République Algérienne Démocratique et Populaire
Ministère de l'enseignement Supérieur et de la Recherche Scientifique
Université de Mohamed El Bachir El Ibrahimi de Bordj Bou Arréridj
Faculté des mathématiques et de l'informatique
Département d'informatique



MÉMOIRE

Présenté en vue de l'obtention du diplôme

Master en Informatique

Spécialité : *Ingénerie de l'informatique décisionnelle*

THÈME

Système de détection de la somnolence basé sur l'apprentissage profond (Deep Learning)

Présenté par :

Yahia BENTALEB
Thabet Amine SAAD SAOUD

Soutenu publiquement le : 15/06/2026

Devant le jury composé de :

Président : Dr. Benabid SONIA
Examinateur :Dr. Hafida CHELLAKH
Examinateur :Dr. Oussama SENOUCI

Encadrant : Dr. Sofiane MAZA

2025/2026

Dédicaces

À nos chères mères

À nos chers pères

Qui n'ont jamais cessé, de formuler des prières à notre égard, de nous soutenir et de nous épauler pour que nouspuissions atteindre nos objectifs.

À nos frères et soeurs

Qui nous ont aidés et supportés dans les moments difficiles.

À nos amis

Pour leur soutien moral, leurs précieux conseils et leur aide tout au long de notre parcours d'étude.

Yahia BENTALEB ,
Thabet Amine SAAD SAOUD

Remerciements

En premier lieu nous remercions dieux le tout puissant miséricordieux qui nous a donné la force et la patience d'accomplir ce modeste travail.

À nos chers parents, notre fierté, et bien sûr la source de notre succès car ils se sont sacrifiés pour nous offrir une ambiance de travail.

À notre superviseur Mr. Sofiane MAZA pour sa disponibilité, ses connaissances et son soutien.

Nous remercions tous nos professeurs qui nous ont enseignés durant nos études dans la faculté des Mathématiques et d'Informatique.

Enfin, nous tenons à remercier tous ceux qui, de près ou de loin, ont contribué à l'aboutissement de ce travail.

Yahia BENTALEB ,
Thabet Amine SAAD SAOUD

Résumé

Dans le cadre de notre projet, nous avons développé un système intelligent visant à détecter la somnolence du conducteur en temps réel, afin de réduire les accidents de la route causés par la perte de concentration la conduite. Ce système repose sur des techniques d'intelligence artificielle et de traitement d'images, permettant d'analyser les caractéristiques faciales du conducteur, notamment le mouvement des yeux, à l'aide d'une caméra. Le système a été conçu en utilisant le langage Python, en s'appuyant sur plusieurs bibliothèques telles que OpenCV et TensorFlow, ainsi que sur des outils et technologies modernes de développement. Il déclenche des alertes instantanées (sonores) lorsque les yeux du conducteur restent fermés pendant une certaine durée, contribuant ainsi à rétablir son attention et à prévenir les risques potentiels. L'objectif principal de ce projet est de proposer une solution pratique et efficace pouvant être intégrée dans les véhicules modernes ou utilisée comme application autonome. Cette solution vise à améliorer la sécurité routière et à réduire le nombre d'accidents, notamment lors des longs trajets et de la conduite nocturne. Nous estimons que ce système représente une avancée importante dans le domaine des systèmes de conduite intelligente et du renforcement de la sécurité sur les routes.

Abstract

As part of our project, we developed an intelligent system aimed at detecting driver drowsiness in real time, in order to reduce road accidents caused by loss of concentration while driving. This system relies on artificial intelligence and image processing techniques to analyze the driver's facial features, particularly eye movements, using a camera. The system was designed using the Python programming language, relying on several libraries such as OpenCV and TensorFlow, along with modern development tools and technologies. It triggers instant (audio) alerts when the driver's eyes remain closed for a certain period, helping to restore attention and prevent potential risks. The main objective of this project is to provide a practical and effective solution that can be integrated into modern vehicles or used as a standalone application. This solution aims to improve road safety and reduce the number of accidents, especially during long trips and nighttime driving. We believe that this system represents an important step forward in the development of intelligent driving systems and the enhancement of road safety.

الملخص

في إطار مشروعنا، قمنا بتطوير نظام ذكي يهدف إلى الكشف عن نعاس السائق في الوقت الحقيقي، وذلك للحد من حوادث المرور الناتجة عن فقدان التركيز أثناء القيادة. يعتمد هذا النظام على تقنيات الذكاء الاصطناعي ومعالجة الصور، حيث يقوم بتحليل ملامح وجه السائق، مثل حركة العينين، باستخدام الكاميرا. تم تصميم هذا النظام باستخدام لغة بايثون، بالاعتماد على مجموعة من المكتبات، من بينها OpenCV وTensorFlow، بالإضافة إلى استخدام أدوات وتقنيات حديثة في مجال التطوير. يقوم النظام بإرسال تنبيهات فورية (صوتية) للسائق عند إغلاق عينيه لمدة زمنية معينة، مما يساعده على استعادة انتباهه وتجنب المخاطر المحتملة.

يهدف هذا المشروع إلى تقديم حل عملي وفعال يمكن دمجه في المركبات الحديثة أو استخدامه كتطبيق مستقل، مما يساهم في تحسين السلامة المرورية وتقليل نسبة الحوادث، خاصة في الطرق الطويلة وأثناء القيادة الليلية. ونعتقد أن هذا النظام يمثل خطوة مهمة نحو تطوير أنظمة القيادة الذكية وتعزيز الأمان على الطرقات.

Table des matières

Résumé III

Abstract IV

الملخص V

Liste des figures VIII

Liste des tableaux IX

Liste des abréviations X

1. *Introduction Générale* 1

1.1. *Contexte* 1

1.2. *Problématique* 2

1.3. *Objectifs* 2

1.4. *Organisation du mémoire* 3

2. *Apprentissage profond* 4

2.1. *Introduction* 4

2.2. *Apprentissage profond* 5

2.2.1. *Qu'est-ce que l'apprentissage profond (DL) ?* 5

2.2.2. *Évolution de l'apprentissage automatique vers l'apprentissage profond (ML TO DL)* 6

2.2.3. *Apprentissage automatique vs apprentissage profond* 7

2.3. *Comment fonctionne l'apprentissage profond ?* 8

2.3.1. *Bases des réseaux de neurones* 8

2.4. *Réseaux de neurones convolutifs (CNNs)* 11

2.4.1. *Architecture des réseaux de neurones convolutifs (Architecture of CNNs)* 11

2.4.2. *Fonctions d'activation dans les CNN* 16

2.5. *Apprentissage par transfert* 19

2.5.1. *Concept de l'apprentissage par transfert (TL)* 19

2.5.2. *Avantages de l'apprentissage par transfert* 21

2.6. *Architectures CNN populaires dans la détection de la somnolence* 22

2.6.1. *MobileNet* 22

2.6.2. *DenseNet* 23

2.6.3. *ResNet* 23

2.7. *Description du jeu de données (Dataset Description MRL)* 23

2.8. *Conclusion* 25

3. *Conception et Réalisation* 26

3.1. *Introduction* 26

3.2. *Architecture Générale du Système* 26

3.3. *Réalisation* 31

3.3.1 *Langages utilisés* 31

3.3.1.1. *Python* 31

3.3.1.1.1 *Bibliothèques utilisées* 31

3.3.1.1.1.1 NumPy 31

3.3.1.1.1.2 Matplotlib 31

3.3.1.1.1.3 TensorFlow 31

3.3.1.1.1.4 OpenCV 32

3.3.2 *Outils utilisés* 32

3.3.2.1 Anaconda 32

3.3.2.1.1 *Jupyter Notebook* 32

3.4. *Résultats* 33

3.4.1. *Introduction* 33

3.4.2. *Cas 1 : Conducteur en état normal* 33

3.4.3. *Cas 2 : Fermeture des yeux* 34

3.4.4. *Cas 3 : Fermeture prolongée des yeux* 35

3.4.5. *Cas 4 : Réouverture des yeux* 36

3.5. *Conclusion* 37

Conclusion Générale 38

LISTE DES FIGURES

2.1 la relation entre IA et ML et DL *5*

2.2 Illustration of deep learning *5*

2.3 Evolution de ML to DL *6*

2.4 Comparaison entre le Machine Learning et le Deep Learning *7*

2.5 Structure du réseau neuronal *10*

2.6 L'architecture d'un réseau de neurones convolutionnel (CNN) *11*

2.7 Une représentation de base du processus de convolution bidimensionnelle. *12*

2.8 Mécanismes de padding et de stride dans l'extraction des caractéristiques *13*

2.9 Exemple de max pooling et d'average pooling *14*

2.10 Couches entièrement connectées *15*

2.11 Représentation graphique des fonctions d'activation Sigmoid et Tanh *17*

2.12 Exemple et courbe de la fonction ReLU *18*

2.13 fonction Softmax *18*

2.14 Apprentissage par transfert pour la détection de la somnolence *22*

2.15 Exemple de la base de données MRL Eye Dataset *25*

3.1 la méthodologie de notre projet *27*

4.1 Cas 1 : Conducteur en état normal *33*

4.2 Cas 2 : Fermeture des yeux *34*

4.3 Cas 3 : Fermeture prolongée des yeux *35*

4.4 Cas 3 : Fermeture prolongée des yeux *36*

LISTE DES TABLEAUX

2.1 Comparaison entre le Machine Learning et le Deep Learning 7

LISTE DES ABRÉVIATIONS

AI : Intelligence artificielle (Artificial Intelligence)

ML : Apprentissage automatique (Machine Learning)

DL : Apprentissage profond (Deep Learning)

SVM : Machines à vecteurs de support (Support Vector Machines)

ANN : Réseau de neurones artificiels (Artificial Neural Network)

DNN : Réseaux de neurones profonds (Deep Neural Networks)

CPU : Unité centrale de traitement (Central Processing Unit)

GPU : Unité de traitement graphique (Graphics Processing Unit)

TPU : Unité de traitement tensoriel (Tensor Processing Unit)

TL : Transfer Learning

ReLU : Rectified Linear Unit

Chapitre 1

Introduction Générale

1.1. Contexte

Avec l'augmentation continue du nombre de véhicules et la hausse des taux d'accidents de la route à travers le monde, la sécurité routière est devenue l'un des défis majeurs auxquels sont confrontées les sociétés modernes. L'état de somnolence du conducteur et la perte de concentration la conduite figurent parmi les principales causes de ces accidents, en particulier lors des longs trajets ou de la conduite nocturne.

Dans ce contexte, les avancées technologiques ont conanu un développement considérable, notamment dans les domaines de l'intelligence artificielle, du traitement d'images et de la vision par ordinateur. Ces progrès ont permis la mise en place de systèmes intelligents capables de surveiller le comportement du conducteur et de prédire son état en temps réel. Ces technologies ont ainsi contribué à proposer des solutions innovantes visant à améliorer la sécurité routière et à réduire les risques liés à la conduite.

Dans cette perspective, notre projet s'inscrit dans le développement d'un système intelligent de détection de la somnolence du conducteur, basé sur l'analyse des caractéristiques faciales à l'aide d'une caméra, dans le but de réduire les accidents et de renforcer la sécurité sur les routes.

1.2. **Problématique**

Bien que certains systèmes traditionnels d'alerte du conducteur soient disponibles, leur efficacité demeure limitée. En effet, ces systèmes ne reposent pas sur une analyse directe de l'état réel du conducteur, mais s'appuient plutôt sur des indicateurs indirects et souvent imprécis, tels que la durée de conduite ou les mouvements du véhicule.
Par ailleurs, de nombreuses solutions existantes ne garantissent ni une précision suffisante ni un fonctionnement en temps réel, ce qui réduit considérablement leur capacité à prévenir les accidents. De plus, certains de ces systèmes sont coûteux ou difficiles à intégrer dans l'ensemble des types de véhicules.
Dans ce contexte, la problématique principale de ce projet peut être formulée comme suit :
Comment concevoir un système intelligent, fiable et performant, capable de détecter la somnolence du conducteur en temps réel avec une grande précision, en se basant sur l'analyse des caractéristiques faciales, notamment le mouvement des yeux ?

1.3. **Objectifs**

L'objectif principal de ce projet est de développer un système intelligent capable de détecter la somnolence du conducteur en temps réel, en s'appuyant sur des techniques d'intelligence artificielle et de traitement d'images.
Afin d'atteindre cet objectif, plusieurs objectifs spécifiques ont été définis, à savoir :

- Analyser les caractéristiques faciales du conducteur à l'aide d'une caméra .
- Suivre le mouvement des yeux et détecter les de fermeture .
- Déterminer la durée de fermeture des yeux afin d'estimer le niveau de somnolence .
- Déclencher des alertes sonores instantanées lors de la détection de la somnolence .
- Développer un système pratique pouvant être utilisé comme application autonome ou intégré dans les véhicules .

1.4. *Organisation du mémoire*

Le présent mémoire est organisé en trois chapitres :

Le premier chapitre présente une introduction générale au projet. Il expose le contexte de l'étude, la problématique abordée, les objectifs visés ainsi que l'organisation générale du mémoire.

Le deuxième chapitre est consacré à l'étude théorique des concepts utilisés dans ce travail. Il présente les notions de l'apprentissage profond, les réseaux de neurones convolutifs (CNN), l'apprentissage par transfert ainsi que les architectures employées dans la détection de la somnolence du conducteur.

Le troisième chapitre décrit la conception et la réalisation du système proposé. Il présente l'architecture générale de la solution, les outils et technologies utilisés, les différentes étapes de développement ainsi que les résultats obtenus et leur interprétation.

Chapitre 2

Apprentissage profond

2.1. Introduction

L'intelligence artificielle (IA) vise à reproduire les capacités humaines de raisonnement et de prise de décision. Elle permet aux machines d'apprendre à partir des données et de s'adapter à leur environnement. Une branche essentielle de l'IA est l'apprentissage automatique (Machine Learning, ML), qui permet aux systèmes de s'améliorer à partir de l'expérience sans être explicitement programmés. [1]

Avec l'augmentation de la complexité des problèmes, l'apprentissage profond (Deep Learning, DL) s'est imposé comme une extension puissante du Machine Learning. Il permet aux modèles d'extraire automatiquement des caractéristiques pertinentes à partir de grandes quantités de données et d'atteindre des performances élevées dans de nombreux domaines scientifiques et technologiques. [2]

Ce chapitre se propose d'étudier l'apprentissage profond, de mettre en évidence ses différences par rapport à l'apprentissage automatique traditionnel, ainsi que son aptitude à traiter de grands volumes de données afin de détecter des motifs complexes avec une grande précision.

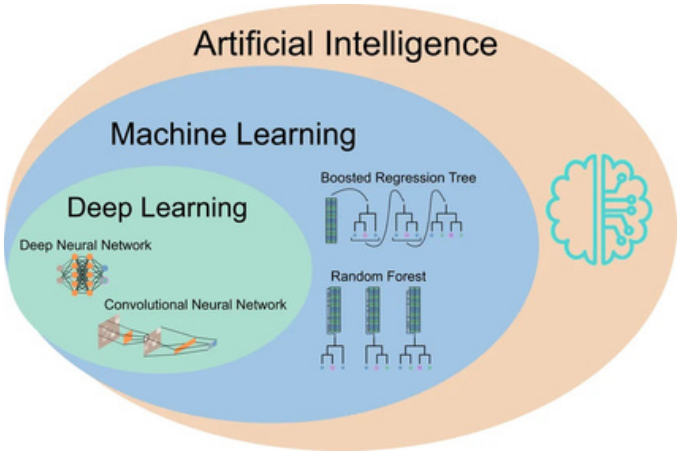


Figure 2.1 la relation entre IA et ML et DL [3]

2.2. **Apprentissage profond**

2.2.1. **Qu'est-ce que l'apprentissage profond (DL) ?**

L'apprentissage profond (Deep Learning) est un type d'apprentissage automatique qui utilise des réseaux de neurones artificiels afin d'apprendre à partir des données, de manière similaire au processus d'apprentissage humain. Les réseaux de neurones artificiels sont inspirés du fonctionnement du cerveau humain et peuvent être utilisés pour résoudre une grande variété de problèmes, tels que la reconnaissance d'images, le traitement du langage naturel et la reconnaissance vocale . [4]

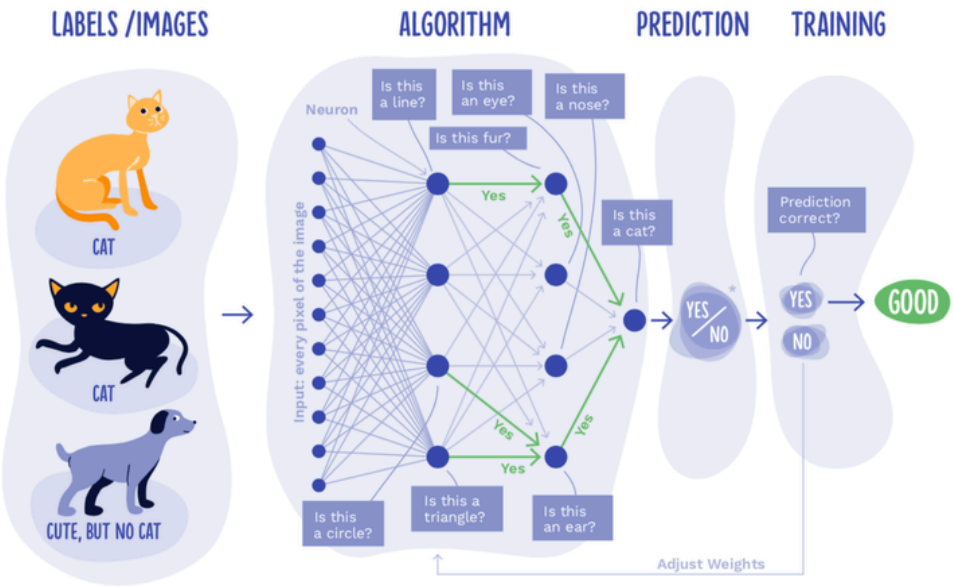


Figure 2.2 Illustration of deep learning [5]

2.2.2. Évolution de l'apprentissage automatique vers l'apprentissage profond (ML TO DL)

Avec l'évolution de l'intelligence artificielle, les méthodes traditionnelles d'apprentissage automatique basées sur des modèles statistiques, tels que les arbres de décision et la régression logistique, ont montré certaines limites. En effet, elles nécessitent souvent une extraction manuelle des caractéristiques (features), ce qui devient particulièrement difficile dans le cas de données complexes comme les images et les textes.

Les réseaux de neurones artificiels (Artificial Neural Networks, ANNs) ont suscité un intérêt croissant, mais leur utilisation était initialement limitée par le manque de données disponibles et la faible puissance de calcul.

Au début du XXIe siècle, des avancées majeures ont été réalisées grâce à l'utilisation des processeurs graphiques (GPU) et à la disponibilité de grandes quantités de données (Big Data), permettant ainsi l'entraînement de modèles plus profonds et plus performants. Par ailleurs, des approches combinant des méthodes comme les machines à vecteurs de support (Support Vector Machines, SVM) avec la vision par ordinateur ont contribué à améliorer significativement la reconnaissance automatique de motifs.

Ces progrès ont conduit à l'émergence des réseaux de neurones profonds (Deep Neural Networks, DNNs), capables d'apprendre directement à partir de données brutes avec une intervention humaine minimale. Aujourd'hui, ces modèles surpassent largement les méthodes traditionnelles dans de nombreux domaines. [6]

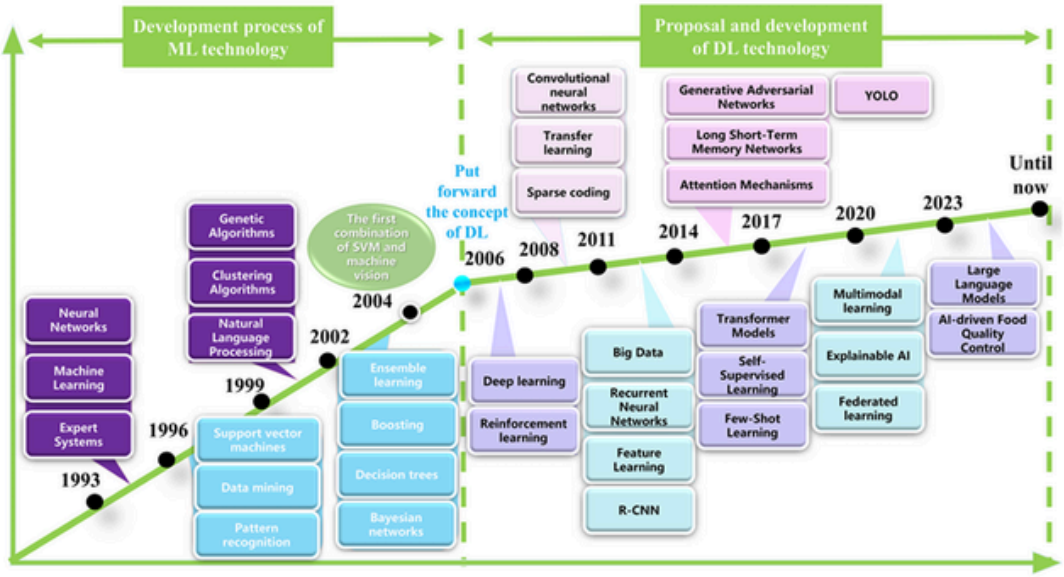


Figure 2.3 Evolution de ML to DL [6_1]

2.2.3. Apprentissage automatique vs apprentissage profond

Le Machine Learning (ML) et le Deep Learning (DL) sont deux branches fondamentales de l'intelligence artificielle (IA), visant à permettre aux ordinateurs d'apprendre à partir des données. Bien que ces deux approches soient utilisées pour effectuer des prédictions et automatiser la prise de décision, elles se distinguent par la manière dont elles traitent les données ainsi que par la complexité des modèles qu'elles utilisent.

Comprendre les différences entre ces deux approches permet de choisir la méthode la plus adaptée à un problème donné, d'optimiser l'utilisation des ressources et d'obtenir de meilleurs résultats dans des applications réelles. [7]

Critère	Machine Learning (ML)	Deep Learning (DL)
Type de données	Fonctionne principalement avec des données structurées (tableaux, CSV)	Adapté aux données non structurées (images, audio, texte)
Extraction des caractéristiques	Nécessite une extraction manuelle des caractéristiques (features)	Apprend automatiquement les caractéristiques à partir des données
Taille des données	Efficace avec de petites ou moyennes quantités de données	Nécessite de grandes quantités de données pour de bonnes performances
Complexité du modèle	Modèles plus simples et faciles à interpréter	Modèles complexes basés sur des réseaux de neurones profonds
Ressources de calcul	Faibles besoins en calcul, fonctionne sur CPU	Nécessite des ressources élevées (GPU/TPU) pour l'entraînement

Table 2.1 Comparaison entre le Machine Learning et le Deep Learning

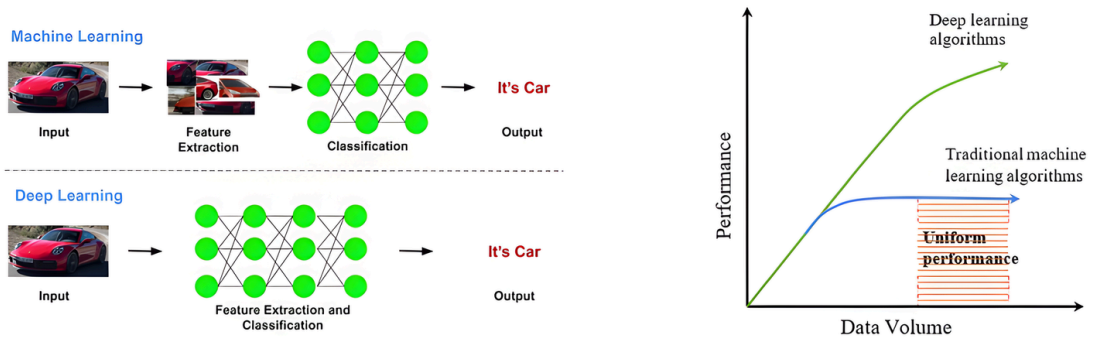


Figure 2.4 Comparaison entre le Machine Learning et le Deep Learning [6.2] [6.3]

2.3. Comment fonctionne l'apprentissage profond ?

L'apprentissage profond fonctionne en utilisant des réseaux de neurones artificiels pour apprendre à partir des données. Ces réseaux sont constitués de plusieurs couches de nœuds interconnectés, où chaque nœud est chargé d'apprendre une caractéristique spécifique des données.
En reprenant l'exemple de la reconnaissance d'images, la première couche peut apprendre à détecter des éléments simples tels que les contours (bords), la deuxième couche peut identifier des formes, tandis que la troisième couche peut reconnaître des objets plus complexes.
Au fur et à mesure de l'apprentissage, les poids des connexions entre les neurones sont ajustés afin d'améliorer la capacité du réseau à classer correctement les données. Ces ajustements sont effectués en fonction des performances du modèle sur les données d'entraînement, généralement à l'aide de techniques telles que la rétropropagation (backpropagation).
Ce processus est appelé entraînement (training), et il peut être réalisé selon différentes approches, notamment l'apprentissage supervisé, l'apprentissage non supervisé et l'apprentissage par renforcement. [8]

2.3. Bases des réseaux de neurones

À un niveau général, l'inspiration des réseaux de neurones provient des neurones biologiques du cerveau humain, qui communiquent à travers des signaux électriques. En 1943, Warren McCulloch et Walter Pitts ont proposé le premier modèle mathématique d'un neurone, montrant que des unités simples pouvaient effectuer des calculs de fonctions. Plus tard, en 1958, Frank Rosenblatt a introduit le perceptron, un algorithme conçu pour la reconnaissance de formes. Le perceptron est l'ancêtre historique des réseaux neuronaux actuels : il s'agit essentiellement d'un modèle linéaire avec une sortie contrainte. Dans la section suivante, nous allons voir plus en détail comment les réseaux de neurones s'inspirent du cerveau humain pour prendre des décisions et reconnaître des motifs. [9]

Un réseau de neurones peut être compris à travers un exemple simple : la détection de spam. Un e-mail est introduit dans le réseau, et des caractéristiques telles que des mots ou expressions comme « prix », « argent », « cher ami » ou « gagner » sont utilisées comme entrées. Les premiers neurones du réseau traitent l'importance de chaque signal, tandis que les couches plus profondes combinent ces informations en représentations de plus haut niveau qui capturent le contexte et le ton. La couche finale calcule ensuite une probabilité indiquant si l'e-mail est un spam, et si cette probabilité est suffisamment élevée, le message est classé comme spam. En résumé, le réseau apprend à transformer des caractéristiques brutes en motifs significatifs afin de faire des prédictions.

Ce processus repose sur deux concepts fondamentaux : les poids et les biais. Les poids agissent comme des réglages qui contrôlent l'influence de chaque caractéristique d'entrée sur la décision — un mot comme « prix » peut avoir un poids plus élevé qu'un mot courant comme « bonjour ». Les biais sont des valeurs ajoutées qui décalent le seuil de décision, permettant à un neurone de s'activer même si les entrées sont faibles. Ensemble, ces paramètres déterminent la contribution de chaque neurone au calcul global. En ajustant ces valeurs pendant l'entraînement, le réseau apprend progressivement à effectuer des prédictions précises, par exemple déterminer si un e-mail est du spam ou non.

Mathématiquement, un réseau de neurones apprend une fonction

$f(X)$ en associant un vecteur d'entrée

$X = (x_1, x_2, x_3, \ldots)$

à une sortie prédite Y. Ce qui distingue les réseaux de neurones des autres algorithmes d'apprentissage automatique traditionnels est leur structure en couches et leur capacité à effectuer des transformations non linéaires.

Un réseau de neurones est composé de : [9]

Couche d'entrée (input layer) : elle contient les caractéristiques brutes (X_1, X_2, X_3, …). [9]

Couches cachées (hidden layers) : elles sont constituées de neurones artificiels (ou nœuds) qui transforment les entrées en nouvelles représentations. Mathématiquement, les couches cachées appliquent une transformation des données d'entrée en multipliant les caractéristiques par leurs poids associés, puis en ajoutant un biais, avant de passer à la couche suivante jusqu'à atteindre la sortie finale. C'est ici que se produit la transformation linéaire entre entrée et sortie. [9]

Couche de sortie (output layer) : après la transformation linéaire effectuée dans la couche cachée, une fonction d'activation non linéaire (tanh, sigmoid, ReLU) est appliquée afin de produire la prédiction finale (par exemple une valeur pour la régression ou une probabilité pour la classification). [9]

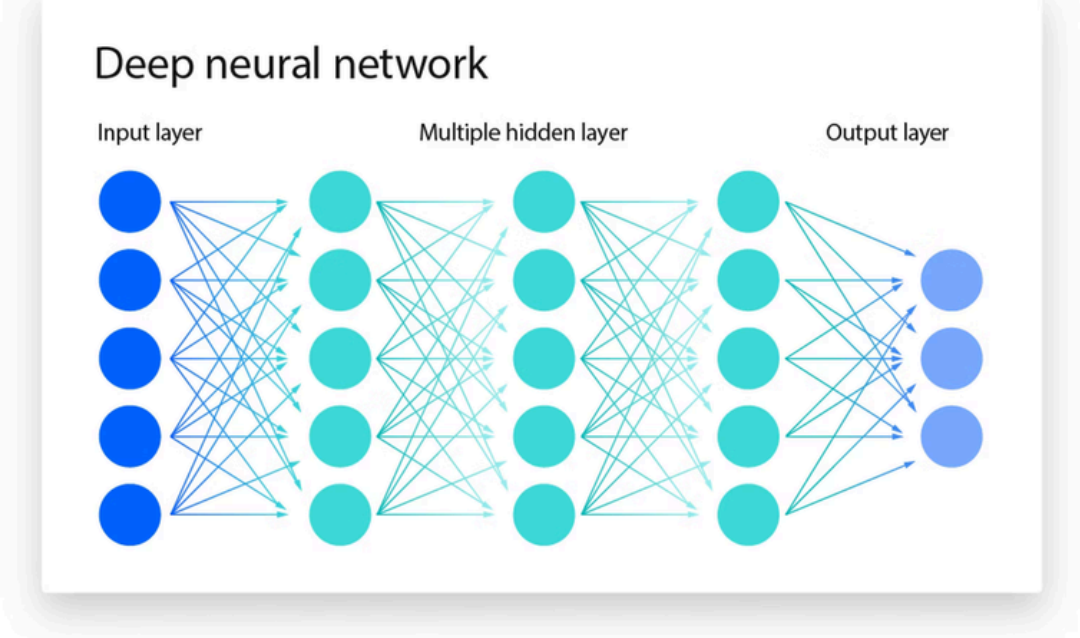


Figure 2.5 Structure du réseau neuronal [9]

2.4. Réseaux de neurones convolutifs (CNNs)

Les réseaux de neurones convolutionnels (CNN) sont très similaires aux réseaux de neurones classiques , ils sont constitués de neurones possédant des poids et des biais apprenables. Chaque neurone reçoit des entrées, effectue un produit scalaire (dot product) et peut ensuite appliquer une fonction de non-linéarité. L'ensemble du réseau représente toujours une fonction de score différentiable unique : des pixels bruts de l'image en entrée jusqu'aux scores de classification en sortie.
De plus, ils utilisent également une fonction de perte (comme SVM ou Softmax) dans la dernière couche entièrement connectée (fully-connected layer), et toutes les techniques utilisées pour entraîner les réseaux de neurones classiques restent valables. [10]

2.4.1. Architecture des réseaux de neurones convolutifs

Les réseaux de neurones convolutionnels (CNN) sont principalement composés de deux blocs essentiels :

- Bloc d'extraction de caractéristiques (Feature Extraction Block) :
- Il est responsable de l'extraction des caractéristiques importantes à partir des données d'entrée, en utilisant des opérations telles que la convolution et le pooling.
- Bloc de classification (Classification Block) :
- Il est chargé de classer les caractéristiques extraites en différentes classes cibles, en utilisant des couches entièrement connectées (fully connected layers).[10]

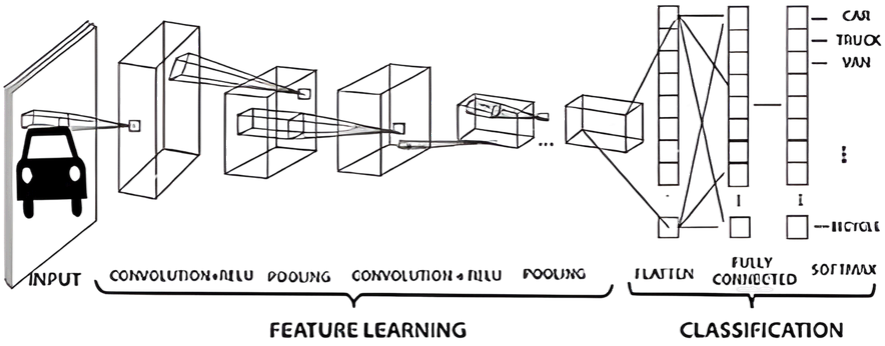


Figure 2.6 L'architecture d'un réseau de neurones convolutionnel (CNN) [10_1]

Le processus commence par l'extraction des caractéristiques importantes à partir des données en utilisant des couches spécialisées, notamment :

a - Couche de convolution :

La couche de convolution effectue une opération mathématique appelée convolution sur les données d'entrée. La convolution est utilisée pour déterminer le degré de chevauchement entre les données d'entrée et un filtre (ou noyau, kernel).

Dans les CNN, le noyau se déplace (glisse) sur la matrice d'entrée, en effectuant une multiplication élément par élément, suivie d'une somme, afin de produire des cartes de caractéristiques (feature maps).

Par exemple, la convolution d'une région 3×3 située dans le coin supérieur droit d'une image avec un noyau spécifique peut produire une valeur comme 77 (voir Figure 2.7).

Cette opération permet de réaliser différentes fonctions de traitement d'image, telles que :

- la détection des contours (edge detection)
- l'accentuation (sharpening)
- le flou (blurring) [10]

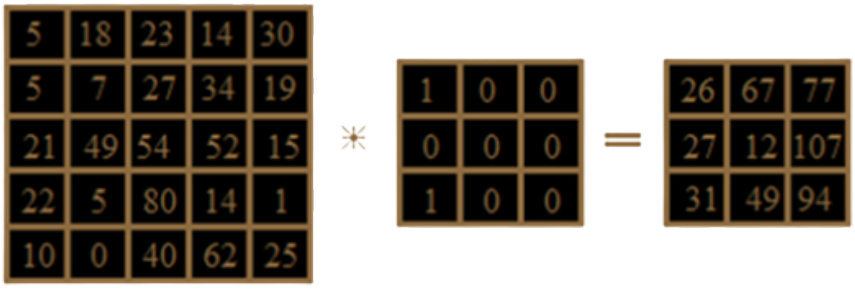


Figure 2.7 Une représentation de base du processus de convolution bidimensionnelle [10 2]

En empilant plusieurs couches de convolution, les CNN peuvent apprendre des représentations hiérarchiques, en partant des pixels bruts, puis en progressant vers les contours (edges), les formes primitives, les parties d'objets, et finalement les objets complets.

Après chaque convolution, une fonction d'activation non linéaire, généralement la Rectified Linear Unit (ReLU), est appliquée afin d'améliorer l'apprentissage en mettant à zéro les valeurs négatives dans les cartes de caractéristiques (feature maps).

Plusieurs hyperparamètres clés influencent fortement les performances des couches de convolution, notamment :

1- Depth (Profondeur)

Fait référence au nombre de filtres utilisés.

Elle détermine combien de caractéristiques différentes le réseau peut apprendre.

2- Stride (Pas)

Indique le nombre de pixels dont le filtre se déplace à chaque étape sur l'entrée.

Il influence directement les dimensions spatiales de la sortie.

3- Padding (Remplissage)

Le zero-padding consiste à ajouter des zéros autour de l'entrée afin de :

- préserver les dimensions spatiales
- conserver les informations sur les bords [10]

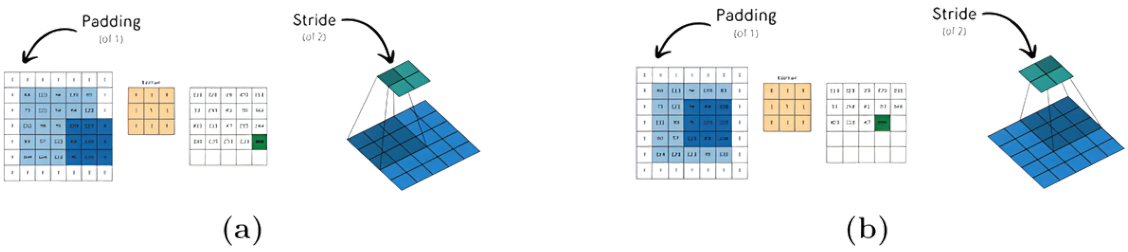


Figure 2.8 Mécanismes de padding et de stride dans l'extraction des caractéristiques[10_3]

b- Couche de pooling

La couche de pooling réduit les dimensions spatiales des cartes d'activation (activation maps) tout en conservant les informations essentielles.

Les stratégies de pooling les plus courantes incluent :

Max pooling

Le max pooling conserve la valeur maximale à l'intérieur de chaque fenêtre de pooling.

Cela permet d'extraire les caractéristiques les plus importantes et les plus dominantes.

Average pooling

Le average pooling calcule la moyenne des valeurs à l'intérieur de chaque fenêtre de pooling.

Il produit ainsi une représentation plus lissée et homogène des données. [10]

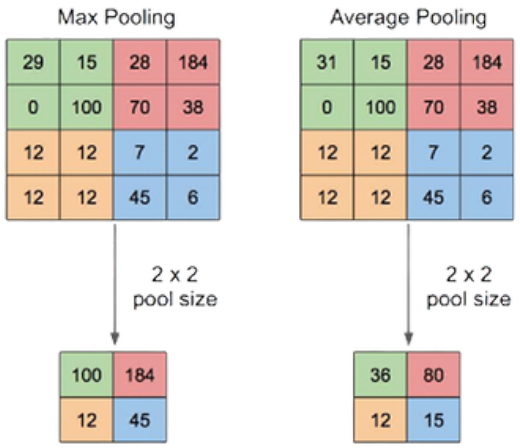


Figure 2.9 Exemple de max pooling et d'average pooling [27]

Le pooling améliore l'invariance du modèle aux transformations de translation et de déformation, tout en réduisant le nombre de paramètres et les coûts de calcul, ce qui augmente l'efficacité globale du réseau.

Après l'extraction des caractéristiques, les cartes de caractéristiques traitées sont envoyées vers la partie de classification, qui comprend :

c- Couche entièrement connectée (Fully Connected Layer)

Les cartes de caractéristiques sont d'abord transformées en un vecteur unidimensionnel grâce à l'opération de flattening.

Ce vecteur est ensuite transmis aux couches entièrement connectées, où chaque neurone est connecté à tous les neurones de la couche suivante.
Ces couches analysent et combinent les caractéristiques apprises afin de produire les prédictions finales.
La sortie de la dernière couche entièrement connectée passe généralement par une fonction d'activation Softmax, qui transforme les sorties brutes en une distribution de probabilités sur les classes cibles.

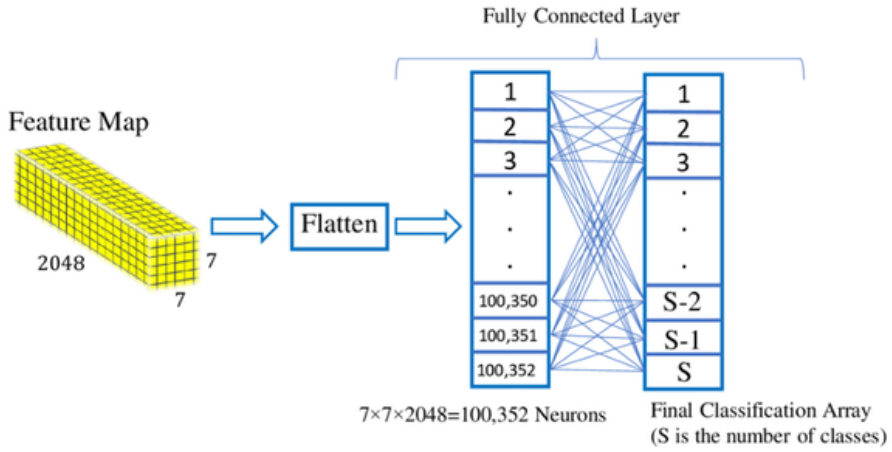


Figure 2.10 Couches entièrement connectées [24]

Cette structure hiérarchique, caractérisée par la succession de couches de convolution, de pooling et de couches entièrement connectées, permet aux CNN d'extraire et d'intégrer progressivement les caractéristiques visuelles. Grâce à cette conception organisée et multi-couches, les CNN atteignent une grande précision et une grande efficacité dans un large éventail d'applications visuelles intelligentes, notamment la reconnaissance faciale et la conduite autonome.

2.4.2. Fonctions d'activation dans les CNN

Les fonctions d'activation sont des composants essentiels des réseaux de neurones convolutifs (CNNs – Convolutional Neural Networks), car elles introduisent la non-linéarité dans le modèle. Cette non-linéarité permet au réseau d'apprendre et d'approximer des motifs et des relations complexes dans les données, ce qui ne peut pas être réalisé uniquement à l'aide de transformations linéaires.
Ci-dessous sont présentées les fonctions d'activation les plus couramment utilisées en deep learning, chacune ayant des caractéristiques et des applications spécifiques :

2.4.2.1 Fonction sigmoid

La fonction sigmoid , également appelée fonction logistique, transforme toute valeur réelle en une valeur comprise entre 0 et 1. Elle est souvent utilisée pour convertir les sorties d'un modèle en probabilités .
Elle est définie mathématiquement comme suit : [11]

$$\sigma(x) = 1/(1+e-x)$$

Sa courbe ressemble à la lettre S, ce qui lui permet de représenter de manière fluide et continue des sorties probabilistes.

2.4.2.2 Tanh (Tangente hyperbolique)

La fonction tangente hyperbolique, ou tanh, est similaire à la fonction sigmoïde, mais elle produit des valeurs comprises entre −1 et 1 .
Elle est définie mathématiquement comme suit : [11]

$$\tanh(x) = 2\sigma(2x)-1$$

Cette fonction a également une forme en S, mais elle est centrée autour de zéro, ce qui lui permet de produire des sorties avec une moyenne proche de zéro. Cela conduit souvent à une meilleure convergence pendant l'entraînement par rapport à la fonction sigmoïde.

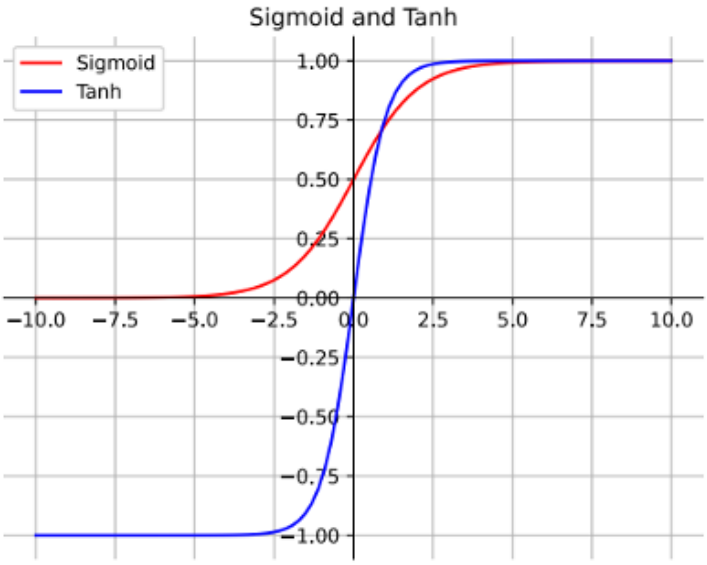


Figure 2.11 Représentation graphique des fonctions d'activation Sigmoid et Tanh [11]

2.4.2.3 ReLU (Rectified Linear Unit)

La fonction ReLU (Rectified Linear Unit) est devenue la fonction d'activation la plus utilisée dans les modèles modernes de deep learning. Elle introduit la non-linéarité en transformant toutes les valeurs d'entrée négatives en zéro, tandis que les valeurs positives sont conservées sans modification [11].

Elle est définie mathématiquement comme suit :

f(x)=max(0,x)f(x) = \max(0, x)f(x)=max(0,x)

La figure ci-dessous illustre le comportement de la fonction ReLU.

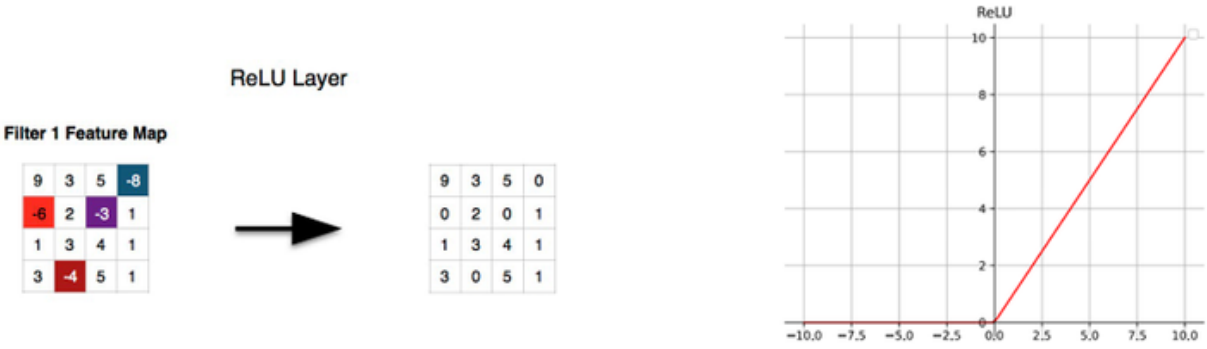


Figure 2.12 Exemple et courbe de la fonction ReLU [11]

2.4.2.4 Fonction Softmax

La fonction softmax est principalement utilisée dans la couche de sortie des réseaux de neurones pour les problèmes de classification multi-classes. Elle transforme les scores bruts (logits) en une distribution de probabilités sur les différentes classes, où la somme de toutes les probabilités est égale à 1.

La fonction softmax met en évidence les valeurs les plus élevées du vecteur d'entrée tout en réduisant l'influence des valeurs plus faibles, ce qui facilite l'identification de la classe la plus probable [11].

Les diagrammes ci-dessous illustrent comment la fonction softmax transforme un ensemble de scores bruts en une distribution de probabilités normalisée.

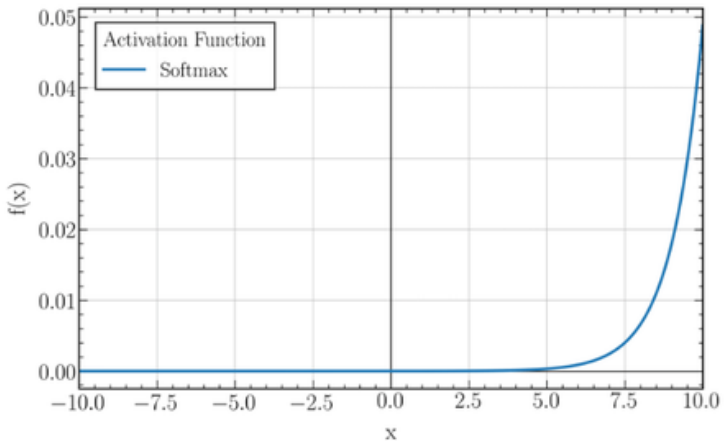


Figure 2.13 fonction Softmax [25]

2.5. Apprentissage par transfert

2.5.1. Concept de l'apprentissage par transfert (TL)

En pratique, très peu de personnes entraînent un réseau convolutionnel complet à partir de zéro (avec une initialisation aléatoire), car il est relativement rare de disposer d'un jeu de données de taille suffisante. Il est courant de pré-entraîner un ConvNet sur un très grand ensemble de données (par exemple ImageNet, qui contient 1,2 million d'images avec 1000 catégories), puis d'utiliser le ConvNet soit comme initialisation, soit comme extracteur de caractéristiques fixes pour la tâche d'intérêt. Les trois principaux scénarios de transfert d'apprentissage sont les suivants :

1-**ConvNet comme extracteur de caractéristiques fixes.** Prenez un ConvNet pré-entraîné sur ImageNet, supprimez la dernière couche entièrement connectée (les sorties de cette couche sont les 1000 scores de classe pour une tâche différente comme ImageNet), puis traitez le reste du ConvNet comme un extracteur de caractéristiques fixes pour le nouveau jeu de données. Dans un AlexNet, cela calculerait un vecteur 4096-D pour chaque image contenant les activations de la couche cachée juste avant le classificateur. Nous appelons ces rubriques des codes CNN. Il est important pour les performances que ces codes soient ReLUd (c'est-à-dire seuils à zéro) s'ils l'étaient également lors de l'entraînement du ConvNet sur ImageNet (comme c'est généralement le cas). Une fois que vous avez extrait les codes 4096-D pour toutes les images, entraînez un classificateur linéaire (par exemple Linear SVM ou Softmax) pour le nouvel ensemble de données.

2-**Ajustement fin du ConvNet.** La deuxième stratégie consiste non seulement à remplacer et réentraîner le classificateur au-dessus du ConvNet sur le nouvel ensemble de données, mais aussi à affiner les poids du réseau pré-entraîné en poursuivant la rétropropagation. Il est possible d'affiner toutes les couches du ConvNet, ou de garder certaines couches antérieures fixes [12]

(en raison de préoccupations de surappui) et d'affiner seulement une partie de niveau supérieur du réseau. Cela est motivé par l'observation que les fonctionnalités antérieures d'un ConvNet contiennent des caractéristiques plus génériques (par exemple des détecteurs de bords ou des détecteurs de blobs de couleur) qui devraient être utiles à de nombreuses tâches, mais les couches ultérieures du ConvNet deviennent progressivement plus spécifiques aux détails des classes contenues dans le jeu de données original. Dans le cas d'ImageNet par exemple, qui contient de nombreuses races de chiens, une part significative du pouvoir représentatif du ConvNet peut être consacrée à des caractéristiques spécifiques à la différenciation entre les races de chiens.

3-**Des modèles pré-entraînés.** Comme les ConvNet modernes nécessitent 2 à 3 semaines d'entraînement sur plusieurs GPU sur ImageNet, il est courant de voir des gens publier leurs derniers points de contrôle ConvNet pour le bénéfice de ceux qui peuvent utiliser les réseaux pour des réglages fins. Par exemple, la bibliothèque Caffe possède un Zoo modèle où les gens partagent leurs poids réseau.

Quand et comment peaufiner ? Comment décidez-vous quel type d'apprentissage par transfert effectuer sur un nouveau jeu de données ? Cela dépend de plusieurs facteurs, mais les deux plus importants sont la taille du nouveau jeu de données (petit ou grand), et sa similarité avec l'ensemble de données original (par exemple, similaire à ImageNet en termes de contenu des images et de classes, ou très différent, comme les images de microscope). En gardant à l'esprit que les fonctionnalités de ConvNet sont plus génériques dans les premières couches et plus spécifiques au jeu de données originales dans les couches suivantes, voici quelques règles générales pour naviguer dans les 4 scénarios majeurs :

- Le nouveau jeu de données est petit et similaire à l'original. Comme les données sont petites, il n'est pas judicieux d'ajuster le ConvNet en raison de préoccupations liées au surapprentissage. Puisque les données sont similaires à celles originales, [12]

nous attendons à ce que des fonctionnalités de plus haut niveau dans le ConvNet soient également pertinentes pour cet ensemble de données. Ainsi, la meilleure idée serait peut-être d'entraîner un classificateur linéaire sur les codes CNN.

- Le nouveau jeu de données est volumineux et similaire à celui d'origine. Comme nous disposons de plus de données, nous pouvons avoir plus confiance que nous ne sur-équiparons pas si nous essayions d'affiner l'ensemble du réseau.
- Le nouveau jeu de données est petit mais très différent de l'original de données. Comme les données sont petites, il est probablement préférable de n'entraîner qu'un classificateur linéaire. Puisque le jeu de données est très différent, il n'est peut-être pas préférable d'entraîner le classificateur depuis le sommet du réseau, qui contient davantage de caractéristiques spécifiques à chaque ensemble. Au lieu de cela, il pourrait être préférable d'entraîner le classificateur SVM à partir d'activations antérieures quelque part dans le réseau.
- Le nouveau jeu de données est vaste et très différent de l'ensemble de données original. Comme le jeu de données est très vaste, on peut s'attendre à pouvoir nous permettre d'entraîner un ConvNet à partir de zéro. Cependant, en pratique, il est très souvent bénéfique d'initialiser avec des poids provenant d'un modèle pré-entraîné. Dans ce cas, nous aurions suffisamment de données et de confiance pour affiner l'ensemble du réseau. [12]

2.5.2. Avantages de l'apprentissage par transfert

Les modèles neuronaux utilisés dans l'apprentissage par transfert reposent sur des ensembles de données variés, ce qui leur permet d'apprendre des représentations générales pouvant être adaptées à de nouvelles tâches. L'un des principaux avantages est la convergence plus rapide de l'entraînement, car les connaissances préalables fournissent une bonne initialisation, réduisant ainsi le temps et les coûts de calcul, en particulier lorsque les données annotées sont limitées [12] .

Ces modèles améliorent également les performances et la capacité de généralisation, atteignant souvent une meilleure précision par rapport aux modèles entraînés à partir de zéro. De plus, ils permettent un apprentissage efficace avec un nombre réduit d'exemples annotés, ce qui les rend particulièrement adaptés aux domaines où les ressources en données sont limitées .

Enfin, leur architecture réutilisable permet une adaptation facile à plusieurs tâches sans devoir tout reconstruire depuis le début, ce qui réduit les coûts de développement et accélère le déploiement. [12]

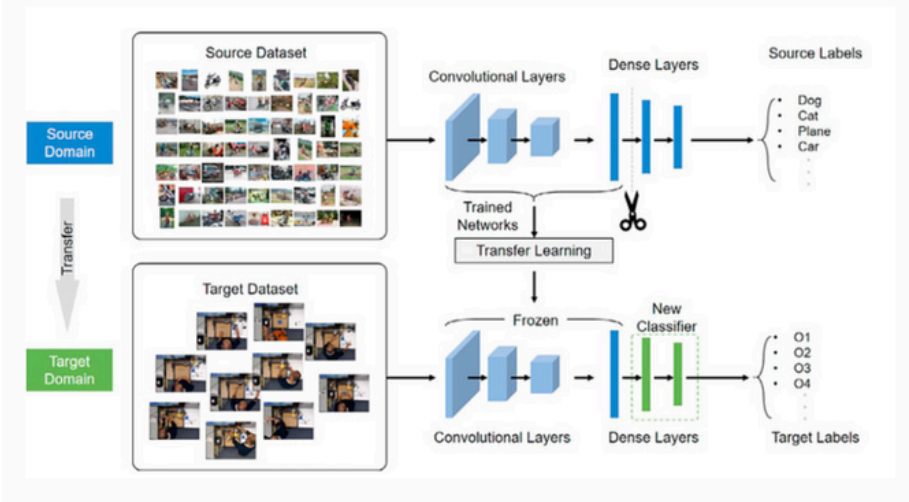


Figure 2.14 Apprentissage par transfert pour la détection de la somnolence [26]

2.6. Architectures CNN populaires dans la détection de la somnolence

Les réseaux de neurones convolutifs (CNN) comprennent un large éventail d'architectures, les plus courantes étant :

2.6.1. MobileNet

MobileNet est une architecture de réseau de neurones convolutif conçue pour les applications mobiles et embarquées. Elle repose sur l'utilisation de depthwise separable convolutions, ce qui permet de réduire considérablement le nombre de paramètres et le coût de calcul tout en maintenant de bonnes performances [13]

2.6.2. DenseNet

DenseNet est une architecture de CNN caractérisée par des connexions denses entre les couches. Chaque couche reçoit en entrée les sorties de toutes les couches précédentes, ce qui améliore la propagation de l'information et réduit le problème de disparition du gradient. [14]

2.6.3. ResNet

ResNet est une architecture de réseau de neurones profond qui introduit les connexions résiduelles (skip connections). Ces connexions permettent de faciliter l'entraînement de réseaux très profonds en évitant la dégradation des performances lorsque la profondeur augmente. [15]

2.7. Description du jeu de données (Dataset Description MRL)

Dans le cadre de ce projet de détection de la somnolence du conducteur, nous avons utilisé la base de données MRL Eye Dataset, une base de données à grande échelle dédiée à l'analyse des yeux humains.
Cette base contient 84 898 images d'yeux, collectées à partir de 37 sujets différents (33 hommes et 4 femmes), dans des conditions réelles de capture. Les images ont été acquises à l'aide de plusieurs capteurs et dans diverses conditions d'éclairage, ce qui rend cette base particulièrement adaptée aux applications de vision par ordinateur en environnement réel. [16]

Les images sont disponibles en différentes résolutions (faible et haute résolution) et incluent des images infrarouges. Cette diversité permet d'entraîner et d'évaluer des modèles robustes pour des tâches telles que la détection des yeux, l'estimation du regard et la détection du clignement.

Chaque image est accompagnée d'annotations détaillées, notamment :

- l'identifiant du sujet
- l'identifiant de l'image
- le genre (homme ou femme)
- la présence de lunettes
- l'état de l'œil (ouvert ou fermé)
- le niveau de réflexion lumineuse (aucune, faible, forte)
- les conditions d'éclairage (bonnes ou mauvaises)
- le type de capteur utilisé (RealSense, IDS, Aptina)

De plus, environ 15 000 images disposent d'annotations précises de la position de la pupille, ce qui permet des analyses avancées telles que la localisation de la pupille et l'estimation du regard.

Les images ont été extraites à l'aide d'un détecteur d'yeux basé sur la méthode Histogram of Oriented Gradients (HOG) combinée avec un classificateur SVM, garantissant ainsi une bonne qualité des régions d'intérêt.

Grâce à sa richesse et à la diversité de ses annotations, cette base de données constitue un outil idéal pour l'entraînement et l'évaluation de modèles de détection de la somnolence, notamment pour distinguer les états des yeux (ouverts/fermés), qui est un indicateur clé de la fatigue du conducteur. [16]

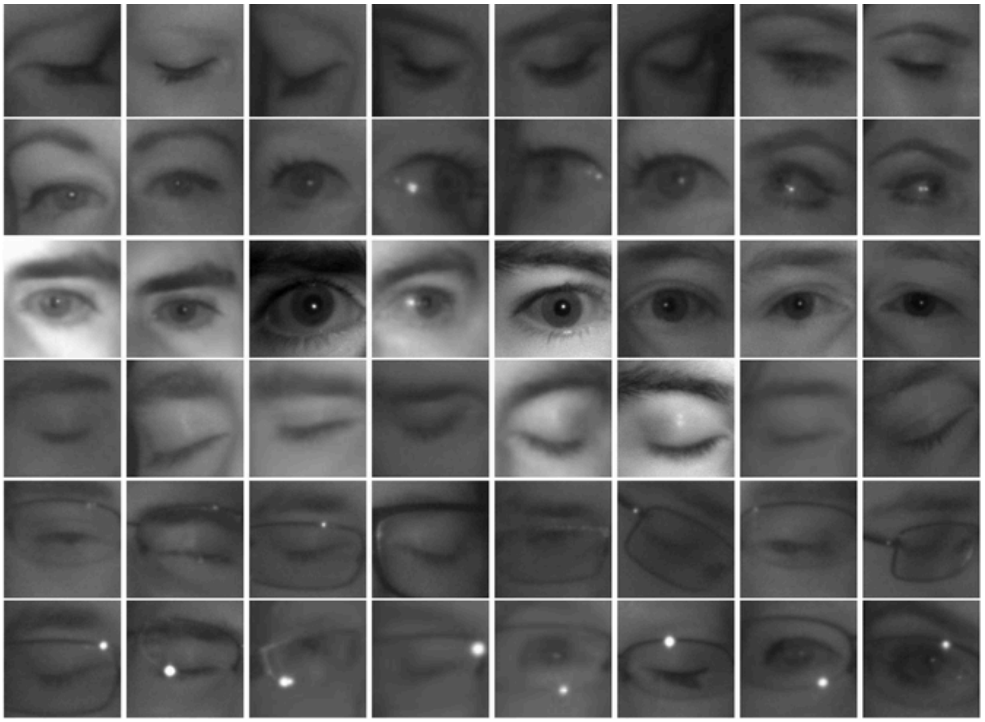


Figure 2.15 : Exemple de la base de données MRL Eye Dataset [16]

2.8. Conclution

Dans ce chapitre, nous avons examiné les fondements théoriques du deep learning, depuis les principes de base des réseaux de neurones jusqu'à leurs applications avancées. Nous avons accordé une attention particulière à la structure analytique des réseaux de neurones convolutifs (CNN), en analysant en détail leurs différentes couches ainsi que leurs mécanismes de fonctionnement.
Par ailleurs, nous avons étudié la méthodologie du transfer learning adoptée dans notre travail. Notre analyse a également porté sur les principales architectures utilisées dans ce domaine.

Chapitre 3

Conception et Réalisation

3.1. **Introduction**

Dans ce chapitre, nous présentons en détail les différentes étapes de réalisation de notre projet, en mettant en lumière les outils et les technologies adoptés pour son développement. Nous décrivons également le langage de programmation utilisé, ainsi que les principales bibliothèques et frameworks ayant contribué à la mise en œuvre du système et à la réalisation de ses différentes fonctionnalités. Ce chapitre vise à exposer l'aspect pratique du projet et à mettre en évidence la méthodologie suivie dans sa conception et son développement de manière structurée et rigoureuse.

3.2 Architecture Générale du Système

Ce diagramme présente l'architecture générale du système de détection de la somnolence du conducteur. Il décrit les différentes étapes du traitement, depuis la capture vidéo réalisée par la caméra jusqu'à la prise de décision finale relative à l'état du conducteur, en passant par les phases de prétraitement et d'analyse basées sur l'apprentissage profond.

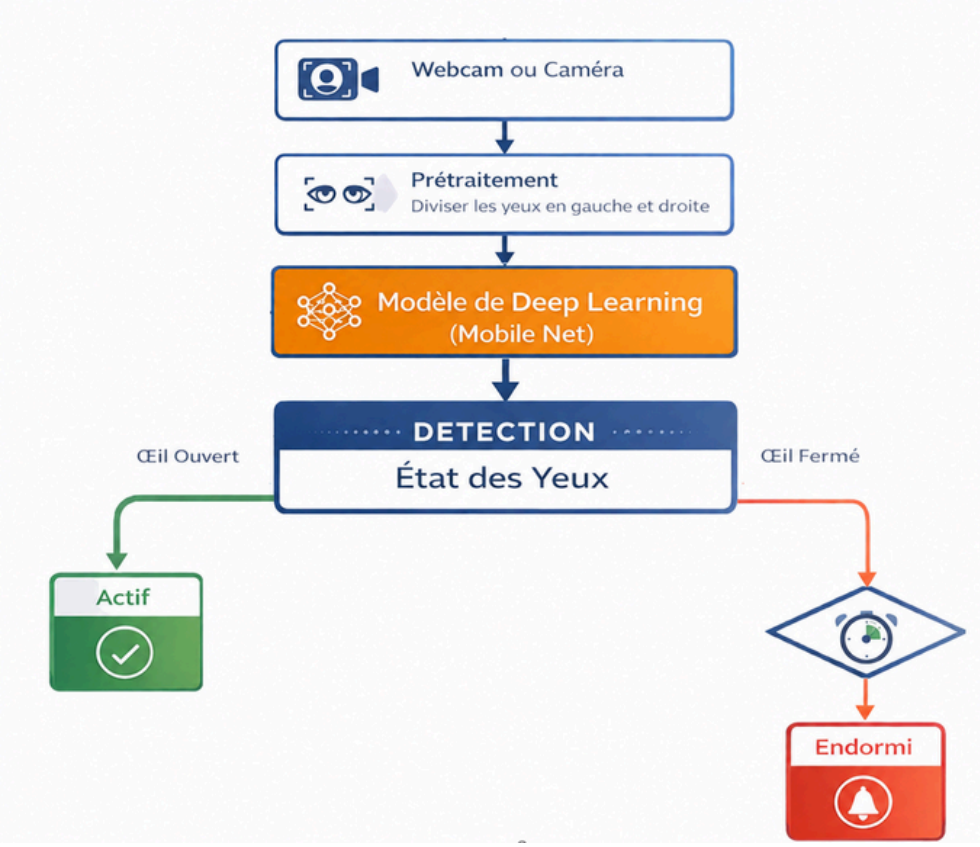


Figure 3.1 Architecture Générale du Système de notre projet

La figure 1.1 illustre l'architecture générale de notre système, en présentant la séquence des étapes adoptées dans le processus de détection de la somnolence du conducteur, lesquelles seront expliquées en détail dans ce qui suit :

1 Webcam ou Caméra

Cette étape constitue le point de départ du système. Elle consiste à utiliser la caméra de l'ordinateur afin de capturer un flux vidéo en temps réel du conducteur pendant la conduite.

L'objectif principal de cette phase est d'assurer une surveillance continue du visage du conducteur en acquérant une succession d'images (frames) qui seront transmises aux étapes de traitement ultérieures.

Plus précisément, cette étape permet de :

- Surveiller en permanence le visage du conducteur ;
- Capturer des images successives à partir du flux vidéo ;
- Transmettre les images acquises aux modules de prétraitement et d'analyse.

Le système fonctionne en temps réel (Real-Time), ce qui signifie que la caméra capture plusieurs dizaines d'images par seconde. Cette acquisition continue est essentielle pour garantir une détection rapide et précise des signes de somnolence, permettant ainsi au système de réagir immédiatement lorsqu'un état de fatigue ou d'endormissement est détecté.

2 Prétraitement

Après l'acquisition des images par la caméra, celles-ci ne sont pas directement transmises au modèle d'intelligence artificielle. Une phase de prétraitement est nécessaire afin d'améliorer la qualité des données et de ne conserver que les informations pertinentes pour l'analyse.

Cette étape comprend plusieurs opérations essentielles :

a) Détection du visage

Dans un premier temps, le système localise le visage du conducteur dans chaque image capturée. Cette opération permet d'éliminer les éléments inutiles présents dans l'environnement et de concentrer l'analyse uniquement sur la région d'intérêt, à savoir le visage du conducteur.

b) Localisation des yeux

Une fois le visage détecté, le système identifie avec précision les deux yeux du conducteur :

- L'œil droit ;
- L'œil gauche.

Cette étape est particulièrement importante, car l'état des yeux constitue l'indicateur principal utilisé pour détecter les signes de somnolence.

c) Extraction de la région des yeux

Après leur localisation, les régions correspondant aux yeux sont extraites de l'image originale. Cette opération permet de réduire la quantité de données à traiter tout en concentrant l'analyse sur les zones les plus significatives pour la détection de la somnolence.

d) Redimensionnement des images

Les images extraites sont ensuite redimensionnées afin de correspondre aux dimensions requises par le modèle MobileNet. Dans notre système, chaque image est redimensionnée à une résolution de 224 × 224 pixels, ce qui garantit la compatibilité avec l'architecture du réseau de neurones et assure un traitement efficace des données.

Grâce à ces différentes opérations de prétraitement, les images obtenues sont optimisées et préparées pour l'étape suivante, qui consiste en leur analyse par le modèle d'apprentissage profond MobileNet.

3 Phase d'apprentissage profond (MobileNet)

Au cours de cette étape, le modèle MobileNet est utilisé. Il s'agit d'un type de réseau de neurones convolutifs (Convolutional Neural Networks - CNN) développé par Google afin d'être léger, rapide et adapté aux applications fonctionnant en temps réel (Real-Time Applications), telles que la détection de la somnolence du conducteur.

MobileNet repose sur la technique de la Depthwise Separable Convolution, qui permet de réduire considérablement le nombre d'opérations de calcul par rapport aux réseaux de neurones convolutionnels traditionnels. Cela le rend particulièrement adapté aux dispositifs disposant de ressources limitées, tout en maintenant une bonne précision dans la classification des images.[13]

4 Phase de détection (Détection - État des yeux)

Après l'analyse effectuée par MobileNet, le système produit l'une des deux classifications suivantes :

Première classe : Œil ouvert

Cela signifie que l'œil est ouvert, indiquant que le conducteur est éveillé et attentif.

Deuxième classe : Œil fermé

Cela signifie que l'œil est fermé. Dans ce cas, le système ne considère pas immédiatement que le conducteur est somnolent, car la fermeture des yeux peut correspondre à un simple clignement naturel.

5 Vérification temporelle

Lorsqu'un œil fermé est détecté, le système commence à mesurer le temps.

Un minuteur est activé afin de calculer la durée de la fermeture continue des yeux.

Par exemple :

Si l'œil reste fermé pendant plus de 2 secondes, cela est considéré comme un indicateur fort de somnolence.

6 Prise de décision

Si l'œil est ouvert, le système passe à l'état :

Actif, ce qui signifie que le conducteur est éveillé et attentif. Le système continue alors la surveillance en temps réel.

Si l'œil reste fermé de manière prolongée, le système passe à l'état :

Endormi, indiquant que le conducteur est en état de somnolence ou de fatigue avancée. Dans ce cas, une alarme sonore est activée afin d'alerter le conducteur.

3.3. Réalisation

3.3.1 Langages utilisés

3.3.1.1 Python

Python est un langage de programmation de haut niveau, interprété et polyvalent, largement utilisé dans le domaine de l'intelligence artificielle et du deep learning. Il se distingue par sa simplicité syntaxique, sa lisibilité et la richesse de son écosystème de bibliothèques scientifiques. [17]

3.3.1.1.1 Bibliothèques utilisées

3.2.1.1.1.1 NumPy

NumPy est une bibliothèque fondamentale pour le calcul scientifique en Python. Elle permet de manipuler efficacement des tableaux multidimensionnels et d'effectuer des opérations mathématiques complexes. [18]

3.2.1.1.1.2 Matplotlib

Matplotlib est une bibliothèque dédiée à la visualisation de données. Elle permet de créer différents types de graphiques (courbes, histogrammes, etc.) pour analyser les résultats. [19]

3.2.1.1.1.3 TensorFlow

TensorFlow est une bibliothèque open-source développée par Google, utilisée pour la conception et l'entraînement de modèles de machine learning et de deep learning.[20]

3.2.1.1.1.4 OpenCV 

OpenCV (Open Source Computer Vision Library) est une bibliothèque spécialisée dans le traitement d'images et la vision par ordinateur, largement utilisée pour la détection d'objets et l'analyse vidéo. [21]

3.3.2 Outils utilisés

3.3.2.1 Anaconda 

Anaconda est une distribution de Python dédiée à la science des données. Elle facilite la gestion des environnements et l'installation de bibliothèques scientifiques. [22]

3.3.2.1,1 Jupyter Notebook 

Jupyter Notebook est un environnement de développement interactif permettant d'exécuter du code, d'afficher des visualisations et de rédiger du texte explicatif dans un même document. Il est particulièrement utilisé en science des données et en machine learning, car il facilite l'expérimentation, l'analyse et la présentation des résultats.
Dans notre projet, Jupyter Notebook a été utilisé via la distribution Anaconda, ce qui permet une gestion simplifiée des bibliothèques et des environnements de travail.
[23]

3.4 Résultats

3.4.1. Introduction

Ce chapitre vise à présenter les résultats expérimentaux du système de détection de la somnolence du conducteur, ainsi qu'à évaluer ses performances dans différentes situations. Plusieurs tests ont été réalisés à l'aide de la caméra de l'ordinateur afin de surveiller l'état des yeux du conducteur et de vérifier la capacité du système à distinguer entre un état normal (éveil) et un état de somnolence.
Le système repose sur des techniques de vision par ordinateur en utilisant la bibliothèque OpenCV. Il permet de détecter les points caractéristiques du visage, en particulier les yeux, afin de déterminer s'ils sont ouverts ou fermés. Lorsque les yeux restent fermés pendant une durée dépassant un seuil prédéfini, une alerte sonore est déclenchée pour avertir le conducteur.

3.4.2. Cas 1 : Conducteur en état normal



Figure 4.1 : Cas 1 : Conducteur en état normal

La figure 4.1 illustre un conducteur en état normal, avec les yeux ouverts. Le système détecte correctement le visage ainsi que les yeux, et classe cet état comme un état d'éveil (conducteur éveillé). Aucune alerte n'est déclenchée, ce qui montre que le système fonctionne correctement dans des conditions normales, sans générer de fausses alarmes.

3.4.3 Cas 2 : Fermeture des yeux

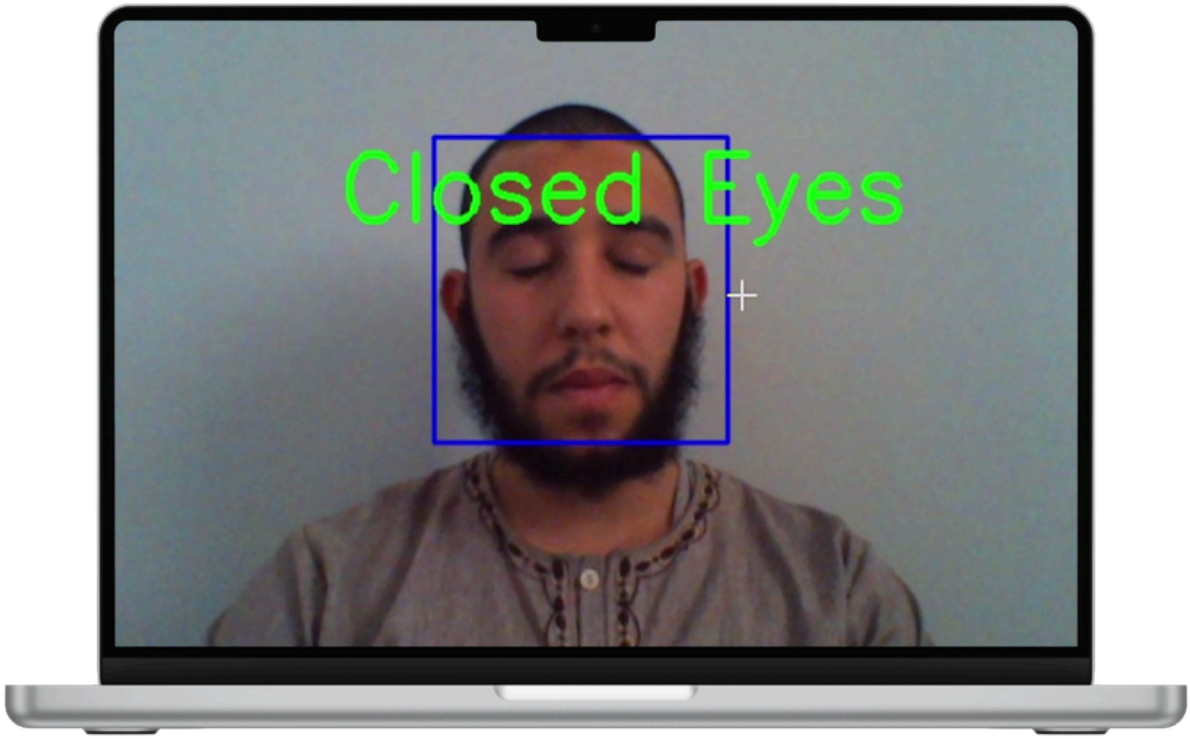


Figure 4.2 : Cas 2 : Fermeture des yeux

À travers la figure 4.2, on observe que le conducteur a fermé les yeux et que le système a correctement détecté cet état en affichant le message « Closed Eyes ». Cependant, aucune alerte sonore n'a été déclenchée. Ce comportement est essentiel afin d'éviter les détections erronées dues au clignement naturel des yeux.

3.4.4 Cas 3 : Fermeture prolongée des yeux

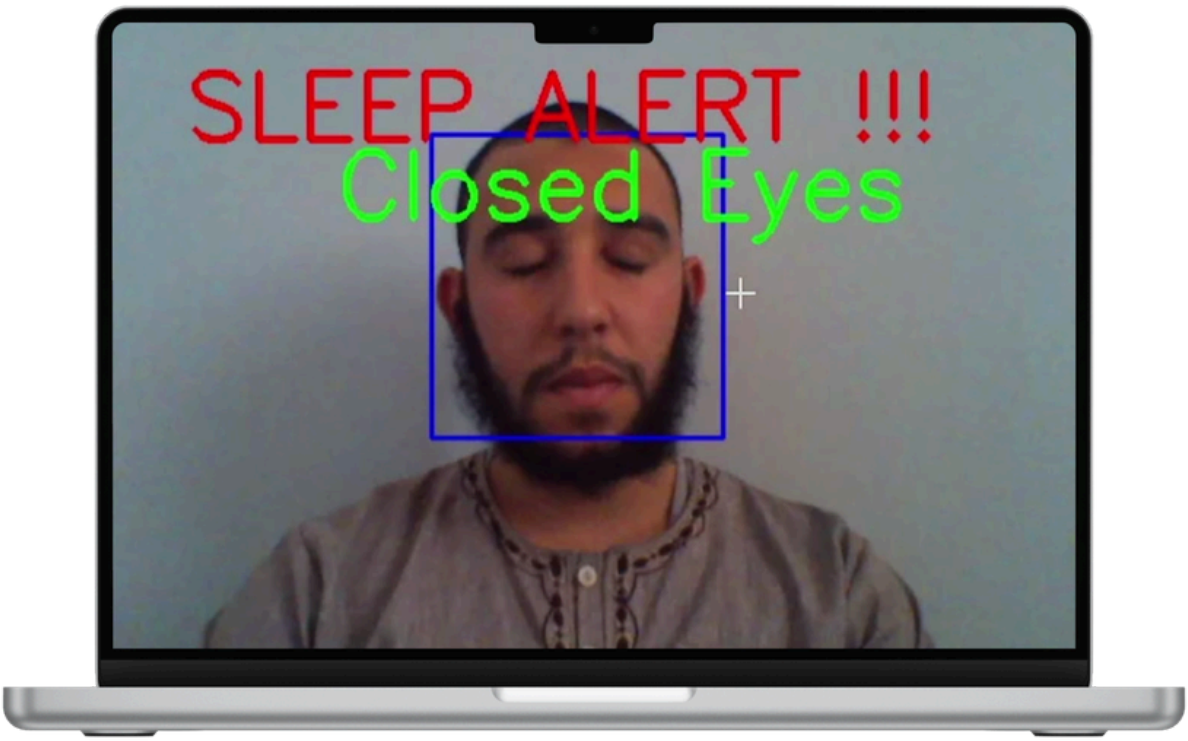


Figure 4.3 : Cas 3 : Fermeture prolongée des yeux

La figure 4.3 montre que le conducteur a gardé les yeux fermés pendant une durée prolongée, dépassant ainsi le seuil autorisé. Par conséquent, le système a affiché un message d'alerte en rouge « Sleep Alert !!! » et a déclenché une alerte sonore.

3.4.5 Cas 4 : Réouverture des yeux



Figure 4.4 Cas 4 : Réouverture des yeux

Après la réouverture des yeux, le système arrête immédiatement l'alerte sonore.Il reclassifie l'état du conducteur comme éveillé (Awake).Ce résultat montre que le système est réactif et qu'il s'adapte dynamiquement aux changements d'état du conducteur sans délai.

3.5 Conclution

En conclusion de cette section consacrée à la conception et à l'implémentation, nous avons présenté les différents outils de développement adoptés, ainsi que le langage de programmation utilisé pour la conception et la réalisation de notre projet. Cela nous a permis d'offrir une vision globale et cohérente de l'environnement technique, tout en mettant en évidence la méthodologie suivie pour la construction du système et la présentation des résultats. Ces éléments contribuent à clarifier les fondements pratiques sur lesquels repose notre travail .

Conclusion Générale

En conclusion, ce projet s'inscrit dans le domaine des systèmes intelligents appliqués à la sécurité routière, en proposant une solution basée sur l'intelligence artificielle pour la détection de la somnolence du conducteur en temps réel. Grâce à l'utilisation de techniques de traitement d'images et de réseaux de neurones, notamment via le modèle MobileNet, le système est capable d'analyser les signaux visuels liés à l'état des yeux et de détecter les situations de fatigue avec une bonne précision.

L'intégration de technologies telles que Python, OpenCV et TensorFlow a permis de développer une application fonctionnelle, rapide et adaptée aux contraintes des environnements temps réel. Le système mis en place contribue ainsi à réduire les risques d'accidents liés à la perte de vigilance, en déclenchant des alertes instantanées lorsque des signes de somnolence sont détectés.

Ce travail constitue une base solide pour de futures améliorations, notamment l'intégration d'autres indicateurs de fatigue (bâillements, posture de la tête) ou son déploiement dans des systèmes embarqués plus avancés. Ainsi, ce projet représente une contribution pertinente dans le domaine de la conduite intelligente et de l'amélioration de la sécurité routière.

Références

[1] [Online]. Available : 1 JUN 2026

Blank, S. (2021). Artificial Intelligence/Machine Learning Explained.

[2] [Online]. Available : 1 JUN 2026

Chollet, F., Kalinowski, T., & Allaire, J. J. (2022). Deep learning with R. Simon and Schuster.

[3] [Online]. Available : 1 JUN 2026

Chauhan, N. K., & Singh, K. (2018, September). A review on conventional machine learning vs deep learning. In 2018 International conference on computing, power and communication technologies (GUCON) (pp. 347-352). IEEE.

[4] [Online]. Available : 1 JUN 2026

Chen, B., & Chen, S. (2026). Deep learning. In Machine Vision Technology (pp. 273-297). Singapore: Springer Nature Singapore.

[5] [Online]. Available : 1 JUN 2026

https://fity.club/lists/u/use-imagemagick-to-quickly-and-easily-process-images-for/

[6] [Online]. Available : 1 JUN 2026

https://www.ibm.com/think/topics/deep-learning

[6_1] [Online]. Available : 1 JUN 2026

https://www.researchgate.net/figure/Evolution-timeline-of-machine-learning-and-deep-learning-based-on-literature-analysis_fig2_385740056

[6_2] [Online]. Available : 1 JUN 2026

https://www.deviantart.com/donomon/journal/GAN-s-and-Artbreeder-An-Info-Thread-878077935

[6_3] [Online]. Available : 1 JUN 2026

https://www.researchgate.net/figure/Performance-of-Deep-Learning-vs-Traditional-Learning-Techniques_fig2_403608738

[7] [Online]. Available : 1 JUN 2026

https://www.geeksforgeeks.org/artificial-intelligence/difference-between-machine-learning-and-deep-learning/

[8] [Online]. Available : 1 JUN 2026

Chen, B., & Chen, S. (2026). Deep learning. In Machine Vision Technology (pp. 273-297). Singapore: Springer Nature Singapore.

[9] [Online]. Available : 1 JUN 2026

https://www.ibm.com/think/topics/neural-networks

[10] [Online]. Available : 1 JUN 2026

https://cs231n.github.io/convolutional-networks/

[10_1] [Online]. Available : 1 JUN 2026

https://www.researchgate.net/figure/The-way-of-working-for-CNN-MathWorks-1_fig2_358656386

[10_2] [Online]. Available : 1 JUN 2026

Coşkun, M., Yıldırım, Ö., Uçar, A., & Demır, Y. (2017). An overview of popular deep learning methods. European Journal of Technique (EJT), 7(2), 165-176.

[10_3] [Online]. Available : 1 JUN 2026

https://studentsxstudents.com/training-a-convolutional-neural-network-cnn-on-cifar-10-dataset-cde439b67bf3

[11] [Online]. Available : 1 JUN 2026

https://cs231n.github.io/neural-networks-1/

[12] [Online]. Available : 1 JUN 2026

https://cs231n.github.io/transfer-learning/

[13] [Online]. Available : 1 JUN 2026

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

[14] [Online]. Available : 1 JUN 2026

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).

[15] [Online]. Available : 1 JUN 2026

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[16] [Online]. Available : 1 JUN 2026

https://mrl.cs.vsb.cz/eyedataset.html

[17] [Online]. Available : 1 JUN 2026

https://www.python.org/doc/

[18] [Online]. Available : 1 JUN 2026

https://numpy.org/doc/

[19] [Online]. Available : 1 JUN 2026

https://matplotlib.org/stable/users/index.html

[20] [Online]. Available : 1 JUN 2026

https://www.tensorflow.org/learn

[21] [Online]. Available : 1 JUN 2026

https://docs.opencv.org/

[22] [Online]. Available : 1 JUN 2026

https://www.anaconda.com/docs/main

[23] [Online]. Available : 1 JUN 2026

https://docs.jupyter.org/en/latest/

[24] [Online]. Available : 1 JUN 2026

https://www.researchgate.net/figure/Classification-with-Flatten-and-Fully-Connected-layers_fig1_380977680

[25] [Online]. Available : 1 JUN 2026

https://www.researchgate.net/figure/Figure-B4-Plot-of-the-Softmax-activation-function-For-a-definition-of-its-functional_fig81_339240163

[26] [Online]. Available : 1 JUN 2026

https://medium.com/@yashwanths_29644/deep-learning-series-23-transfer-learning-4cf6b2157a81

[27] [Online]. Available : 1 JUN 2026

https://www.instagram.com/p/C5Sb_AZNMEX/