les fonctionnalités que le modèle a appris de ces données et les utilisons pour notre problème.
Les images de formation et de validation ont été propagées à travers le réseau du VGG16 une seule fois, les caractéristiques des goulots d'étranglement de sortie ont été enregistrées dans des fichiers et ont ensuite été introduites dans le modèle supérieur du classificateur entièrement connecté nouvellement créé. Il est important de noter que seul le modèle supérieur a été formé sur le nouvel ensemble de données, et qu'à ce titre, il a appris des caractéristiques qui ne sont présentes que dans l'ensemble de données du conducteur distrait. Le principal élément à retenir est le classificateur

entièrement connecté qui a été cousu au sommet du modèle VGG16 pour être formé sur les images du conducteur distrait. Il convient également de noter qu'en raison du temps de formation réduit, nous avons pu augmenter la taille des images à 224 x 224, ce qui a permis au modèle de retenir beaucoup d'informations.

— La première des choses à faire consiste à tout d'abord configurer l'environnement, et répartir les différents fractionnements d'ensembles de données en 3 catégories qui sont :
Ensembles de données de formation : L'ensemble de données réel que nous utilisons pour former le modèle (poids et biais dans le cas d'un réseau neuronal). Le modèle voit et apprend de ces données.
Jeu de données de validation : échantillon de données utilisé pour fournir une évaluation impartiale d'un ajustement du modèle sur l'ensemble de données d'apprentissage lors du réglage des hyperparamètres du modèle.
L'ensemble de validation est utilisé pour évaluer un modèle donné, mais c'est pour une évaluation fréquente. Nous utilisons ces données pour affiner les hyperparamètres du modèle. Par conséquent, le modèle voit de temps en temps ces données, mais il ne les prend pas en considération. Nous utilisons les résultats de l'ensemble de validation et mettons à jour les hyperparamètres de niveau supérieur. Ainsi, l'ensemble de validation affecte un modèle, mais seulement indirectement.
Jeu de données de test : échantillon de données utilisé pour fournir une évaluation impartiale d'un ajustement final du modèle sur l'ensemble de données de formation.
L'ensemble de données de test fournit l'étalon-or utilisé pour évaluer le modèle. Il n'est utilisé qu'une fois qu'un modèle est complètement formé (à l'aide des ensembles de train et de validation).

— Ensuite, installer toutes les bibliothèques citez ci-dessus.

— Importer les bibliothèques utilisées.

— Ensuite, nous allons importer le type de modèle séquentiel de Keras. Il s'agit simplement d'un empilement linéaire de couches de réseaux de neurones à travers la commande : from keras. models import Sequential

— Nous importerons les couches "principales" de Keras. Voici les couches utilisées dans presque tous les réseaux de neurones : From keras. Layers import Dense, Dropout, Activation, Flatten.

La figure ci-dessus illustre la création de notre modèle.

```python
from tensorflow.python.keras.models import Sequential
from tensorflow.keras.layers import Dropout, Flatten, Dense

target_size = 224, 224
batch_size = 16 #nombre d'exemples de formation utilisés dans une itération
class_labels_encoded = ['c0', 'c1', 'c2', 'c3', 'c4', 'c5', 'c6', 'c7', 'c8', 'c9']
class_labels = ['safe_driving', 'texting_right', 'talking_on_phone_right', 'texting_left', 'talking_on_phone_left',
                'operating_radio', 'drinking', 'reaching_behind', 'doing_hair_makeup', 'talking_to_passanger']
num_classes = len(class_labels)

def create_top_model(activation_func, input_shape):

    model = Sequential()

    model.add(Flatten(input_shape=input_shape))
    model.add(Dense(256, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(num_classes, activation=activation_func))

    return model
```

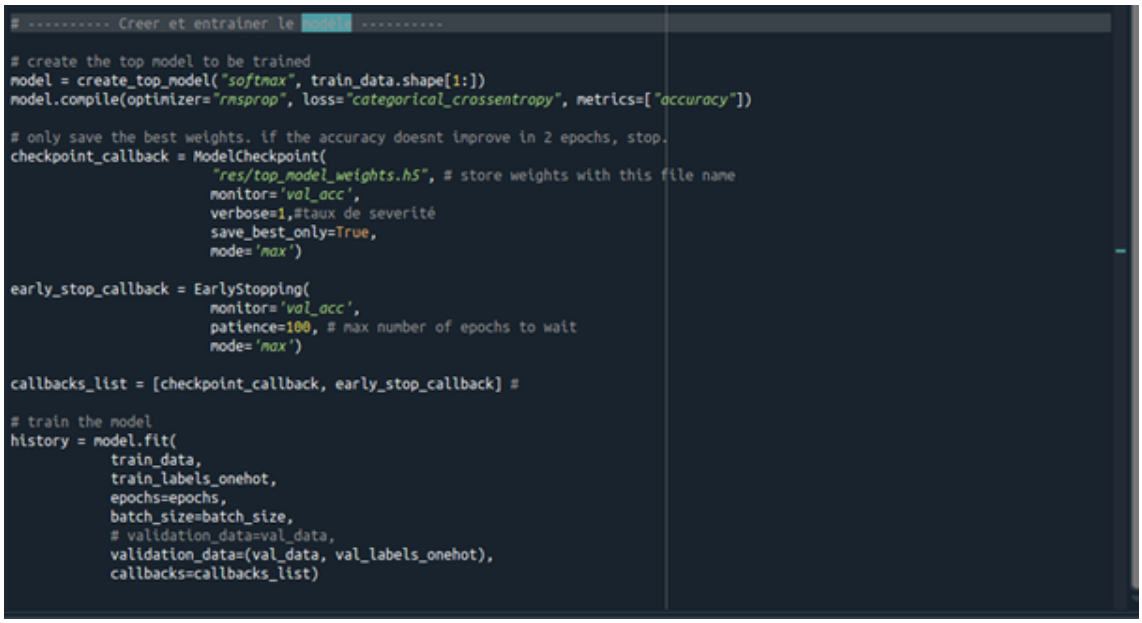
FIGURE 5.3 – **Création du modèle VGG16.**

5.3.3 Entrainement du modèle

La formation du top model a été effectuée de manière similaire à la méthode de formation appliquée dans le modèle Simple CNN. L'ensemble de validation a été utilisé avec une patience de 60 et a fonctionné avec 50 époques. Après 14 époques, il n'y a plus eu de changements (positifs) dans la précision de l'ensemble de validation, de sorte que l'entraînement a pris fin et que les meilleurs poids ont été conservés. L'exécution du modèle sur l'ensemble de test a permis d'obtenir une précision de 99,396 %. Ces résultats sont assez surprenants, car l'entraînement n'a pris que quelques minutes. Nous allons essayer de lui attribuée certains paramètres. Dans les réseaux de neurones, les paramètres sont utilisés pour former le modèle et faire des prédictions. Il existe deux types de paramètres :

- **Les paramètres du modèle** sont internes au réseau neuronal - par exemple, les poids des neurones. Ils sont estimés ou appris automatiquement à partir d'échantillons d'apprentissage. Ces paramètres sont également utilisés pour faire des prédictions dans un modèle de production.
- **Les hyperparamètres sont** des paramètres externes définis par l'opérateur du réseau neuronal - par exemple, la taille du lot utilisé lors de la formation. Les hyperparamètres ont un impact énorme sur la précision d'un réseau de neurones, il peut y avoir différentes valeurs optimales pour différentes valeurs et il n'est pas trivial de découvrir ces valeurs.
  Nous allons lister quelque hyperparamètres et paramètres que nous allons utiliser :
  Categorical-crossentropy : permet la réduction de l'erreur tout en utilisant l'algorithme de la descente de gradient.

Epoch : désigne le nombre d'itération dans notre base de données.
Optimiser : permet de réduire les poids des erreurs.
Loss : désigne le taux d'erreur.
Accuracy : désigne le taux de précision.
History : elle permet de donner au modèle toutes les images dont elle aura besoin en fonction du nombre d'itération (epochs) définit, dans le but de réduire le taux d'erreur.



```
# ---------- Creer et entrainer le modèle ----------

# create the top model to be trained
model = create_top_model("softmax", train_data.shape[1:])
model.compile(optimizer="rmsprop", loss="categorical_crossentropy", metrics=["accuracy"])

# only save the best weights. if the accuracy doesnt improve in 2 epochs, stop.
checkpoint_callback = ModelCheckpoint(
                      "res/top_model_weights.h5", # store weights with this file name
                      monitor='val_acc',
                      verbose=1,#taux de severité
                      save_best_only=True,
                      mode='max')

early_stop_callback = EarlyStopping(
                      monitor='val_acc',
                      patience=100, # max number of epochs to wait
                      mode='max')

callbacks_list = [checkpoint_callback, early_stop_callback] #

# train the model
history = model.fit(
           train_data,
           train_labels_onehot,
           epochs=epochs,
           batch_size=batch_size,
           # validation_data=val_data,
           validation_data=(val_data, val_labels_onehot),
           callbacks=callbacks_list)
```

FIGURE 5.4 – **Entraînement du modèle.**

La figure 5.4 montre la compilation de notre modèle VGG16, le taux d'apprentissage de l'optimiseur rmsprop (optimiseur qui utilise l'amplitude des gradients récents pour normaliser les gradients , le taux d'apprentissage change avec le temps), nous avons utilisé categorical-crossentropy comme perte car la sortie du modèle est catégorique .
Nous avons implémenté EarlyStopping, les rappels ModelCheckpoint pour le modèle comme nous avons adapté le modèle en utilisant fit-generator qui sert à former notre modèle d'apprentissage profond.La fonction fit-generator est utilisé lorsque l'ensemble de données est trop volumineux pour tenir en mémoire et où l'augmentation des données doit être appliquée mais surtout dans toute situation où il est plus pratique de produire des données d'entraînement par lots.

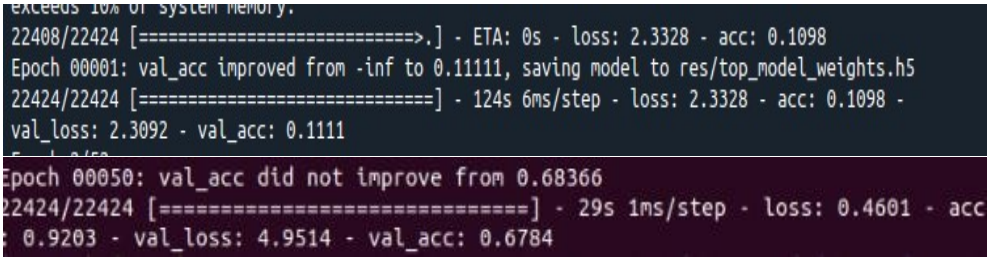


FIGURE 5.5 – **Résultat de l'entraînement du modèle.**

Si nous faisons une comparaison de la (Figure5.5) nous constatons qu'entre ces deux itérations le taux d'erreur baisse tandis que notre précision augmente cela signifie que notre modèle a été bien entraîné et réponds d'ailleurs à la définition du réseau de neurones qui disait que plus le réseau de neurones est profond,meilleur sont ses performances.

5.3.4 Implémentions et résultats expérimentaux

Classe	**Description**
C0	Conduite sûre
C1	Écrire message (main droite)
C2	Parler au téléphone (main droite)
C3	Écrire message (main gauche)
C4	Parler au téléphone (main gauche)
C5	Allumer la radio
C6	Boire
C7	Mettre au diapason
C8	Coiffer et maquiller
C9	Parler au passager(s)

TABLE 5.1 – **Différentes classes de distractions du conducteur.**

Le tableau 5.1 contient les dix activités possibles, elles sont étiquetées de c0 à c9, chacune représentant une activité de conduite distraite spécifique dont l'une est la conduite en toute sécurité. Les autres images appartiennent à des classes où le conducteur peut être considéré comme distrait (par exemple, l'envoi de SMS).
C0 représente la conduite en toute sécurité, tandis que c1 et c8 représentent respectivement l'envoi de SMS de la main droite et la coiffure et/ou le maquillage. Il convient de noter que toutes les images sont généralement prises sous le même angle, dans les mêmes conditions environnementales (c'est-à-dire pendant la journée), ce qui est à la fois bon et mauvais pour la formation du modèle. Cela peut être considéré comme un événement positif car si un sous-ensemble de ces données est utilisé comme ensemble de test alors le modèle aura très probablement de bonnes performances puisque les images de test ont les mêmes conditions. Toutefois, cela pose un problème lorsque le

modèle est généralisé à des images qui ne remplissent pas nécessairement les mêmes conditions que celles de l'ensemble de données original.

5.3.4.1 Phase de test

Dans cette phase, nous allons évaluer notre modèle Vgg16 en faisant appel à quelques fonctions de la bibliothèque matplotlib qui sert à tracer et visualiser des données sous forme de graphiques.La figure 5.6 montre le code de la fonction plot-roc() utilisé pour dessiner les graphes de la phase de test.

```
def plot_roc(y_true, y_pred):

    # calculate roc and auc
    false_pos_rate = dict()
    true_pos_rate = dict()
    roc_auc = dict()

    for i in range(num_classes):
        false_pos_rate[i], true_pos_rate[i], _ = roc_curve(y_true[:,i], y_pred[:,i])
        roc_auc[i] = auc(false_pos_rate[i], true_pos_rate[i])

    # plot all

    cmap = plt.get_cmap('tab10')
    colors = cmap(np.linspace(0, 1, num_classes))

    for i, color in zip(range(num_classes), colors):
        plt.plot(false_pos_rate[i], true_pos_rate[i], lw=2, c=color,
                 label='c{0} (auc = {1:0.2f})'.format(i, roc_auc[i]))

    # plot random guess roc
    plt.plot([0, 1], [0, 1], 'k--',color='salmon', lw=2, label='Random Guess')

    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title("Receiver Operating Characteristic (ROC) Curve")
    plt.legend(loc="lower right", fontsize=8)
    plt.grid()

    plt.show()

# ---------- Avoir les données du test ----------

# create datagen and train generator to load the data from directory
datagen = ImageDataGenerator(rescale=1.0/255.0)
test_generator = datagen.flow_from_directory(
                        '../../../dataset/split_data/test/',
                        target_size=target_size,
                        batch_size=batch_size,
                        class_mode='categorical', # specify categorical
                        shuffle=False) # data is ordered

# load vgg features
test_data = np.load('res/vgg_test_features.npy')

test_labels = test_generator.classes # actual class number
test_labels_onehot = to_categorical(test_labels, num_classes=num_classes) # class number in onehot

# ---------- TEST MODEL ----------
model = create_top_model("softmax", test_data.shape[1:])
model.load_weights("res/_top_model_weights.h5")
```

FIGURE 5.6 – **Code de la partie Test.**

Le modèle VGG semble avoir des frais généraux pour créer les fonctions de goulot d'étranglement, mais comme elles peuvent être stockées hors ligne et chargées à tout moment, la formation ultérieure des couches entièrement connectées prend beaucoup moins de temps. Il semble que ce modèle soit le meilleur jusqu'à présent, car il permet non seulement d'obtenir la plus grande précision, mais aussi de s'entraîner en un minimum de temps. Il est si puissant que son entraînement

peut même être effectué sur un ordinateur portable ordinaire avec un temps d'entraînement très raisonnable.

Analysons plus en détail le modèle en examinant la courbe des caractéristiques de fonctionnement du récepteur (ROC) illustrée à la figure 5.7, en comparant le taux positif réel (TPR) et le taux positif fictif (FPR). Une prédiction optimale aura un TPR de 1 et un FPR de 0, ce qui correspond au coin supérieur gauche du graphique. Une augmentation plus forte et plus rapide dans le coin supérieur gauche indique une solution proche de l'optimum, comme le montre la figure 5.7, une autre caractéristique à noter dans le graphique est sa forte courbure, qui indique une séparation très nette et presque parfaite.

La modification de la précision et de l'entropie croisée catégorielle (appelée la perte) peut être observée dans la figure 5.8, car le modèle est formé à travers les différentes époques. Les précisions de l'entraînement et de la validation augmentent progressivement au fil des époques de la figure 8 jusqu'à ce qu'elles atteignent un certain seuil, indiquant qu'aucune autre amélioration n'est apportée et entraînant la fin de l'entraînement. De même, dans la figure 5.9 la perte de validation et de formation diminue au fur et à mesure que le modèle apprend, jusqu'à ce qu'il se termine finalement lorsque le modèle commence à se sur-ajuster.

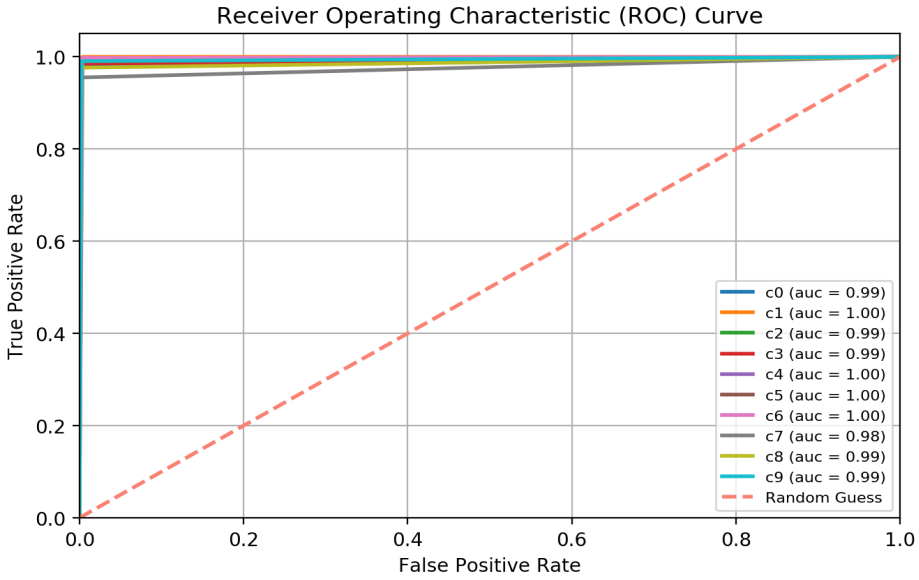


FIGURE 5.7 – **Courbe ROC des prévisions de la distraction du conducteur.**

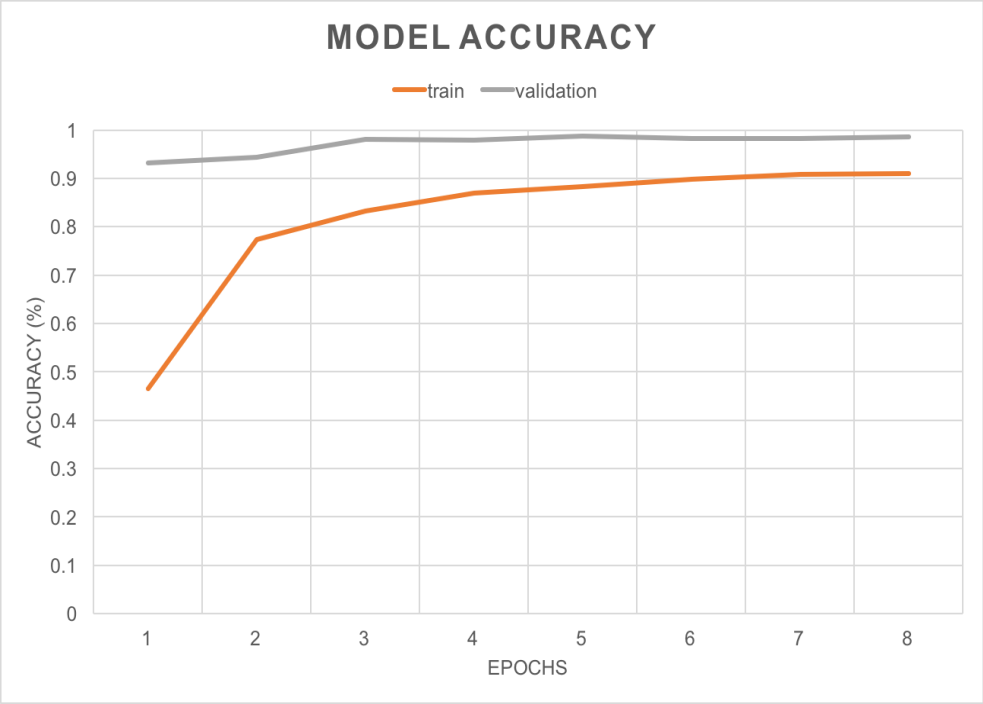


FIGURE 5.8 – **Précision du modèle.**

FIGURE 5.9 – **Modèle de perte.**

Une autre mesure d'évaluation intéressante que nous pouvons voire est la matrice de confusion présentée dans la figure 5.10. La plupart des étiquettes prédites ont une précision proche de 100%. Cependant, c7 se démarque avec une précision de 0,96 qui est relativement faible, par rapport au reste des classes. Cette classe correspond à « Se mettre au diapason », comme le montre le tableau

5.1. La matrice de confusion nous permet de conclure que la prédiction de classe c7 est la plus mal classé ou la plus difficile à prédire. En outre, la matrice de confusion nous permet de constater que c7 est le plus souvent mal étiqueté comme c1, ce qui signifie que regarder au diapason est généralement due à un texte mal étiqueté de la main droite. Cela est compréhensible, puisque les conducteurs qui se mettent derrière effectuent généralement cette action avec la main droite levée. Ainsi, le modèle pensera parfois qu'il parle au téléphone avec la main droite s'il est capturé dans un cadre de mouvement spécifique. Enfin, nous avons pu enregistrer le poids du modèle, afficher des images inédites de conducteurs distraits et afficher les cinq prédictions les plus importantes du modèle. Les résultats d'un échantillon d'image sont présentés à la figure où l'étiquette réelle de c0 : Conduite sûre est prédite comme étant 99,978% d'être un conducteur prudent.

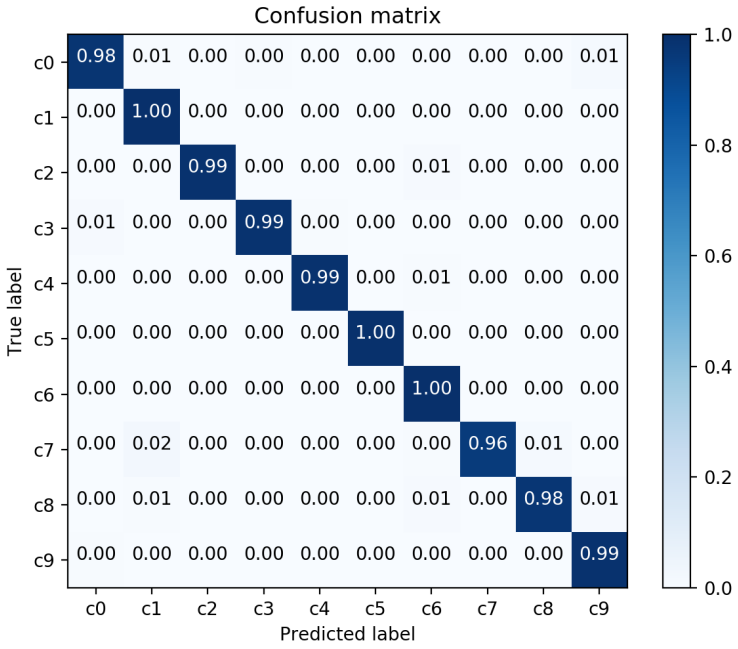


FIGURE 5.10 – **Matrice de confusion du modèle.**

5.3.5 Prédiction du modèle.

```python
from tensorflow.python.keras.preprocessing.image import load_img, img_to_array
from helper import create_top_model, class_labels, target_size
import numpy as np
from tensorflow.python.keras import applications
import operator
import matplotlib.pyplot as plt
import argparse

cap = cv2.VideoCapture('dd.mp4')
if (cap.isOpened()):
    print("Camera OK")
else:
    cap.open()

while (True):
    ret, image = cap.read()

    frame = cv2.resize(image, (224, 224))
    frame = frame[:, :, ::-1]
    image_arr = img_to_array(frame) # convert from PIL Image to NumPy array
    image_arr /= 255

    # to be able to pass it through the network and use batches, we want it with shape (1, 224, 224, 3)
    image_arr = np.expand_dims(image_arr, axis=0)
    # print(image.shape)
    # build the VGG16 network
    model = applications.VGG16(include_top=False, weights='imagenet')

    # get the bottleneck prediction from the pre-trained VGG16 model
    bottleneck_features = model.predict(image_arr)

    # build top model
    model = create_top_model("softmax", bottleneck_features.shape[1:])

    model.load_weights("res/top_model_weights.h5")

    predicted = model.predict(bottleneck_features)
    decoded_predictions = dict(zip(class_labels, predicted[0]))
    decoded_predictions = sorted(decoded_predictions.items(), key=operator.itemgetter(1), reverse=True)

    cv2.putText(image, "Label: {}".format(decoded_predictions[:1][0:]), (10, 35), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 25, 200),
    count = 1
    for key, value in decoded_predictions[:5]:
        print("{}. {}: {:8f}%".format(count, key, value*100))
        count+=1

    cv2.imshow("Classification",image)

    if (cv2.waitKey(1) & 0xFF == ord('q')):
        break;

cap.release()
```

FIGURE 5.11 – **Code de prédiction de notre modèle.**

Pour mieux tester l'efficacité du modèle développé dans ce projet, nous avons créé un fichier python « predict.py » (Figure 5.11) pour faire des prédictions sur des images et un autre fichier«predict2.py» pour faire des prédictions sur des vidéos en temps réel. Nous avons utilisé la bibliothèque OpenCV pour capturer la vidéo, comme nous avons utilisé la bibliothèque PIL pour convertir notre image en tableau numpy. Les résultats obtenus sont illustrés dans les figures ci-dessous.

FIGURE 5.12 – **Prédiction d'une conduite sûre.**

FIGURE 5.13 – **Prédiction sur un conducteur distrait (Parler à un passager).**

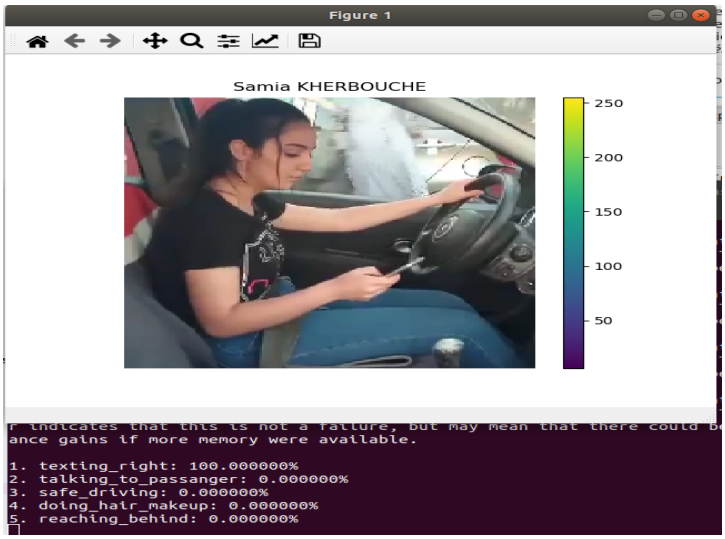


FIGURE 5.14 – **Prédiction sur un conducteur distrait (Écrire un message).**

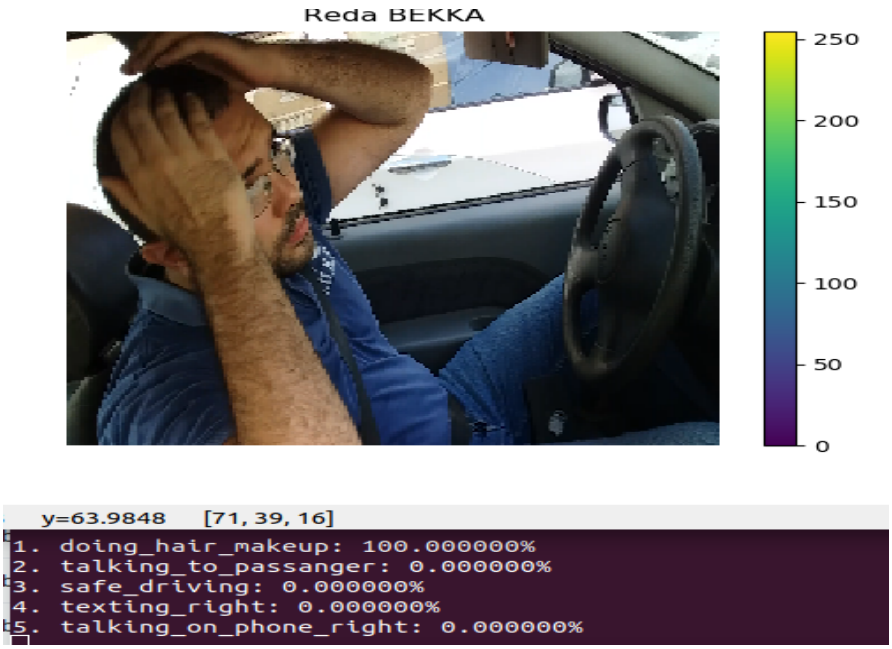


FIGURE 5.15 – **Prédiction sur un conducteur distrait (Se coiffer).**

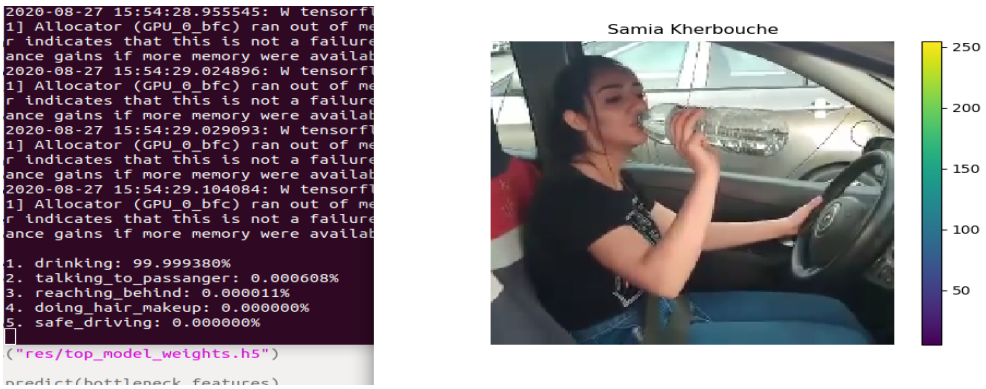


FIGURE 5.16 – **Prédiction sur un conducteur distrait (Boire).**

FIGURE 5.17 – **Prédiction sur un conducteur distrait (Regarder en arrière).**

Nous constatons que les résultats pratiques concordent avec la théorie de l'algorithme VGG16.

5.4 Modèle de détection de somnolence

La détection de somnolence est un sujet qui offre un certain intérêt dans plusieurs projets de recherches. Dans notre cas nous comptons améliorer notre système de détection des distractions en intégrant un modèle de détection de somnolence du conducteur Nous pouvons facilement discerner que le sujet de la détection automatique de somnolence au volant peut regrouper la détection du visage, la détection des yeux, le pourcentage d'ouverture des yeux ...etc.

5.4.1 L'algorithme de détection de somnolence

Le flux général de notre algorithme de détection de la somnolence est assez simple. Tout d'abord, nous allons configurer une caméra qui surveille un flux pour les visages : Si un visage est trouvé, nous appliquons la détection des points de repère du visage et extrayons les régions oculaires.

```
def eye_aspect_ratio(eye):

    A = dist.euclidean(eye[1], eye[5])
    B = dist.euclidean(eye[2], eye[4])

    C = dist.euclidean(eye[0], eye[3])

    ear = (A + B) / (2.0 * C)

    return ear

ap = argparse.ArgumentParser()
ap.add_argument("-p", "--shape-predictor", required=True,
                help="path to facial landmark predictor")
ap.add_argument("-a", "--alarm", type=str, default="",
                help="path alarm .WAV file")
ap.add_argument("-w", "--webcam", type=int, default=0,
                help="index of webcam on system")
args = vars(ap.parse_args())
```

FIGURE 5.18 – **Détection des points de repère de l'œil.**

La valeur de retour du rapport hauteur / largeur des yeux sera approximativement constante lorsque l'œil est ouvert. La valeur diminuera ensuite rapidement vers zéro pendant un clignotement. Dans notre code (Figure 5.18), nous avons utilisé un argument de ligne de commande requis suivi de deux arguments facultatifs : –shape-predictor : Ceci est le chemin vers le détecteur de points de repères faciaux préformé de dlib (détecter et localiser les repères faciaux). –alarm : Nous spécifions le chemin vers un fichier audio d'entrée à utiliser comme alarme. –webcam : Cet entier contrôle l'index de webcam / caméra USB intégrée.
En ce qui concerne le code de l'alarme, nous avons opté pour l'utilisation de la bibliothèque pygame qui facilite le développement de jeux vidéo en temps réel avec le langage de programmation Python.

La figure 5.19 montre la détection des régions oculaires.

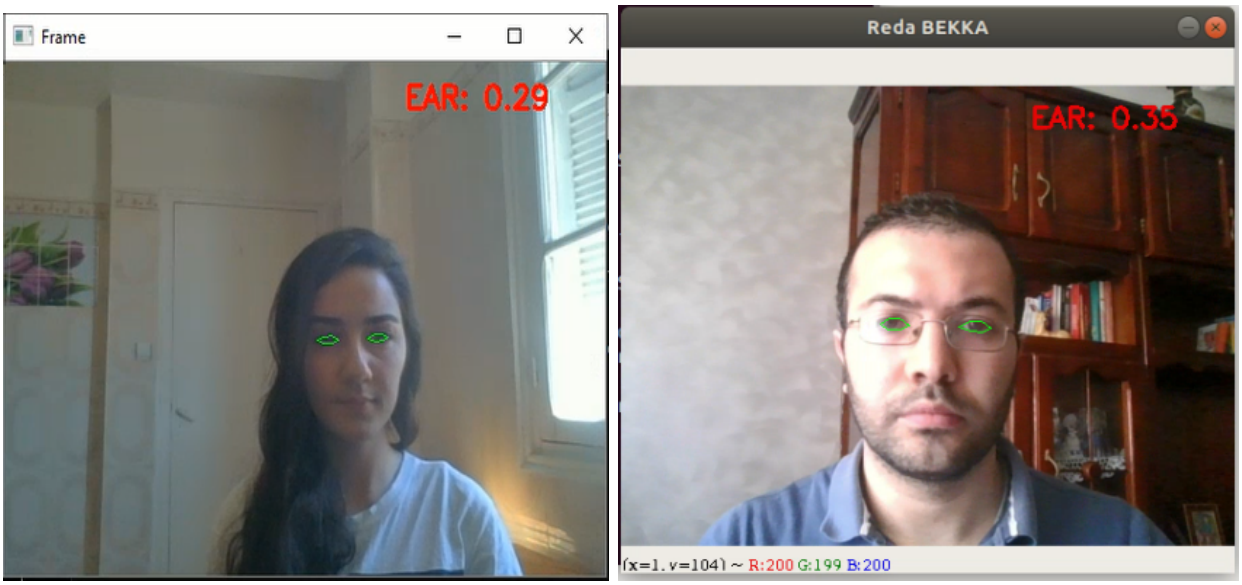


FIGURE 5.19 – **Extraction des régions oculaires du visage.**

```
if ear < EYE_AR_THRESH:
    COUNTER += 1

    if COUNTER >= EYE_AR_CONSEC_FRAMES:

        if not ALARM_ON:
            ALARM_ON = True
            pygame.init()
            pygame.mixer.music.load("alarm.wav")
            pygame.mixer.music.play()
            time.sleep(2)
            pygame.mixer.music.stop()
    cv2.putText(frame, "DROWSINESS ALERT!", (10, 30),
                cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

else:
    COUNTER = 0
    ALARM_ON = False
```

FIGURE 5.20 – **Activation de l'alarme.**

Nous expliquons quelque ligne de notre code illustré dans la figure 5.20 qui nous ont permis d'activer l'alarme : Si COUNTER dépasse EYE_AR_CONSECFRAMES, alors nous supposons que

la personne commence à s'endormir, un autre contrôle est effectué pour voir si l'alarme est activée - si ce n'est pas le cas, nous l'activons.
pygame.init () : - Ceci initialise tous les modules requis pour PyGame.
pygame.mixer.music.load() : Ceci permet de charger la music .
pygame.mixer.music.play() : Ceci permet de démarrer la music.
pygame.mixer.music.stop() : Ceci permet de stopper la music .
Maintenant nous pouvons calculer le rapport hauteur / largeur des yeux pour déterminer si les yeux sont fermés. Dans le cas où le rapport hauteur/largeur des yeux indique que les yeux ont été fermés pendant une durée suffisamment longue, nous déclencherons une alarme pour réveiller le conducteur (Figure 5.21).

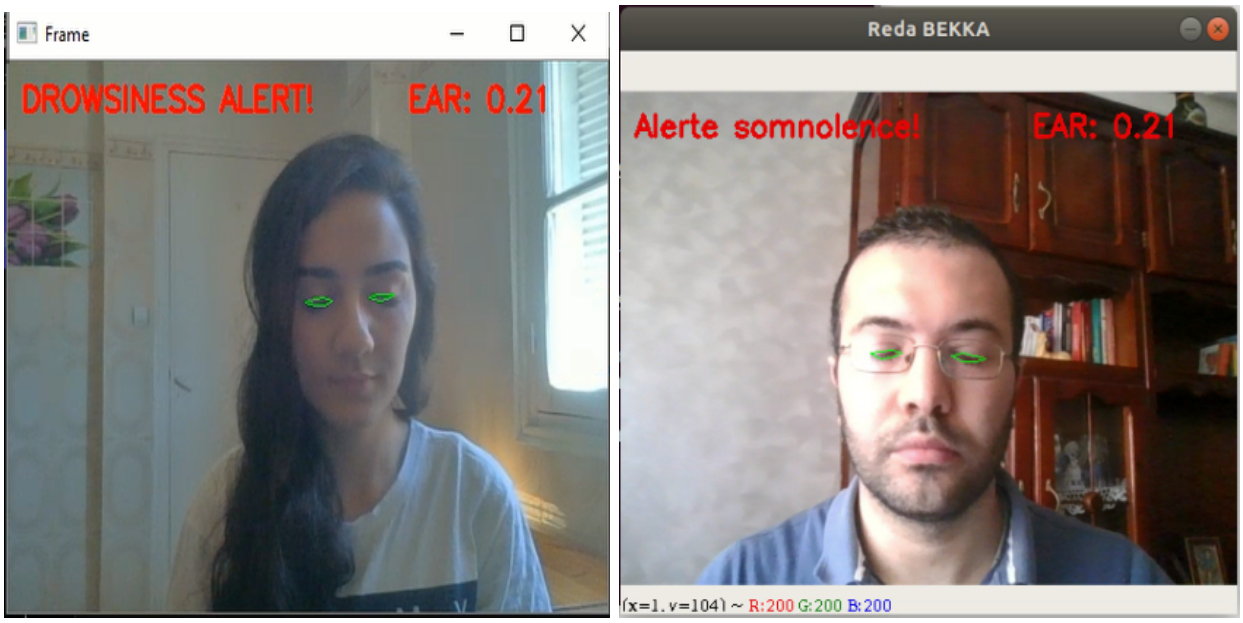


FIGURE 5.21 – **Détection de somnolence.**

Dans notre modèle de détection de somnolence, nous constatons qu'il est capable de détecter la moindre somnolence, puis de faire déclencher une alarme sonore pour attirer l'attention du conducteur.
Le détecteur de somnolence est même capable de fonctionner dans diverses conditions, y compris la lumière directe du soleil lors de la conduite sur route et un éclairage faible / artificiel ou même l'obscurité totale.
La figure 5.22 suivante montre la détection d'un conducteur hypovigilant dans un garage sans lumière.

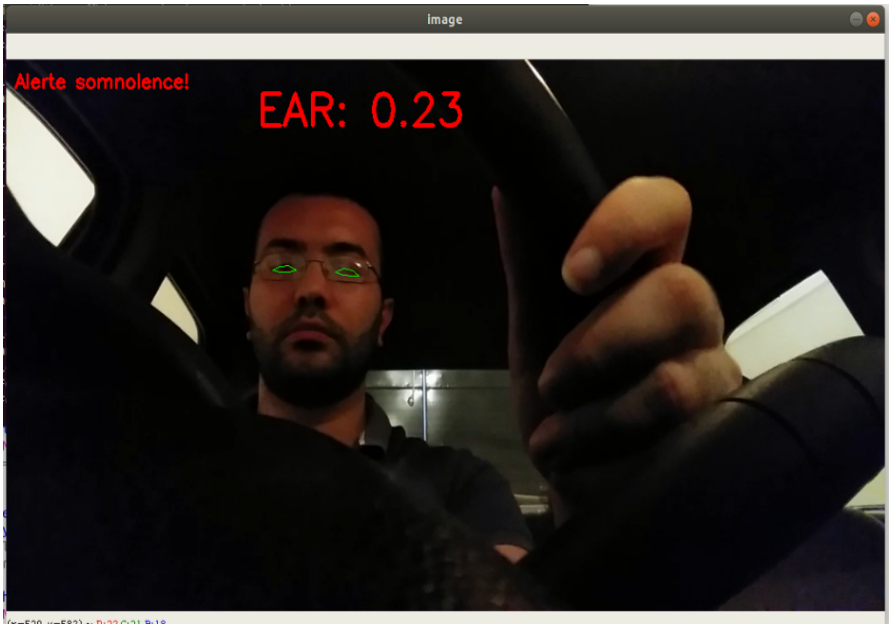


FIGURE 5.22 – **Détection d'un conducteur en état de somnolence dans des lieux sombres.**

La figure 5.23 montre la détection des régions oculaires d'un conducteur en plein jour.

FIGURE 5.23 – **Détection des régions oculaires d'un conducteur en plein jour.**

5.5 Choix de la carte d'acquisition

Pour la carte d'acquisition on a choisi la carte Raspberry Pi (Figure 5.24) cette dernière est un petit ordinateur de la taille d'une carte de crédit, qui peut facilement être connecté à Internet et servir d'interface à de nombreux composants électroniques. En effet Malgré sa petite taille, celle-ci est aussi puissante qu'un smartphone. Raspberry Pi3, comprend un processeur 4 cœurs ainsi que 1GB de RAM. On peut y installer un vrai système d'exploitation, comme par exemple Raspbian, Ubuntu ou Windows. Aussi une carte Raspberry Pi n'est pas seulement un ordinateur. Avec 40 pins GPIO, on peut facilement connecter notre carte avec de nombreux capteurs et composants électroniques. De plus tout le monde peut utiliser une carte Raspberry Pi. Tout ce que nous avons à faire, c'est de télé charger un système d'exploitation, de l'écrire sur une carte microSD, de connecter notre Raspberry Pi à un écran. En fin le plus important c'est que cette carte peut facilement être connecté à Internet et servir d'interface à de nombreux composants électroniques comme dans notre cas la caméra, le module WIFI.
Nous avons télé-chargé et installé le système d'exploitation à l'aide de NOOBS (New Out Of the Box Software),qui est un gestionnaire de système d'exploitation pour télé-charger et installer facilement des systèmes d'exploitation sur Raspberry Pi. Le système d'exploitation officiel Raspberry Pi s'appelle Raspbian qui est une version de Linux conçue spécifiquement pour Raspberry Pi. Il est livré avec NOOBS. NOOBS prend également en charge une liste de systèmes d'exploitation parmi lesquels choisi.

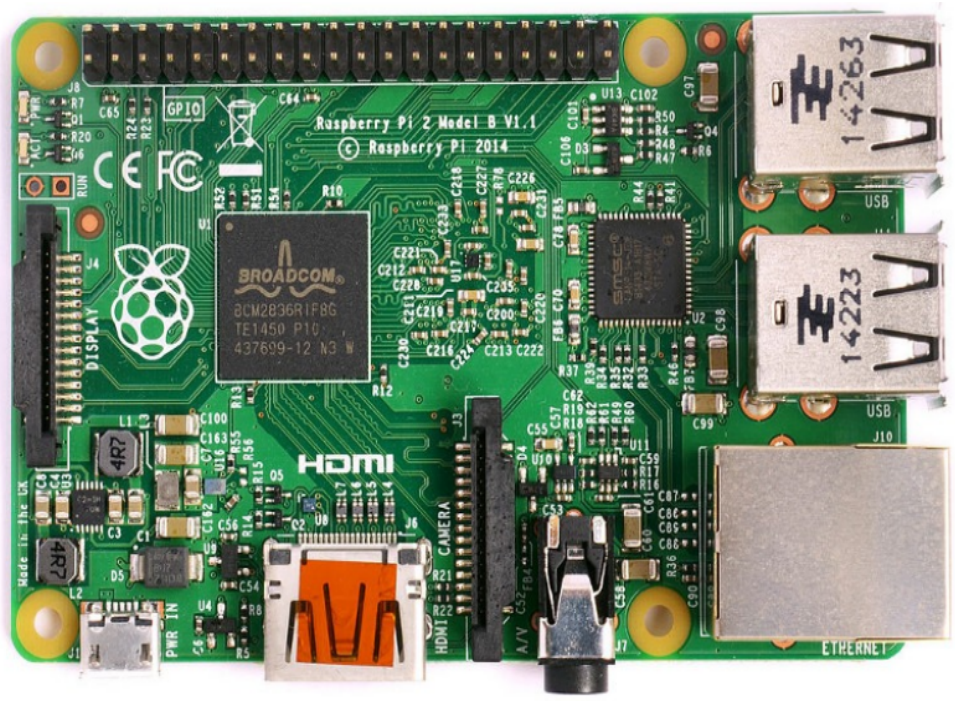


FIGURE 5.24 – **vue de dessus d'une carte Raspberry Pi.**

5.5.1 Configuration de l'environnement

Afin d'utiliser les techniques et différentes architectures, il est préférable d'utiliser «Coral»,une carte de développement ultra-rapide , «sans Internet requise» et un accélérateur USB lancé par Google.Coral TPU(Tensor Processing Unit) est suffisamment rapide pour effectuer des inférences directement sur l'appareil plutôt que d'envoyer l'image / le cadre au cloud pour inférence et prédiction.Cependant nous allons nous contenter d'installer les divers bibliothèques nécessaires pour notre modèle. La première chose est d'installer OpenCV sur un Raspberry Pi 3 avec toutes les dépendances.Nous devons également installer un module python appelé picamera [array]. Le module fournit une interface pour représenter les images de la caméra sous forme de tableaux NumPy.En deuxième lieu,nous installons tensorflow et keras comme le montre la figure 5.25

```
$ wget https://github.com/samjabrahams/tensorflow-on-raspberry-pi/releases/download/v1.1. 0
/tensorflow-1.1. 0 -cp27-none-linux_armv7l.whl
$ pip installe tensorflow-1.1. 0 -cp27-none-linux_armv7l.whl
```

FIGURE 5.25 – **Installation de tensorflow sur Raspberry Pi.**

Après l'installation de tensorflow,nous avons intégré a notre mini ordinateur HDF5 + h5py.Ces bibliothèques qui permettent de charger notre modèle pré-entraîné à partir du disque.Pour ce qui du modèle de détection de somnolence,nous aurons juste a importer notre code de ubuntu vers le raspberry en effectuant quelque modifications tels que :

- **Traffic Hat** : Ajout d'un module complémentaire qui est le Traffic Hat qui joue le rôle de haut parleurs permettant de déclencher l'alarme bruyante pour réveiller le conducteur..
- **RPi.GPIO et gpiozero** :
  Nous avons installé ces 2 bibliothèques en utilisant le gestionnaire de paquets pip afin de lier traffic hat a notre mini-ordinateur comme le montre la figure 5.26 .

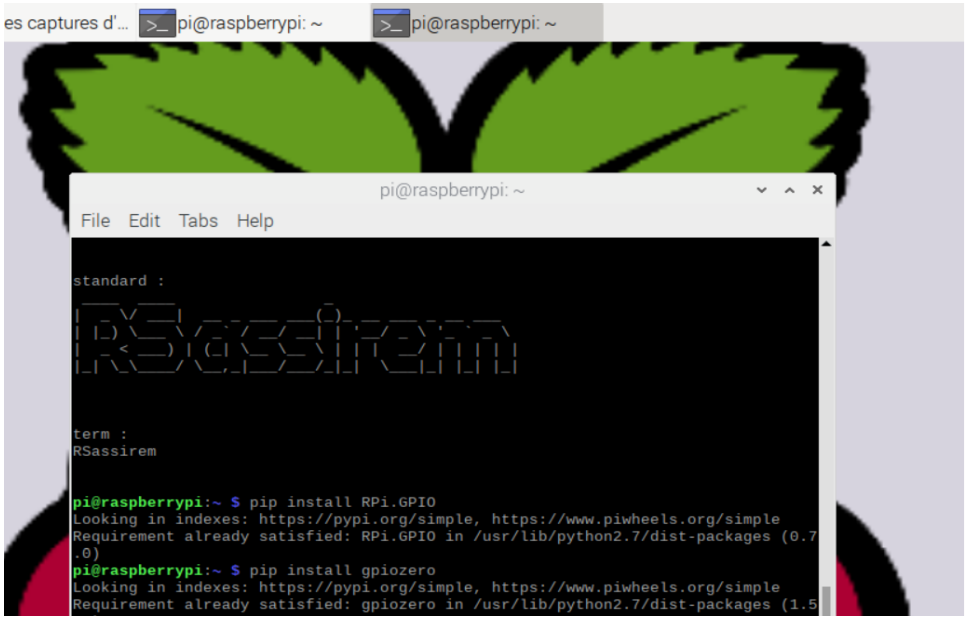


FIGURE 5.26 – **Installation des packages GPIO.**

5.6 Présentation du logiciel (RSassirem)

RSassirem est un système de détection de distractions des conducteurs de transport public en temps réel se constituant d'un logiciel bureau conçu en utilisant la bibliothèque de python la mieux placé pour créer des interfaces graphiques PyQt5, d'une partie électronique (Raspberry pi) et d'un modèle deep learning (VGG16).
PyQt5 est considéré comme la meilleure bibliothèque python pour la création de logiciel bureau en offrant la possibilité de modifier le CSS des interfaces, il permet de gérer le code python d'interfaces graphiques en utilisant l'utilitaire graphique QtDesigner, comme il est multiplateforme.

5.6.1 Présentation des interfaces du logiciel

5.6.1.1 Interface d'authentification

Afin d'exploiter pleinement les fonctionnalités de RSassirem, pour chaque administrateur il est nécessaire de passer par un système d'authentification (Figure 5.27).
En effet l'authentification consiste à introduire l'émail et le mot de passe de l'administrateur, une fois ces informations sont reconnues le système affiche la page appropriée. Dans le cas contraire le système affiche une indication d'erreur.

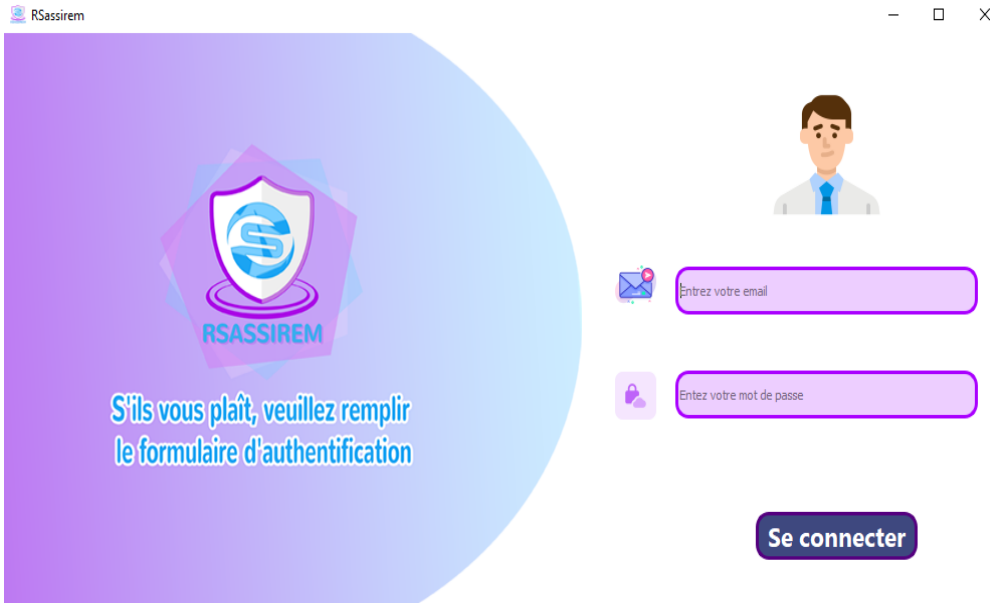


FIGURE 5.27 – **Page d'authentification.**

5.6.1.2 Interface du rapport journalier

C'est la première interface qui s'affiche après l'authentification, ou les administrateurs peuvent faire un rapport journalier sur les conducteurs distraits, comme ils peuvent exporter les données vers un fichier Excel (Figure 5.28).

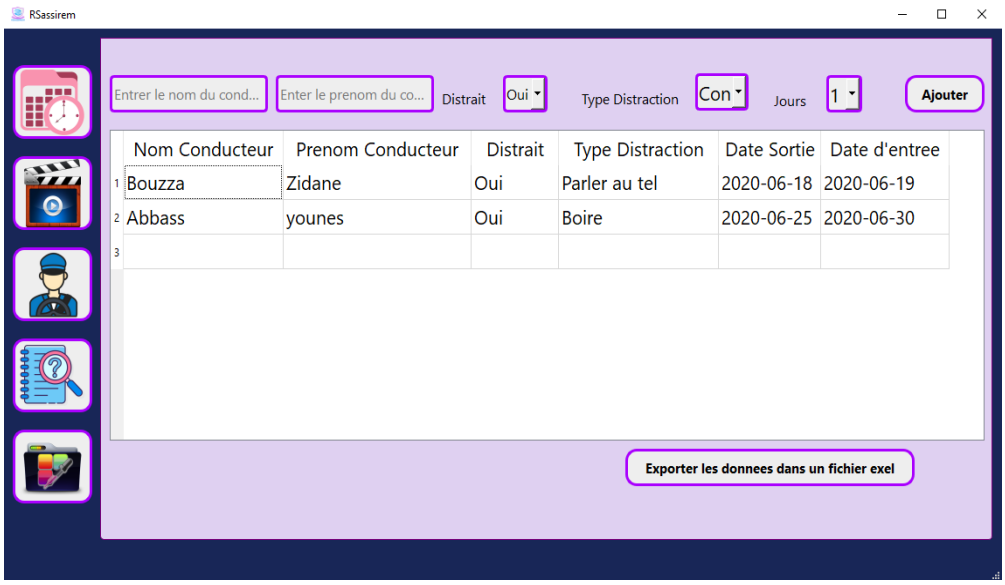


FIGURE 5.28 – **Interface du rapport journalier.**

5.6.1.3 Interface vidéo-surveillance

Cette interface (Figure 5.29) permet de surveiller les conducteurs en temps réel, ainsi afficher sur les vidéos le type de distraction. En outre l'administrateur peut enregistrer la vidéo dans un dossier sur sa machine. Nous avons intégré notre algorithme de détection des distractions avec notre logiciel en liant le fichier de prédiction « predict.py » avec la partie opencv de notre logiciel.

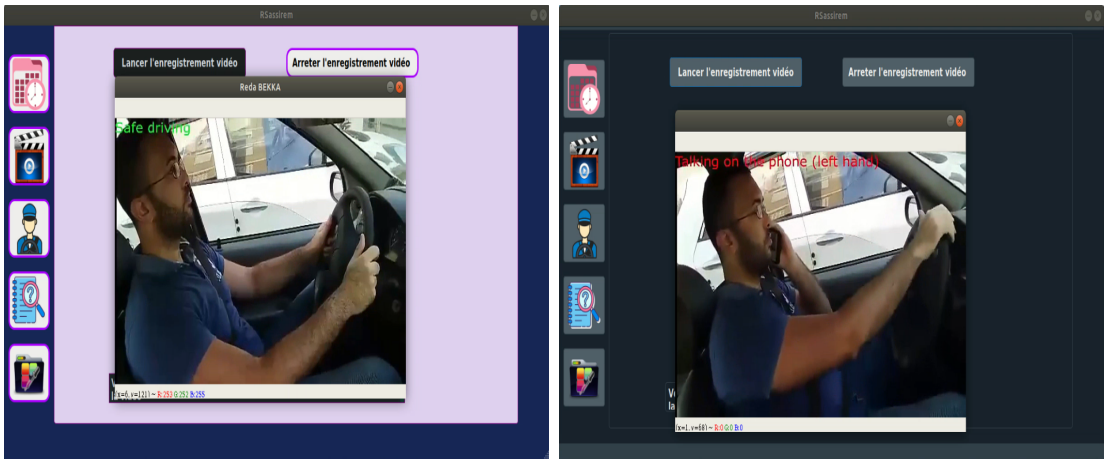


FIGURE 5.29 – **Interface vidéo-surveillance.**

5.6.1.4 Interface mise à jour

Les administrateurs peuvent effectuer les opérations de mise à jour entre autres : Ajout, modification, suppression et la recherche d'un conducteur selon son identifiant. Ses données seront affichées sur un tableau comme le montre la figure 5.30.

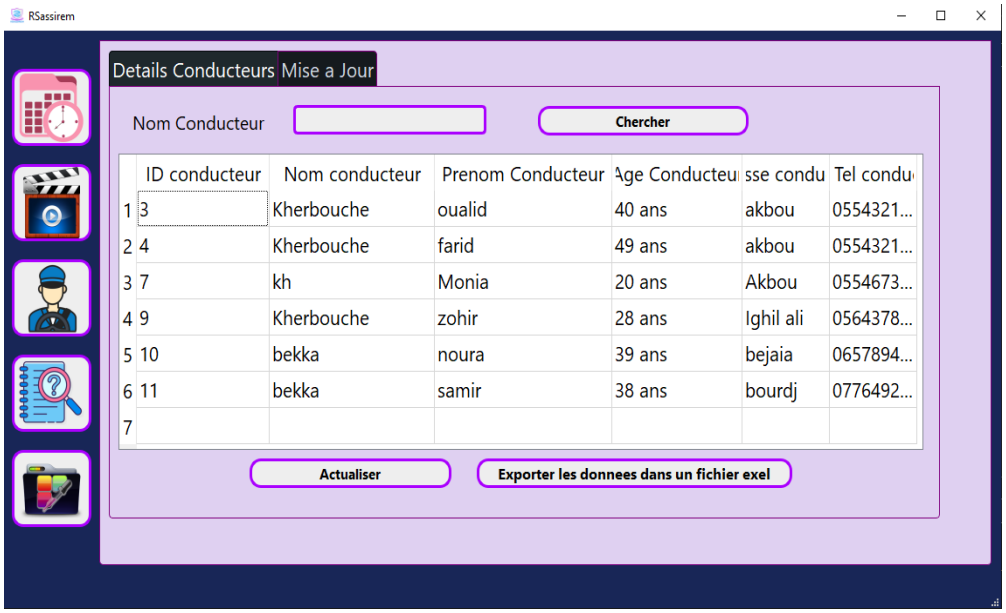


FIGURE 5.30 – **Interface détails conducteurs.**

5.6.1.5 Interface aide

Nous avons conçu une interface de guide pour l'administrateur (Figure 5.31) afin de lui faciliter la manipulation du logiciel, cette interface contient 2 cards : l'une est pour présenter le projet et l'autre pour introduire le logiciel.

FIGURE 5.31 – **Interface Aide.**

Remarque :

Lorsque on clique sur les boutons « présentation du projet » et « présentation du logiciel », une

autre fenêtre apparaît, comme le montre les figures 5.32 et 5.33.

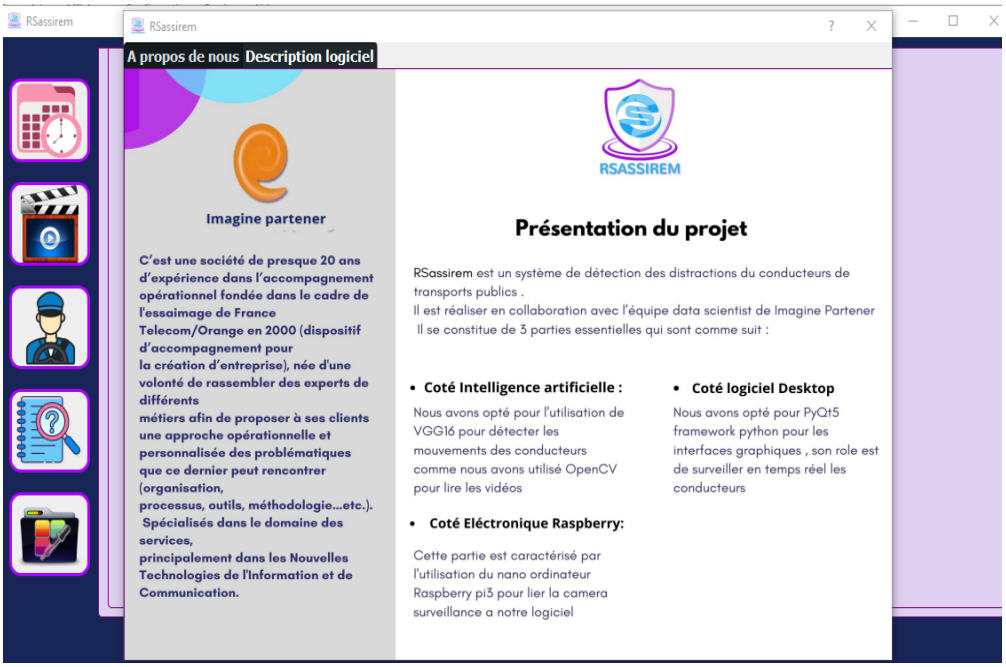


FIGURE 5.32 – **Présentation du projet.**

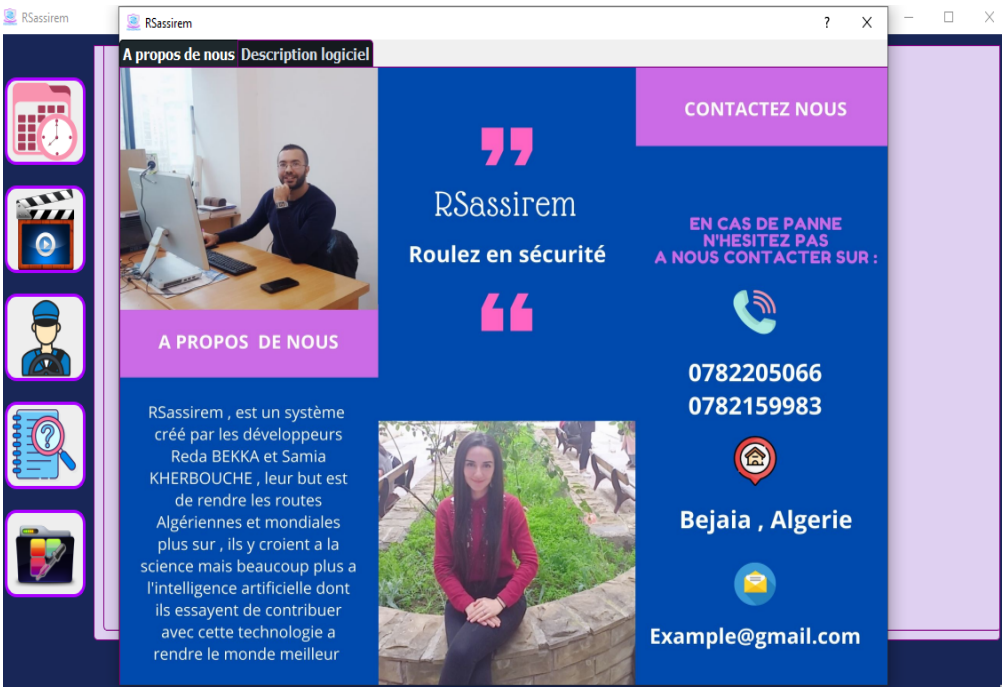


FIGURE 5.33 – **A propos de nous.**

Nous avons conçu une fenêtre qui aura pour rôle de guider l'utilisateur de RSassirem comme

par exemple sur quel bouton clique pour visualiser ou arrêter la vidéo (Figure 5.34).

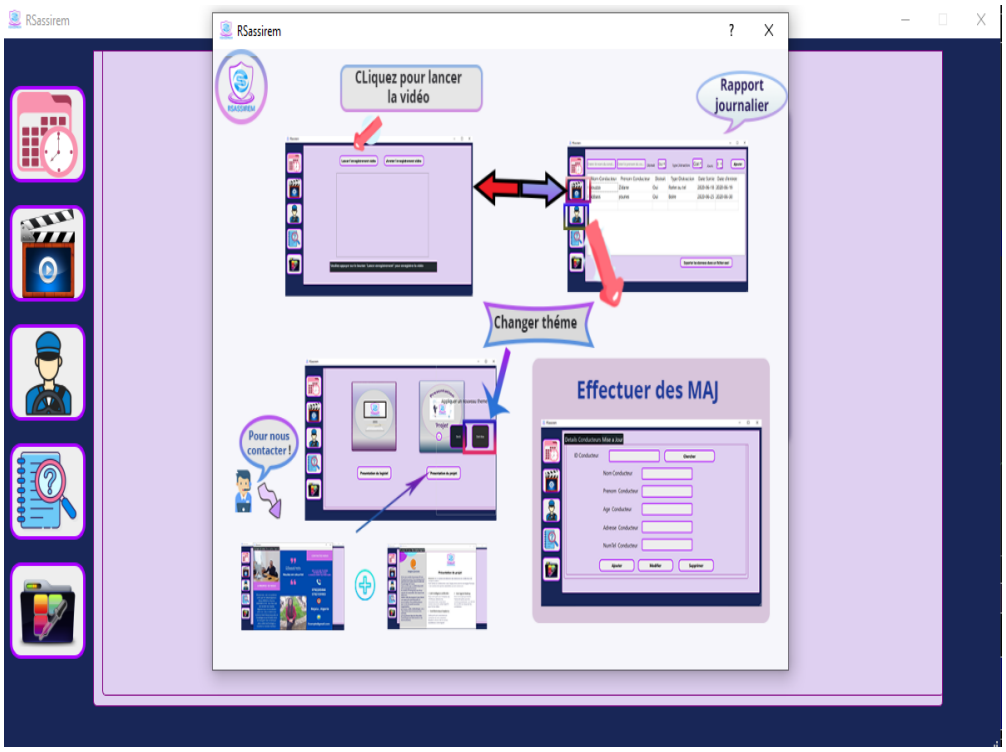


FIGURE 5.34 – **Interface guide de l'administrateur.**

5.7 Conclusion

Au cours de ce dernier chapitre, nous avons présenté notre système de détection des distractions du conducteur de transport public en temps réel basée sur les réseaux de neurones de type CNN, nous avons utilisé l'architecture VGG16 et notre système a été testé en utilisant les données de Kaggle, en outre nous avons présenté les outils nécessaires pour la réalisation de notre travail ainsi que l'environnement de travail.
Pour rendre notre système plus fiable, nous l'avons renforcé en ajoutant un système de détection de somnolence au volant ainsi nous pensons que les résultats obtenus sont satisfaisants par rapport à d'autres architectures dont la sensibilité n'est pas très efficace.
La conclusion de ce chapitre est réservée à la présentation de notre logiciel tout en s'appuyant sur quelques captures d'écran qui expliquent le déroulement et le fonctionnement de notre travail.

Conclusion générale

Depuis sa naissance en 1956 et jusqu'à ce jour, l'intelligence artificielle a connu une évolution phénoménale, les principaux progrès étant effectués dans tous les domaines, en particulier, la sécurité routière. Aujourd'hui, bien que les ordinateurs surpassent les hommes par leur rapidité, ils sont néanmoins loin d'égaler les capacités de réflexion de ces derniers ! C'est là une des principales limites technologiques car la complexité du cerveau humain rend impossible toute reproduction artificielle. L'ordinateur tel que nous l'avons conçu n'est rien de plus qu'un outil à notre service. Mais l'idée de remplacer l'homme dans le futur pour certaines tâches n'est pas si mal, aujourd'hui grâce au progrès de l'intelligence artificielle et au développement au moyen du calcul rapide ce rêve s'approche de la réalité. Ce qui nous a motivé à étudier ce sujet intéressant.
Notre objectif est en fait de présenter un nouveau modèle de détection des distractions du conducteur de transports public en temps réel afin de diminuer le nombre d'accidents et de décès. Pour cela, nous avons utilisé un ensemble d'outils algorithmiques et logiciels, ainsi que des techniques d'intelligence artificielle. Ils nous ont fourni de plus, une source pour établir les liens entre la théorie et la pratique. De ce fait, nous avons essayé de développer un modèle plus robuste par rapport aux modèles que nous avons étudié dans le chapitre 3 ainsi d'offrir d'autres alternatives qui rendront notre système plus fiable pour effectuer sa mission comme le système de détection de somnolence.
Notre système de détection de distraction est basé sur le Vgg16 pour la détection des différents mouvements séparé en 10 classes. La raison de ce choix est double. D'une part, le modèle exploité est imposé par l'entreprise d'accueil, d'autre part vgg16 surpasse considérablement la génération précédente de modèles dans les compétitions ILSVRC-2012 et ILSVRC-2013 et il représente un excellent élément constitutif à des fins d'apprentissage car il est facile à mettre en œuvre.
Pour ce qui est de notre modèle de détection de somnolence, nous avons utilisé OpenCv qui est une bibliothèque requise pour les manipulations d'affichages et d'images ainsi que pour la détection de visage comme nous avons utilisé Dlib pour détecter et localiser les repères faciaux.
Ce mémoire de fin d'études nous a permis de :

- Approfondir nos connaissances théoriques et pratique déjà acquises, maîtriser les nouvelles techniques et compléter notre initiale pour atteindre ainsi un niveau de perfection supérieur et de pouvoir apprendre d'autres nouveautés,

- Construire des savoirs et des savoir-faire dans le domaine de vision par ordinateur plus précisément la détection de mouvements qui est un thème d'actualité,
  .

- Comprendre les différentes étapes pour créer l'algorithme de Deep Learning en python,
- Acquérir le savoir nécessaire dans le domaine électronique plus exactement comprendre la liaison entre notre modelé et le Raspberry pi.

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

- Il serait intéressant de travailler avec une base d'images plus large et plus variante,
- Construire une reconnaissance faciale qui fonctionnera avec le modèle de détection de somnolence et aura pour but de faciliter la reconnaissance des conducteurs de transport public. Pour construire ce modèle nous ferons appel aux trois algorithmes PCA (Principle Component Analysis), LDA (Linear Discriminant analysis) et comme base de donnée nous utiliserons yalesfaces,LBP(Local binary patterns),
- Enfin, nous renforcerons notre système en ajoutant la technologie de semi-autonome aux véhicules (classification et détection des symboles routiers ainsi la détection lignes de voie‘) et pour effectuer cette tache nous allons utiliser le réseau neuronal leNet et pour ce qui est du dataset nous utiliserons 'german-traffic-sign' qui est accessible via bitbucket et qui se constitue de 42 classes.

Bibliographie

[1] « Le phénomène des accidents de la route, à qui la faute ? », Echoroukonline, 21-janv-2020. [En ligne]. Disponible sur : https ://www.echoroukonline.com/le-phenomene-des-accidents-de-la-route-a-qui-la-faute/. [Consulté le : 01-avr-2020].

[2] « Accident de la route : L'Algérie à la 98ème place au classement mondial ». [En ligne]. Disponible sur : http ://www.transactiondalgerie.com/index.php/actualite/327-accident-de-la-route-l-algerie-a-la-98eme-place-au-classement-mondial. [Consulté le : 01-avr-2020].

[3] « Accidents de la route pour l'année 2019 : Alger en tête du classement national | El Watan ». https ://www.elwatan.com/regions/centre/alger/accidents-de-la-route-pour-lannee-2019-alger-en-tete-du-classement-national-26-12-2019 (consulté le mars 31, 2020).

[4] « Statistiques sur les accidents de le route (mortalité, contexte etc.) - ... » http ://www.fiches-auto.fr/articles-auto/l-auto-en-chiffres/s-582-statistiques-sur-les-accidents-de-le-route-mortalite-contexte-etc.php (consulté le mars 31, 2020).

[5] « Définition de l'Intelligence Artificielle ». http ://tpe-intelligence-artificielle-2013.e-monsite.com/pages/definition-de-l-intelligence-artificielle.html (consulté le mars 31, 2020).

[6] G. Saint-Cirgue, « Introduction au Machine Learning - Machine Learnia ». https ://machinelearnia.com/machine-learning-introduction/ (consulté le mars 31, 2020).

[7] +Bastien L, « Deep Learning ou apprentissage profond : définition, concept », LeBigData.fr, juill. 16, 2018. https ://www.lebigdata.fr/deep-learning-definition (consulté le avr. 01, 2020).

[8] P. C. Jorat, « Comprendre le machine learning et le deep learning », Bial-R, mai 22, 2019. https ://www.bial-r.com/2019/05/22/comprendre-le-machine-learning-et-le-deep-learning/ (consulté le avr. 01, 2020).

[9] O. Kharkovyna, « An Intro to Deep Learning for Face Recognition », Medium, juin 26, 2019. https ://towardsdatascience.com/an-intro-to-deep-learning-for-face-recognition-aa8dfbbc51fb (consulté le avr. 01, 2020).

[10] « L'utilisation des réseaux de neurones en métallurgie - MetalBlog ». https ://metalblog.ctif.com/2019/02/04/lutilisation-des-reseaux-de-neurones-en-metallurgie/ (consulté le avr. 01, 2020).

[11] « Définition | Deep Learning - Apprentissage profond | Futura Tech ». https ://www.futura-sciences.com/tech/definitions/intelligence-artificielle-deep-learning-17262/ (consulté le avr. 01, 2020).

[12] admin, « Deep learning Machine learning : what's the difference ? », Parsers, mai 20, 2019. https ://parsers.me/deep-learning-machine-learning-whats-the-difference/ (consulté le avr. 01, 2020).

[13] khaled kadouddour, lezeregue abdelwadoud, « Developpement d'une application de detection somnolence au volant du conducteur pour eviter les accidents de la route », universite Saad Dahlab - Blida 1 (USDB), memoire de fin d'etude, 2019 2018.

[14] M. Mansouri, « Intelligence artificielle et détection humaine dans les contenus audiovisuels : état de l'art, application à la publicité et éléments méthodologiques », Le Blog de l'ARPP, mars 07, 2019. https ://blog.arpp.org/2019/03/07/intelligence-artificielle-et-detection-humaine-dans-les-contenus-audiovisuels-etat-de-lart-application-a-la-publicite-et-elements-methodologiques/ (consulté le avr. 02, 2020).

[15] A. Submission, « Rapport pour le projet d'Apprentissage Avancé : Fish Conservancy Challenge », p. 8.

[16] K. Vikram et S. Padmavathi, « Facial parts detection using Viola Jones algorithm », in 2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS), 2017, p. 1–4.

[17] A. Parande, « Understanding and Implementing Viola-Jones (Part Two) », Medium, janv. 27, 2019. https ://medium.com/datadriveninvestor/understanding-and-implementing-viola-jones-part-two-97ae164ee60f (consulté le avr. 04, 2020).

[18] W. Ian et Fernandez John, « Facial feature detection using Haar classifiers », J. Comput. Sci. Coll., avr. 2006, Consulté le : avr. 04, 2020. [En ligne]. Disponible sur : https ://dl.acm.org/doi/abs/10.5555/1127389.1127416.

[19] Accessed : 2019-06-20 Adam Geitgey, www.linkedin.com/learning/deep- learning-face-recognition. Deep learning : Face recognition.

[20] « Object Detection with 10 lines of code - Towards Data Science ». https ://towardsdatascience.com/object-detection-with-10-lines-of-code-d6cb4d86f606 (consulté le avr. 05, 2020).

[21] R. Huang, J. Pedoeem, et C. Chen, « YOLO-LITE : a real-time object detection algorithm optimized for non-GPU computers », in 2018 IEEE International Conference on Big Data (Big Data), 2018, p. 2503–2510.

[22] M. Wu, « Real time vehicle detection using YOLO », Medium, août 20, 2017. https ://medium.com/@xslittlegrass/almost-real-time-vehicle-detection-using-yolo-da0f016b43de (consulté le avr. 05, 2020).

[23] V. K. Jonnalagadda, « Object Detection YOLO v1, v2, v3 », Medium, janv. 31, 2019. https ://medium.com/@venkatakrishna.jonnalagadda/object-detection-yolo-v1-v2-v3-c3d5eca2312a (consulté le avr. 05, 2020).

[24] J. Redmon et A. Farhadi, « Yolov3 : An incremental improvement », ArXiv Prepr. ArXiv180402767, 2018.

[25] D. Murray et A. Basu, « Motion tracking with an active camera », IEEE Trans. Pattern Anal. Mach. Intell., vol. 16, no 5, p. 449-459, mai 1994, doi : 10.1109/34.291452.

[26] « VGG16 - Convolutional Network for Classification and Detection », nov. 20, 2018. https ://neurohive.io/en/popular-networks/vgg16/ (consulté le avr. 06, 2020).

[27] L. Pal, « Image classification : A comparison of DNN, CNN and Transfer Learning approach », Medium, sept. 13, 2019. https ://medium.com/analytics-vidhya/image-classification-a-comparison-of-dnn-cnn-and-transfer-learning-approach-704535beca25 (consulté le avr. 06, 2020).

[28] M. Jabon, J. Bailenson, E. Pontikakis, L. Takayama, et C. Nass, « Facial expression analysis for predicting unsafe driving behavior », IEEE Pervasive Computing, vol. 10, no 4, p. 84–95, 2010.

[29] M.-H. Sigari, M. Fathy, et M. Soryani, « A driver face monitoring system for fatigue and distraction detection », Int. J. Veh. Technol., vol. 2013, 2013.

[30] L. Fu, W. Wu, Y. Zhang, et R. Klette, « Unusual motion detection for vision-based driver assistance », Int. J. Fuzzy Log. Intell. Syst., vol. 15, no 1, p. 27–34, 2015.

[31] D. Tran, H. M. Do, W. Sheng, H. Bai, et G. Chowdhary, « Real-time detection of distracted driving based on deep learning », IET Intelligent Transport Systems, vol. 12, no 10, p. 1210–1219, 2018.

[32] M. Jeong et B. C. Ko, « Driver's facial expression recognition in real-time for safe driving », Sensors, vol. 18, no 12, p. 4270, 2018.

[33] T. Wilhelm, « Towards Facial Expression Analysis in a Driver Assistance System », in 2019 14th IEEE International Conference on Automatic Face Gesture Recognition (FG 2019), 2019, p. 1–4.

[34] M. Alotaibi et B. Alotaibi, « Distracted driver classification using deep learning », Signal, Image and Video Processing, p. 1–8, 2019.

[35] Y. LeCun, « Deep learning convolutional networks », in 27th IEEE Hot Chips Symposium, HCS 2015, 2016, p. 7477328.

[36] « Snapshot ». Consulté le : juin 19, 2020. [En ligne]. Disponible sur :https ://developers.google.com/machine-learning/practica/image-classification/convolutional-neural-networks.

[37] S. Lawrence et C. L. Giles, « Overfitting and neural networks : conjugate gradient and backpropagation », in Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing : New Challenges and Perspectives for the New Millennium, 2000, vol. 1, p. 114–119.

[38] Y. Gultom, A. M. Arymurthy, et R. J. Masikome, « Batik classification using deep convolutional network transfer learning », Jurnal Ilmu Komputer dan Informasi, vol. 11, no 2, p. 59–66, 2018.

[39] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, et Y. Bengio, « Identifying and attacking the saddle point problem in high-dimensional non-convex optimization », in Advances in neural information processing systems, 2014, p. 2933–2941.

[40] G. Hinton, N. Srivastava, et K. Swersky, « Neural networks for machine learning lecture 6a overview of mini-batch gradient descent », Cited on, vol. 14, no 8, 2012.

RÉSUMÉ

La sécurité routière est un enjeu majeur, tant en termes de nombre de victimes sur les routes que du coût économique de ces accidents au niveau mondial, régional et national. La lutte contre l'insécurité routière est une préoccupation prioritaire pour chaque pays, étant donné que les déplacements ne cessent d'augmenter, et malgré les mesures prises dans de nombreux pays pour améliorer la sécurité routière, il reste beaucoup à faire pour réduire le nombre de décès et d'accidents. Dans ce mémoire, nous passons en revue les approches les plus appliquées en matière de détection des distractions dans l'assistance au conducteur et nous présentons une nouvelle approche de prévention des accidents de la route, en nous appuyant sur les approches existantes avec des améliorations clés pour augmenter le taux de détection des différentes distractions. Les expériences préliminaires indiquent que l'approche proposée a conduit à une efficacité et une précision très élevées pour détecter la somnolence et les distractions du conducteur de véhicule de transport public.

Mots clés : Apprentissage par transfert,Cnn, Détection de distractions, détection de somnolence, Intelligence artificielle, Opencv, Vgg16.

ABSTRACT

Road safety is a major issue, both in terms of the number of victims on the roads and the economic cost of these accidents at the global, regional and national levels. Combating road insecurity is a priority concern for every country, as travel continues to increase, and despite the measures taken in many countries to improve road safety, much remains to be done to reduce the number of deaths and accidents. In this brief, we review the most applied approaches to distraction detection in driver assistance and present a new approach to road crash prevention, building on existing approaches with key improvements to increase the detection rate of different distractions. Preliminary experiments indicate that the proposed approach has led to very high efficiency and accuracy in detecting drowsiness and distractions of the driver of public transport vehicles.

Key words : Artificial intelligence, Cnn, Distraction detection, Drowsiness detection, Opencv,Transfer learning, Vgg16.

AGZUL

Taellist n yiberdan(tamsebridt) d tadwilt (asrus) meqqren deg wayen yerzan aman n titas deg yiberdan di tama tadamsant id-yessegran isahwa deg uswir agralan , amnaan , aelnaw . Yessefk ad yili uareb (umennu) mgal-is deg yal tamurt , imi amussu ur yebis ara deg walluy , as akken ttwaddmen yiezziben deg tuget n tmura i usselhu n tellist timsebridt, maca yeqqim aas n temsal akken ad issenqes deg uman n wid yettmettaten deg yiberdan . Deg ukatay-a (takatut) , ad nual er wayen nettekk deg tarrayin yettwasensen n usukkes n tucciwin aya s tallelt n umaway , yessisin-a-d tulmist tamaynut i uuddu d uezzeb deg ubrid. Ad nefer tulmisin yellan yakkan akked usselhu agejdan akken ad nessali tazmilt n usukkes s umgarad n tmazlalt . Tirmitin n yinzewren mmlent-d tulmist n tnehrawt tesa tamilalt d tiseddi meqqren s waas i uskan n usnuddem d tmazlalt « tucciwin» n umaway amesni anagdud .

Awalen-tisura : Asukkes n tmazlalt « tucciwin», Asukkes n usnuddem, Cnn, Opencv, Tigzi timelit, Vgg16.