UNIVERSITÉ MOHAMMED V de Rabat

Faculté des Sciences

Université Mohammed V
Faculté des Sciences
Rabat

Département d'Informatique

Filière Licence fondamentale

en Sciences Mathématiques et Informatique

PROJET DE FIN D'ÉTUDES

intitulé :

INTELLIGENCES ARTIFICIELLE: DETECTION DE LA FATIGUE CHEZ LES CONDUCTEURS AUTOMOBILISTES

Présenté par :

Ghizlane Bouazama

Amine Mohachtou

soutenu le 22 Septembre 2022 devant le Jury

Pr.Ahmed Drissi El maliani	Faculté des Sciences de Rabat	*Encadrant*
Pr. Lotfi Dounia	Faculté des Sciences de Rabat	*Examinatrice*

Année universitaire 2019-2020

Table des matières

Liste des figures

Liste des tableaux

Remerciement

Ma gratitude s'adresse tout particulièrement à mon encadrant, Monsieur ***Drissi el Maliani Ahmed,*** *qui a eu l'amabilité de m'accorder son temps précieux, ses conseils, son soutien et ses suggestions.*

Je remercie également tous les professeurs de la FSR pour la qualité de leur enseignement. Ils ont contribué à l'enrichissement de ce travail en nous faisant partager leurs connaissances tout au long de ces années d'études.

Je tiens à remercier également, ***Khadija Sifou****. Ma mère, mon vrai binôme durant mes années d'études, cette merveilleuse femme qui a œuvré pour ma réussite, par son amour, son soutien, ses prières, tous les sacrifices consentis et ses précieux conseils, pour toute son assistance et sa présence dans ma vie, et les longues nuits blanches qu'elle a fait de bon cœur pour m'encourager et me remonter le moral, pour m'avoir donnée la force dans les moments difficiles d'éditer ce mémoire et le finir. Qu'elle puisse recevoir à travers ce travail aussi modeste soit-il, l'expression de mes sentiments et de mon éternelle gratitude.*

Mon père, ***LahcenBouazama*** *qui peut être fier et trouver ici le résultat de longues années de sacrifices et de privations pour m'aider à avancer dans la vie. Puisse Dieu faire en sorte que ce travail porte son fruit ; Merci pour les valeurs nobles, l'éducation et le soutien permanent venu de toi.*

Un grand merci à ma petite famille. Mon grand frère ***Ayoub,*** *merci d'être mon frère et mon meilleur ami, ma petite sœur****Hajar*** *et mon petit frère* ***Ali,*** *je vous aime...Mes remerciements vont également à mes tantes qui m'ont accueilli durant mes années d'études à Rabat, Tati* ***Aicha****, Tati* ***Fatima*** *et* ***Zahra Sifou****.*

Finalement, je remercie de tout cœur mon cher ***Khalid Larhlid****,mes cousins et cousines et toute personne ayant contribué de près ou de loin à la réalisation de ce travail.*

Résumé

Le monde va vers l'informatisation de tous les secteurs .De nos jours , plusieurs applications sont basées sur l'intelligence artificielle et le deeplearning qui consistent à programmer un ensemble d'algorithmes conférant à une machine des capacités d'analyse et de décision lui permettant de s'adapter intelligemment aux situations en faisant des prédictions à partir de données déjà acquises.

Dans notre travail, nous nous intéressons à ce type d'applications en se concentrant sur la détection facial du visage d'un conducteur tout en se basant sur la détection du clignement de ses yeux, l'angle de sa tête par rapport à la chaise et finalement on va détecter les grimaces qui réfèrent aux symptômes de la somnolence.

Tout d'abord on a introduit l'intelligence artificielle en général, l'apprentissage machine, l'apprentissage profond, le data science et leurs relation avec l'IA ensuite, on a entamé la vision par ordinateur, le traitement d'images en niveau de gris et les images colorées et puis on a définit la reconnaissance faciale et son fonctionnement, finalement on a discuté le code de l'application pour en finir par une conclusion générale.

Abstract

The world seems to be moving towards the computerization of all sectors. Nowadays, several applications are based on artificial intelligence and deep learning which consist in programming a set of algorithms giving a machine the capacities of analysis and decision. Allowing it to intelligently adapt to situations by making predictions from data already acquired.

In our work, we will be detecting a car driver‘s face to make sure he is not sleepy, first of all we are going to detect the eyes blinking, the angle of his head in and the chair. and finally we will detect the grimaces that refer to the symptoms of drowsiness.

We first introduced artificial intelligence in general, machine learning, deep learning, data science and their relationship with AI then, computer vision, image processing at the level of gray and colored images and then we defined the facial recognition and how it works, finally we discussed the application’s programsbefore we end it with a general conclusion.

Introduction générale

Après la vitesse et l'alcool au volant, l'endormissement est l'une des principales causes d'accidents. Plus précisément, un accident mortel sur trois est dû à la somnolence sur les voies rapides. Les constructeurs automobiles travaillent à réduire ce taux en utilisant des systèmes électroniques de plus en plus élaborés.

Les chiffres de la Sécurité routière en matière de mortalité routière mettent en évidence deux causes principales : une vitesse trop élevée et un taux d'alcoolémie au-dessus de la moyenne. Un peu derrière, la somnolence et la fatigue au volant sont également d'importants facteurs accident gènes. Ainsi, au Maroc, un accident mortel sur trois est dû à l'endormissement.. Il est prouvé que prendre la route en état de fatigue ou se forcer à rester éveillé longtemps a des effets négatifs sur les réflexes du conducteur, de la même façon que s'il avait consommé de l'alcool.

Fort de ce constat, les autorités ont renforcé leur attention sur les autoroutes, voies où le risque de somnolence est le plus élevé en raison de la monotonie du trajet. Les panneaux d'affichage sont par exemple mis à contribution pour passer des messages de prévention du style "Faites une pause en cas de fatigue" ou "Faites le plein de vigilance" ; messages hélas souvent ignorés des conducteurs.

Le salut viendra sans doute des constructeurs automobiles, grâce au progrès de l'électronique embarquée et à l'évolution des tableaux de bord. C'est ainsi qu'apparaît dans les années 2000 le premier message de prévention sous forme d'une tasse de café. Il s'agit de la fonction Attention Assist : Pause ! qui se déclenche après deux heures de route sans halte.

Chapitre 1 : l'intelligence artificielle

1. Motivation

1.1 Intelligence artificielle

On pourrait dire que l'intelligence artificielle (IA) est un ensemble de techniques permettant à des machines d'accomplir des tâches et de résoudre des problèmes normalement réservés aux humains et à certains animaux.

Les tâches relevant de l'IA sont parfois très simples pour les humains, comme par exemple reconnaître et localiser les objets dans une image, planifier les mouvements d'un robot pour attraper un objet, ou conduire une voiture. Elles requièrent parfois de la planification complexe. Les tâches les plus compliquées requièrent beaucoup de connaissances et de sens commun, par exemple pour traduire un texte ou conduire un dialogue.

Depuis quelques années, on associe presque toujours l'intelligence aux capacités d'apprentissage. C'est grâce à l'apprentissage qu'un système intelligent capable d'exécuter une tâche peut améliorer ses performances avec l'expérience. C'est grâce à l'apprentissage qu'il pourra apprendre à exécuter de nouvelles tâches et acquérir de nouvelles compétences.

1.1.1 Les trois niveaux de l'intelligence artificielle

L'Intelligence Artificielle peut prendre différentes formes. Les experts s'accordent sur une classification en trois catégories de l'Intelligence Artificielle.

a. L'Intelligence Artificielle faible (ou Artificial Narrow Intelligence, ANI)

L'intelligence Artificielle dite "Faible" n'est pas destinée pour correspondre ou excéder les capacités d'êtres humains, l'IA faible ne pense pas, elle exécute seulement la tâche qui lui a été incombé d'accomplir.

Un bon exemple est un logiciel comme Deep Blue, fait pour jouer aux échecs. Deep Blue est capable de battre un champion mondial aux échecs, c'est justement pour cette

raison qu'il a été fait: c'est sa tâche. Mais on ne peut pas effectivement dire que l'ordinateur "comprend" ce qu'il fait. Il fait quelque-chose car il a été programmé de cette façon, si jamais il avait été programmé différemment alors ça aurait été différent.

On ne peut pas dire que c'est un IA "intelligent" comme l'est un homme, il exécute sa tâche parce que l'homme lui a donné les moyens de le faire. Leur performance est élevée uniquement dans leur domaine d'activité. On peut dire que la très grande majorité des IA existants aujourd'hui sont "faibles». Il s'agit du niveau 1 de l'intelligence artificielle. Le « machine learning » est souvent associé à l'Artificial Narrow Intelligence.

b. L'Intelligence Artificielle Générale (AGI)

Une AGI est multifonctions, elle peut effectuer plusieurs tâches et résoudre différents types de problèmes, raisonner, penser de manière abstraite, appréhender des idées complexes, apprendre rapidement, se nourrir de ses expériences. La création par l'homme d'intelligences artificielles générales est un projet ambitieux, mais non encore réalisé. Nous en sommes toujours à la phase de recherche. Pour certains experts, il faudra attendre des dizaines d'années avant que la première intelligence artificielle voit le jour. C'est le niveau 2 de l'intelligence artificielle.

c. La Super-Intelligence Artificielle (Artificial Super Intelligence, ASI)

Avec l'ASI, on franchit encore un pas. Une super-intelligence artificielle est une intelligence plus perfectionnée, plus performante, plus complexe que l'intelligence humaine, une intelligence capable de créativité scientifique et artistique, douée de capacités sociales et même d'une forme de 'sagesse'. Pour certaines personnes, les ASI sauveront l'humanité. Pour d'autres, elles en constituent le grand péril. Les opinions sur les ASI sont très divergentes. Pour la plupart des experts, l'intelligence artificielle générale n'est qu'une étape, un tremplin vers la super-intelligence artificielle. Il y a continuité entre les deux formes.

1.2 L'apprentissage machine

De manière générale, un programme informatique tente de résoudre un problème pour lequel nous avons la solution. Par exemple : calculer la moyenne générale des étudiants, classer les étudiants selon leur moyenne etc…

Pour certains problèmes, nous ne connaissons pas de solution exacte et donc nous ne pouvons pas écrire de programme informatique. Par exemple : reconnaître automatiquement des chiffres écrits a la main a partir d'une image scannée Pour ces problèmes il est facile d'avoir une base de données regroupant de nombreuses instances du problème,

L'apprentissage automatique consiste alors a programmer des algorithmes permettant d'apprendre automatiquement de données et d'expériences passées.

On distingue deux types de problèmes en ML :

1.2.1 Supervised Learning (Apprentissage Supervisé)

Avec l'apprentissage supervisé, la machine peut apprendre à faire une certaine tâche en étudiant des exemples de cette tâche. Par exemple, elle peut apprendre à reconnaître une photo de chien après qu'on lui ait montré des millions de photos de chiens. Ou bien, elle peut apprendre à traduire le français en chinois après avoir vu des millions d'exemples de traduction français-chinois.

D'une manière générale, la machine peut apprendre une **relation** **f:x->y** qui relie **x** à **y** en ayant **analysé** des millions d'exemples d'associations.

L'apprentissage supervisé fonctionne en 4 étapes :

a. **Importer un Dataset (x,y) qui contient nos exemples:**

La première étape d'un algorithme de Supervised Learning consiste donc à importer un Dataset qui contient les exemples que la machine doit étudier.

Ce Dataset inclut toujours 2 types de variables :

- Une variable objectif (target)
- Une ou plusieurs variables caractéristiques (features)

Par exemple, imaginons qu'on visite une série d'appartements différentes. Pour chaque appartement, on note dans un tableau Excel le prix **y** et les caractéristiques **x** de l'appartement (la surface, la qualité, ville, etc.).

Exemple de Dataset sur des appartements

Target y | Features x_1 x_2 x_3

Prix	Surface	Qualité	Adresse postale
313,000	90	3	95000
720,000	110	5	93000
250,000	40	4	44500
290,000	60	3	67000
190,000	50	3	59300
...	...	...	...

m (nombre de lignes) — n (nombre de features)

Par convention:
m: nombre d'exemples
n: nombre de features

Tableau 1 : Exemple de Dataset sur des appartements

Par convention, on dit que notre Dataset contient **m** exemples, c'est-à-dire **m** lignes. Si on visite 6 appartements, alors **m=6**.

Par convention, on note également **n** le nombre de features dans notre Dataset, c'est-à-dire le nombre de colonnes (hormis la colonne **y**). Si on note 3 caractéristiques pour ces appartements (Surface, qualité, ville), alors **n=3**.

Quand on développe un programme de vision par ordinateur, les *features* de notre Dataset peuvent être les pixels présents sur l'image. Un Dataset d'images de 8×8 pixels donne donc 64 *features* (n=64). Chacune de ces *features* correspondra à la valeur du pixel (un pixel noir = 1, un pixel blanc = 0).

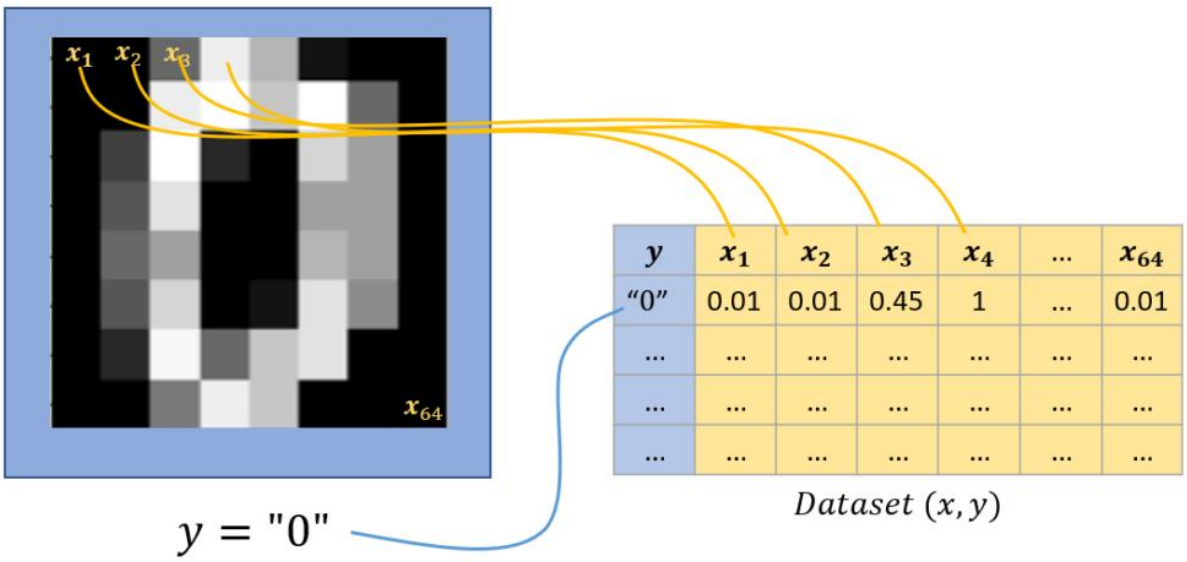


Figure 1 :Représentation des pixels sur le tableau

Notation : pour désigner une cellule de notre tableau, on note $X\binom{ligne}{colonne}$ c'est-à-dire que pour désigner la qualité du 3ième appartement qu'on a visité, on écrit : $X\binom{3}{2}$

$$Dataset\ (x, y)$$

y	x_1	x_2	x_3	...	x_n
$y^{(1)}$	$x_1^{(1)}$	$x_2^{(1)}$	$x_3^{(1)}$	...	$x_n^{(1)}$
$y^{(2)}$	$x_1^{(2)}$	$x_2^{(2)}$	$x_3^{(2)}$	...	$x_n^{(2)}$
$y^{(3)}$	$x_1^{(3)}$	$x_2^{(3)}$	$x_3^{(3)}$	...	$x_n^{(3)}$
...	...	...	...	...	...
$y^{(m)}$	$x_1^{(m)}$	$x_2^{(m)}$	$x_3^{(m)}$	...	$x_n^{(m)}$

Tableau 2 :Représentation de Dataset (x ,y)

Avec un tel Dataset, il devient possible de prédire de nouvelles valeurs **y** à partir de valeurs de **x** en développant un modèle, ce qui nous amène à la deuxième étape dans la résolution d'un problème de Supervised Learning :Développement d'un modèle.

b. Développer un Modèle aux paramètres aléatoires

Le modèle est en quelque sorte le cœur du programme, c'est lui qui va effectuer la tâche qu'on cherche à accomplir, par exemple reconnaître un animal sur une photo ou prédire le prix d'un appartement.

Qu'est ce qu'un modèle ?

Un modèle est une représentation simplifiée de la réalité, que l'on peut utiliser pour prédire ce qui se passerait dans certaines conditions. Ça peut être un dessin, une équation physique, une fonction mathématique, une courbe… bref, n'importe quelle représentation.

Par exemple, si je lâche une pomme de 100 g depuis une hauteur de 4 mètres, en combien de temps tombera-t-elle sur terre ? On peut prédire cela avec les équations de Newton. Dans ce cas, notre modèle est déterministe : il donnera toujours la même réponse sous les mêmes conditions.

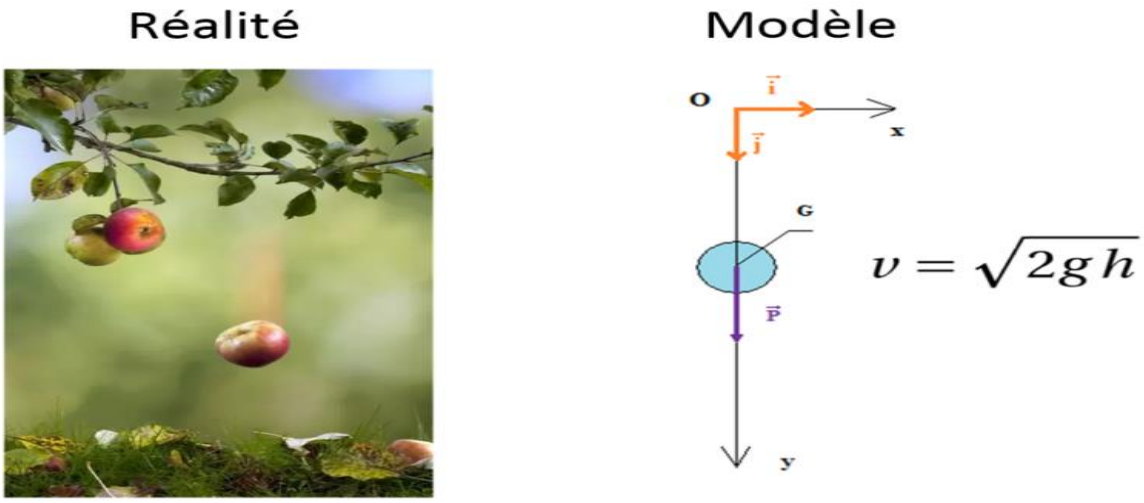


Figure 2 : Exemple de la représentation de la réalité et le modèle

Mais pour prédire le prix d'un appartement en fonction de toutes ses caractéristiques, quelle est l'équation mathématique à entrer dans la machine ? Et quelle est l'équation pour reconnaître un chat sur une photo?

La solution : Laisser la machine trouver le modèle qui correspond le mieux à notre Dataset (x,y) , c'est de l'apprentissage supervisé.

c. Développer une Fonction Coût qui mesure les erreurs entre le modèle et le Dataset

Pour savoir quel modèle est le meilleur parmi 2 candidats, il faut les évaluer. Pour cela, on mesure l'erreur entre un modèle et le Dataset, et on appelle ça la Fonction Coût.

Dans le cas d'une régression, on peut par exemple mesurer l'erreur entre la prédiction du modèle $f(x)^i$ et la valeur $(y)^i$ qui est associée à ce $(x)^i$ dans notre Dataset.
C'estsimilaire à l'idée de mesurer la distance entre la flèche ($f(x)^i$) et le centre de la cible dans le jeu de tir à l'arc, qui n'est autre que le point ($(y)^i$) qu'elle est sensée atteindre.

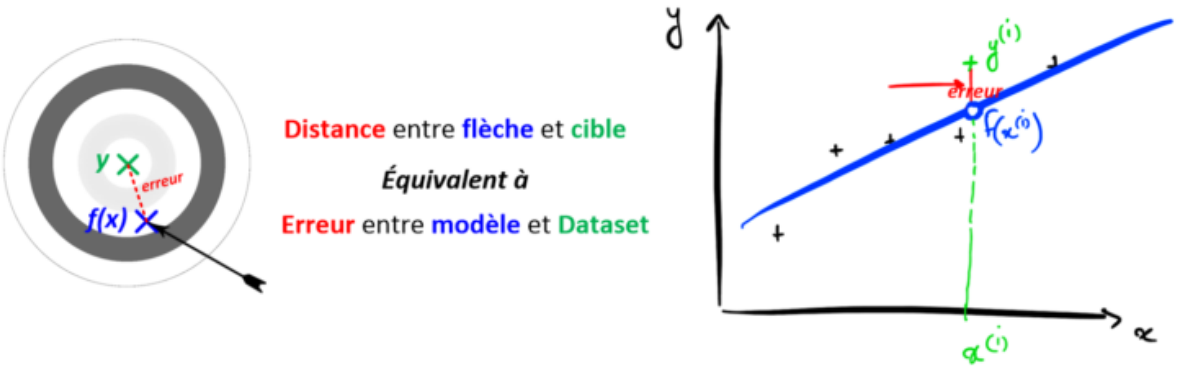


Figure 3:Exemple de la Fonction Coût qui mesure les erreurs entre le modèle et le Dataset

Maintenant, dans le cas d'une classification, on peut construire notre Fonction Coût en mesurant le nombre d'exemples du Dataset que notre modèle aura mal classé avec sa frontière de décision.

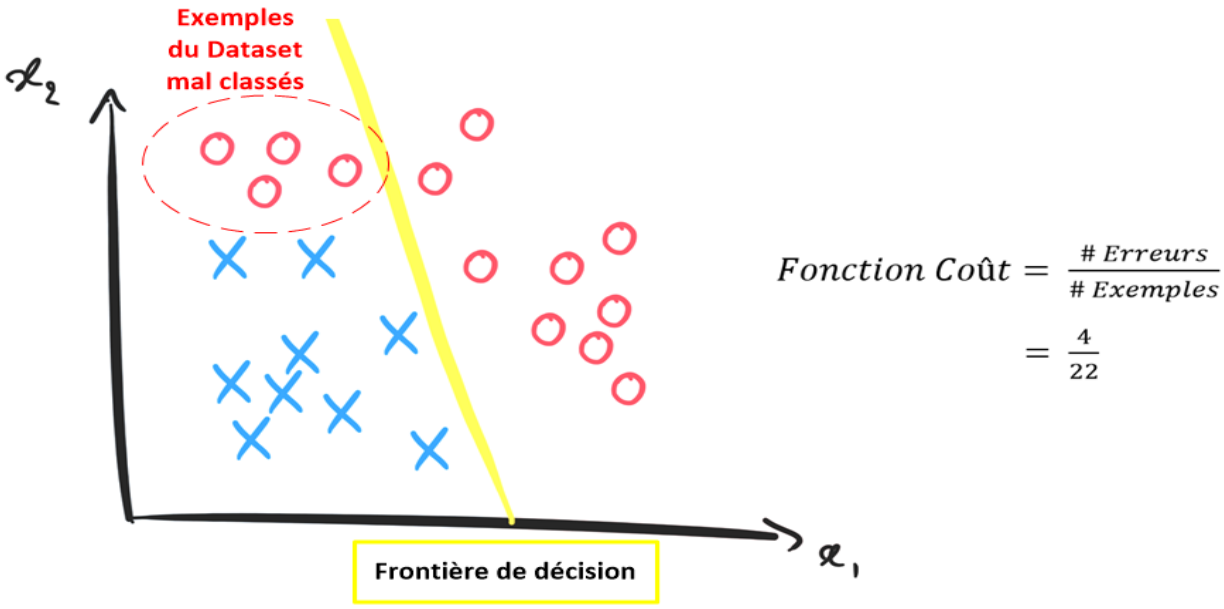


Figure 4: Frontière de décision

Maintenant il est temps de passer à l'étape la plus importante en Machine Learning, celle qui donne vie à notre programme, l'étape d'apprentissage.

d. Développer un Algorithme d'apprentissage pour trouver les paramètres du modèle qui minimisent la Fonction Coût.

Logiquement, avoir un bon modèle, c'est avoir un modèle qui de petites erreurs. Ainsi, en SupervisedLearning, la machine cherche les paramètres de modèle

qui minimisent la Fonction Coût. C'est ça qu'on appelle l'apprentissage et c'est la phase de la plus importante du machingLearning. Pour trouver les paramètres qui minimisent la fonction Coût, il existe un paquet de stratégies, on pourrait par exemple développer un algorithme qui tente au hasard plusieurs combinaisons de paramètres, et qui retient la combinaison avec la Fonction Cout la plus faible, cependant cette stratégie est assez inefficace la plupart du temps.

Une autre stratégie, très populaire en Machine Learning, est de considérer la Fonction Coût comme une fonction convexe, c'est-à-dire une fonction qui n'a qu'un seul minimum, et de chercher ce minimum avec un algorithme de minimisation appelé Gradient Descent. Cette stratégie apprend de façon graduelle, et assure de converger vers le minimum de la fonction Coût (si convexe).

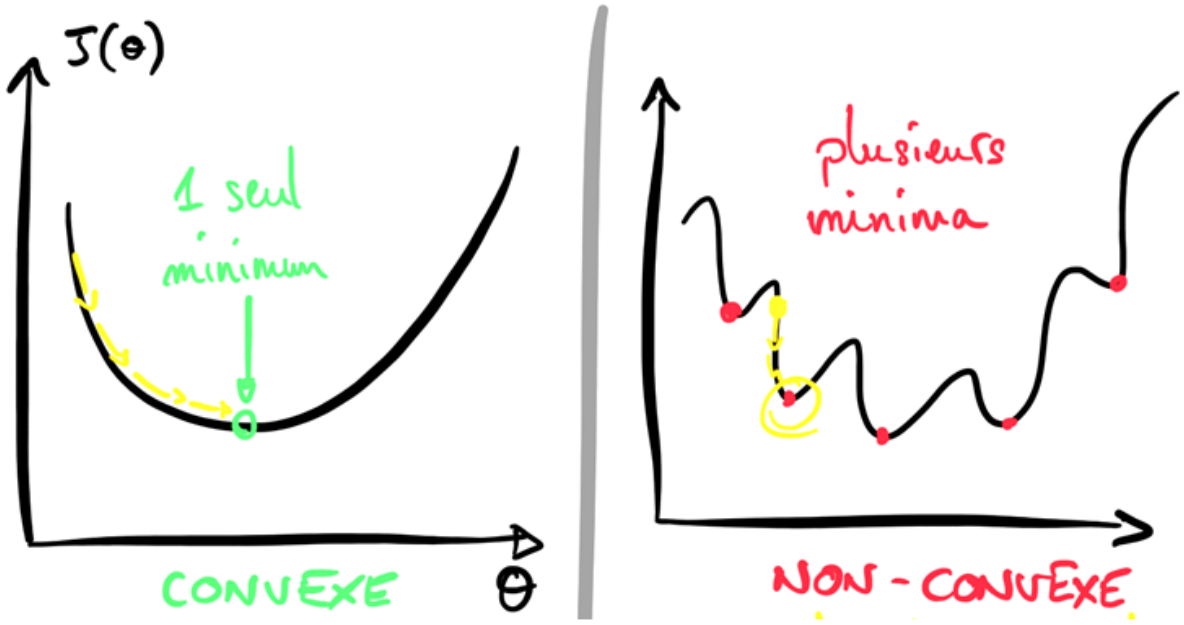


Figure 5: La Fonction Cout convexe et non convexe

Dans le jargon, on appelle cette étape la phase d'entrainement du modèle. La machine choisit des paramètres pour le modèle, puis évalue sa performance (Fonction Coût) puis cherche des paramètres qui peuvent améliorer sa performance actuelle, etc.
Une fois la phase d'entrainement terminée… on a un modèle de MACHINE LEARNING.

1.2.2 Unsupervised Learning (Apprentissage Non supervisé)

Une autre méthode d'apprentissage pour développer des programmes de Machine Learning est l'apprentissage non–Supervisé (Unsupervised Learning). Cette méthode est utilisée quand notre Dataset ne contient pas d'exemples qui indiquent ce que l'on cherche.

Exemple: Est-ce qu'on peut regrouper ces images en 2 familles selon leur ressemblance?

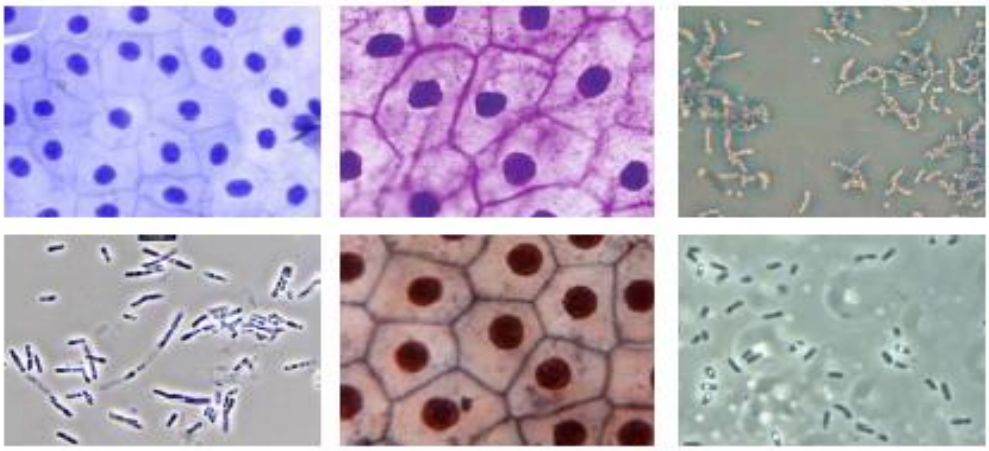


Figure 6 : cellules animales, de bactéries et de protéines

Oui bien sûr et c'est très simple. Nul besoin de savoir s'il s'agit de cellules animales, de bactéries ou de protéines pour apprendre à classer ces images**.** En regardant les images notre cerveau a en fait reconnu des **structures communes** dans les données que nous lui avons montrées.

Dans l'apprentissage non-supervisé, on dispose ainsi d'un Dataset x sans variable y, et la machine apprend à reconnaître des structures dans les données **x** qu'on lui montre.

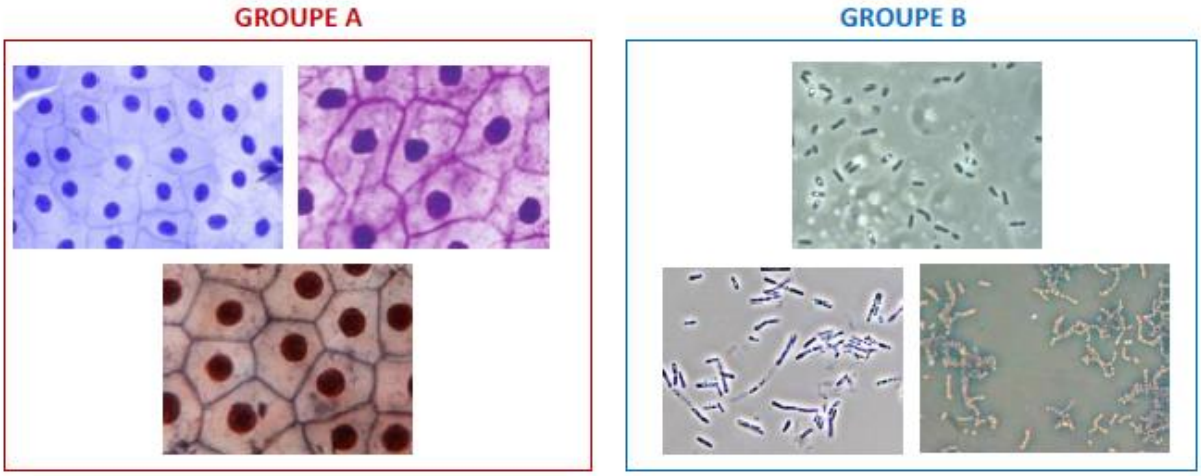


Figure 7 :Regroupement des images de la figure 6 en deux familles

On peut ainsi regrouper des donnés dans des clusters (c'est le **Clustering**).La figure suivante montre la difference entre l'apprentissage supervise et non supervise:

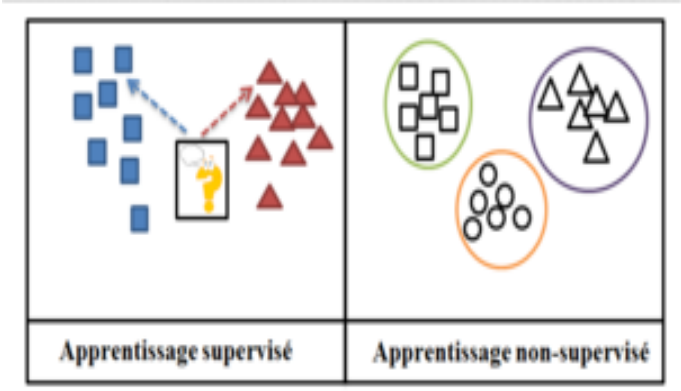


Figure 8 :L'apprentissage supervisé et non supervisé

1.3 L'apprentissage profound

L'apprentissage profond ou DeepLearningen en anglais est un réseau de neurones artificiels convolutifs. Depuis les débuts de l'IA, il y a cette ambition de reproduire le cerveau humain, ou plus précisément de s'en inspirer. Pour ce faire, les chercheurs se sont d'abord intéressés de près au fonctionnement des cerveaux biologiques. Ils ont étudié notamment le fonctionnement des neurones. Ces neurones, qui composent notre cerveau, communiquent entre eux grâce à des synapses.

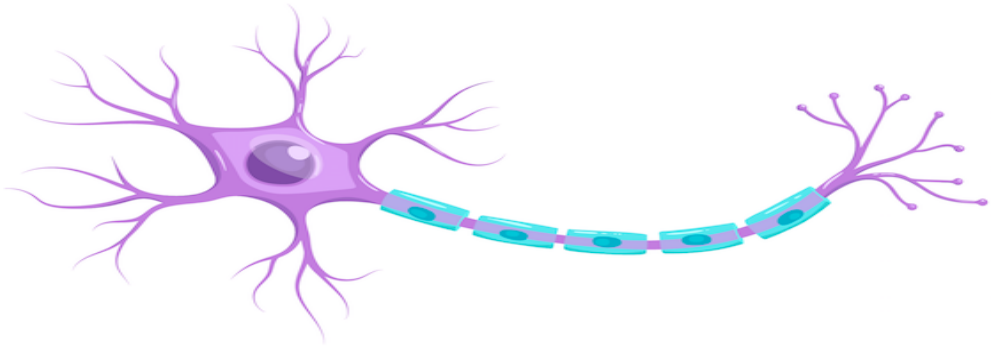


Figure 9: Représentation d'un neurone (une inspiration)

On a donc créé des algorithmes qui reprennent cette architecture neuronale. Bien sûr, les neurones sont ici une inspiration, qu'on a ensuite adaptée à l'intelligence artificielle.

La **convolution** n'est rien d'autre qu'un **filtrage** de notre image. Plutôt que de traiter l'image en un bloc, nous allons la diviser en différents carrés que nous allons analyser séparément.

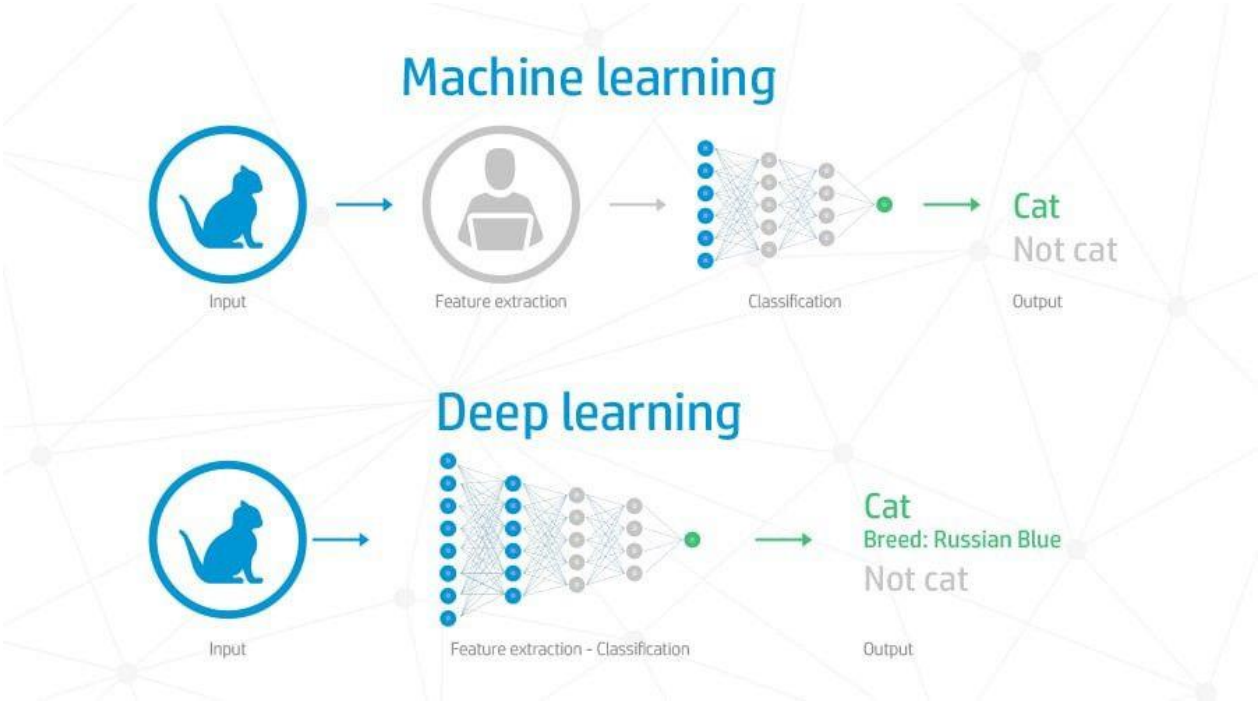


Figure 10 : Le concept de l'apprentissage machine et l'apprentissage profond

La plupart des modèles d'apprentissage en profondeur sont basés sur des réseaux de neurones artificiels. Un réseau de neurones est essentiellement constitué de couches sur des couches de nœuds qui se connectent les uns aux autres de manière magique. Lorsque vous avez plus de 1 ou 2 couches entre vos couches d'entrée et de sortie, vous avez un réseau profond,ce qui est vraiment cool, c'est que lorsque vous formez le réseau, il trouve en quelque sorte comment s'organiser pour reconnaître les visages (par exemple). Il peut affecter le premier calque au regroupement de pixels, le second à la détection des contours, le troisième à la compréhension du nez, et ainsi de suite… mais il le comprend tout seul.

Notre réseau est composé de nombreux neurones, qui sont groupés en plusieurs couches de neurones. Je vous propose d'en détailler trois types: la **couche d'entrée,** les **couches intermédiaires** et la **couche de sortie.**

- ✓ **La couche d'entrée :** C'est elle qui reçoit l'information de notre image. Notre cliché est composé de millions de points appelés **pixels.** Chaque pixel alimente un neurone de la couche d'entrée.
- ✓ **Les couches intermédiaires :** Les couches intermédiaires peuvent être nombreuses (de quelques dizaines à plusieurs centaines). Les neurones d'une couche sont connectés aux neurones de la couche suivante. Les neurones d'une couche interagissent avec ceux de la couche suivante en réalisant des **opérations mathématiques** élémentaires comme l'addition, la soustraction, la multiplication et la division.

- ✓ **La couche de sortie** : est la dernière couche qui porte le résultat final.

1.4 Data Science, ou la science des données, discipline liée à l'IA

Les données, ce sont ces informations qui sont enregistrées pour être utilisées par les programmes informatiques.

Le big Data : À l'échelle de la société, nous produisons collectivement de très nombreuses données.Pour avoir une idée plus claire,on cite qu'à chaque minute :

- Google est sollicité près de 4 millions de fois ;
- 4,5 millions de vidéos sont visionnées sur YouTube ;
- 188 millions d'emails sont échangés.

Ce sont toutes ces données qui forment le concept de Big Data. On pourrait le traduire par "données massives".

Le concept de Big Data a été forgé pour désigner ce phénomène d'explosion des données. L'élément-clé dans le Big Data, c'est le volume de données qui est considérable.

Ensuite, il faut garder à l'esprit que les données du Big Data sont variées. Il peut s'agir de nombres, de texte mais aussi de vidéo, d'audio, etc. Ces données ne concernent pas uniquement le monde de l'Internet mais sont également issues de capteurs dans le monde "physique". Dans les transports par exemple, un bus peut enregistrer régulièrement sa position pour assurer un service fluide pour les usagers.

L'intelligence artificielle est fortement liée au Big Data. Seuls des algorithmes sophistiqués sont aujourd'hui capables de traiter autant d'informations en instantanée. Ensuite, comme toute loi de statistiques et des probabilités, plus l'intelligence artificielle a de données à traiter, plus elle a de chance d'en tirer une tendance générale.

Au final, on peut représenter les différents champs d'étude comme imbriqués ainsi :

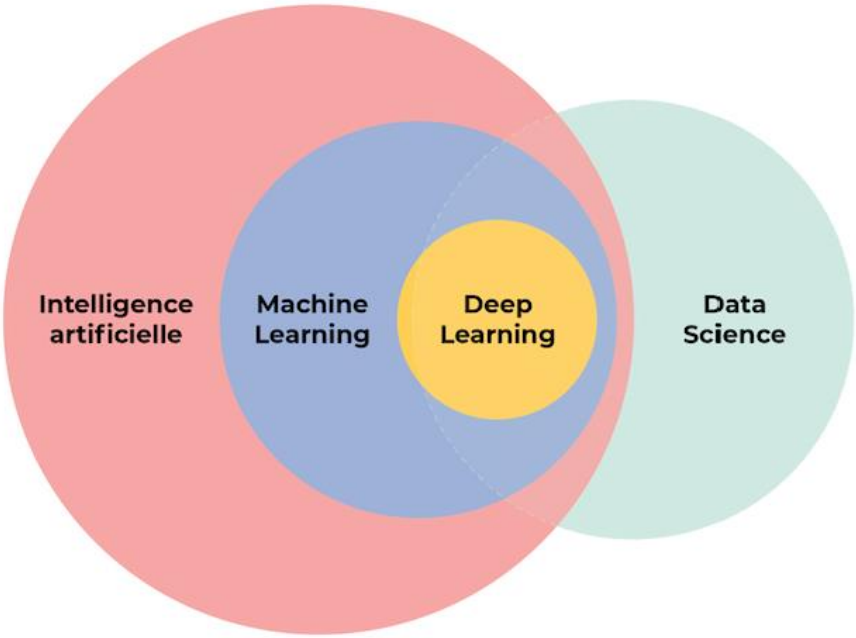


Figure 11 : Une représentation de l'organisation des différentes disciplines présentées

Chapitre 2: Vision par ordinateur

1. Technologie et outils de de développement

1.1 Python

Python est un langage de programmation interprété (sans phase de compilation) orienté objet, multi-paradigmes et multi-plateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. Il est doté d'un typage dynamique fort, d'une gestion automatique de la mémoire par ramasse-miettes et d'un système de gestion d'exceptions. Il est ainsi similaire à Perl, Ruby, Scheme, Smalltalk et Tcl. Le langage Python est placé sous une licence libre et fonctionne sur la plupart des platesformes informatiques, des supercalculateurs aux ordinateurs centraux, de Windows à Unix avec notamment GNU/Linux en passant par macOS, ou encore Android, iOS, et aussi avec Java ou encore .NET. Il est conçu pour optimiser la productivité des programmeurs en offrant des outils de haut niveau et une syntaxe simple à utiliser. Il est également apprécié par certains pédagogues qui y trouvent un langage où la syntaxe, clairement séparée des mécanismes de bas niveau, permet une initiation aisée aux concepts de base de la programmation.

1.2 OpenCV

OpenCV (Open Source Computer Vision Library), est une bibliothèque de fonctions de programmation principalement destinées à la vision par ordinateur en temps réel. Initialement développé par Intel, il a ensuite été soutenu par Willow Garage puis Itseez (qui a ensuite été acquis par Intel []. La bibliothèque est multiplateforme et gratuite pour une utilisation sous la licence BSD open-source.

1.3 Dlib

Dlib est une bibliothèque logicielle multiplateforme à usage général écrite dans le langage de programmation C ++. Sa conception est fortement influencée par les idées issues de la conception par contrat et par l'ingénierie logicielle basée sur les composants. Il s'agit donc avant tout d'un ensemble de composants logiciels indépendants. Il s'agit d'un logiciel open source publié sous une licence logicielle Boost.

Depuis le début du développement en 2002, Dlib s'est développé pour inclure une grande variété d'outils. À partir de 2016, il contient des composants logiciels pour gérer la mise en réseau, les threads, les interfaces utilisateur graphiques, les structures de données, l'algèbre linéaire, l'apprentissage automatique, le traitement d'images, l'exploration de

données, l'analyse XML et de texte, l'optimisation numérique, les réseaux bayésiens et de nombreuses autres tâches. Ces dernières années, une grande partie du développement s'est concentrée sur la création d'un large éventail d'outils d'apprentissage automatique statistique et en 2009, Dlib a été publié dans le Journal of Machine Learning Research. Depuis lors, il a été utilisé dans un large éventail de domaines.

1 .4 NumPy

NumPy est une extension du langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux. Plus précisément, cette bibliothèque logicielle libre et open source fournit de multiples fonctions permettant notamment de créer directement un tableau depuis un fichier ou au contraire de sauvegarder un tableau dans un fichier, et manipuler des vecteurs, matrices et polynômes. NumPy est la base de SciPy, regroupement de bibliothèques Python autour du calcul scientifique.

2. Traitement d'images

2.1 Les images en niveau de gris

2.1.1 Qu'est-ce qu'une image ?

Une image est une représentation visuelle, voire mentale, de quelque chose (objet, être vivant et/ou concept), il s'agit de **la projection de rayons lumineux** provenant du monde extérieur sur un plan. Les rayons lumineux qui viennent frapper le plan de l'image sont caractérisés par leur intensité uniquement. Cela signifie que sur notre plan, que l'on munit d'un repère en bons cartésiens que nous sommes, on associe à chaque paire de coordonnées (x,y) une valeur I(x,y) que l'on appellera un niveau de gris : plus le point est lumineux, plus la valeur du niveau de gris est élevée. On peut donc considérer qu'une image est une fonction définie comme ceci :

$$\begin{array}{rccc} I: & \mathbb{R}^2 & \to & \mathbb{R} \\ & (x,y) & \mapsto & I(x,y) \end{array}$$

L'image visuelle artificielle peut être :

- **enregistrée** à partir du réel : photographie, vidéo, radiographie, IRM, …

- **fabriquée** à partir d'une construction ou d'une restitution du réel : dessin, peinture, image de synthèse, …

2.1.2 Les images numériques

Un ordinateur ne manipule que des valeurs numériques (représentées sous forme binaire) ; une image numérique devra donc être codée par des nombres.

On distingue deux types d'image numérique :

- **les images matricielles** : formées de points (des pixels).
- **les images vectorielles** : formée d'éléments géométriques (segments de droite, arcs de cercles, …).

2.1.3 Les images numériques matricielles

On appelle **image matricielle** un ensemble fini de points colorés appelés pixels (contraction de pictureelements) organisés en matrice (tableau à 2 dimensions).

Cette matrice contraint l'image à des dimensions fixes : une largeur (width) et une hauteur (height), le tout exprimé en pixels.

Une troisième dimension, appelée profondeur, permet de coder la couleur de chaque pixel. La profondeur d'une image peut être de différentes tailles :

- 1 bit : 0 ou 1 pour les couleurs « noir » ou « blanc »,
- 1 octet : pour coder 256 niveaux de gris, ou bien une palette de 256 couleurs,
- 3 octets : pour coder les proportions de rouge, de vert et de bleu (256 valeurs possible chacun),
- 4 octets : pour coder en plus une information de transparence,

Afin de bien montrer le lien entre une image et une matrice, prenons le tout petit bout d'image suivant :

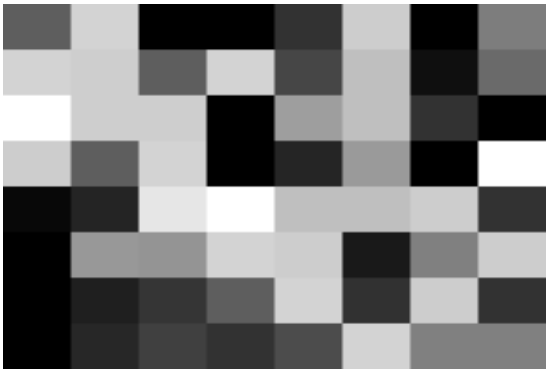


Figure 12 : Image agrandi 20 fois

Et voici maintenant la matrice correspondante:

$$\begin{pmatrix} 94 & 211 & 0 & 0 & 50 & 205 & 0 & 125 \\ 211 & 206 & 94 & 211 & 70 & 191 & 15 & 106 \\ 255 & 206 & 206 & 0 & 158 & 191 & 50 & 0 \\ 205 & 94 & 211 & 0 & 37 & 154 & 0 & 255 \\ 8 & 36 & 230 & 255 & 191 & 191 & 205 & 50 \\ 0 & 152 & 148 & 211 & 205 & 25 & 128 & 205 \\ 0 & 31 & 53 & 94 & 211 & 49 & 205 & 50 \\ 0 & 39 & 64 & 50 & 76 & 211 & 128 & 128 \end{pmatrix}$$

Les pixels blancs correspondent au niveau de gris 255 alors que les pixels noirs correspondent au niveau 0. On dit que l'intensité lumineuse est *échantillonnée* sur 256 valeurs.Maintenant que nous avons établi que les images numériques sont des matrices de pixels, on va se pencher sur la structure IplImage dans OpenCV.

2.1.4 La structure IplImage

```
typedefstruct _IplImage {
        // ...
char *imageData;
        // ...
        } IplImage;
```

Le type **char** est utilisé ici parce qu'il représente exactement un octet en mémoire. Il faut donc lire que les données de l'image sont contenues de la façon la plus naturelle du monde dans un tableau d'octets, même si les pixels ne tiennent pas forcément sur un octet). Ainsi, que ce soit en mathématiques ou dans la RAM de la machine : une image n'est ni plus

ni moins qu'un tableau. Mais cela ne nous dit pas encore dans quel ordre est stocké ce tableau en mémoire. Pour cela, on doit observer un peu plus en détail la structure IplImage :

```
typedefstruct _IplImage {
// ...
intwidth; // nb de colonnes de la matrice
intheight; // nb de lignes de la matrice
// ...
intimageSize; // taille totale des données en octets
char *imageData; // pointeur sur les données
intwidthStep; // largeur d'une ligne en octets
 // ...
} IplImage;
```

On peut déduire de ce morceau de déclaration que les images, dans OpenCV, sont stockées en mémoire ligne par ligne. En anglais, on parle de row-major order. Le schéma suivant montre une image en niveau de gris avec quelques pixels remarquables, ainsi que la façon dont elle sera stockée en mémoire par OpenCV:

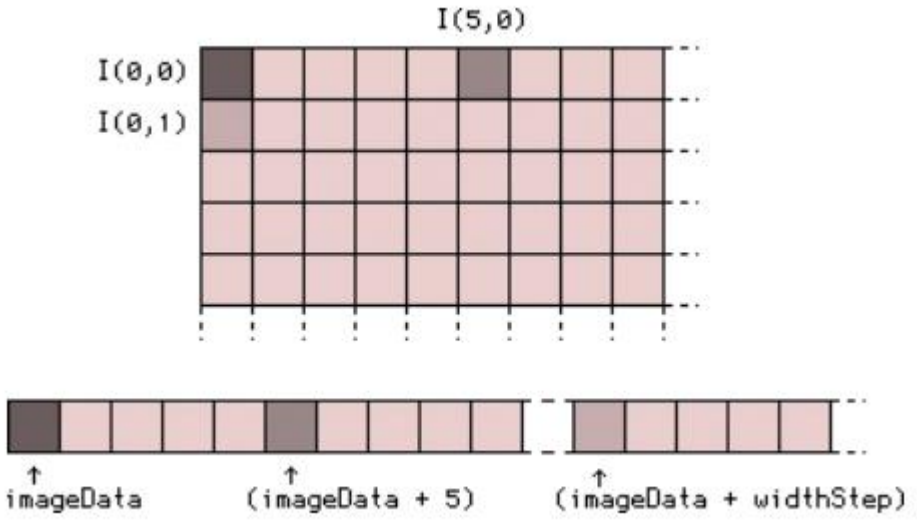


Tableau 3 : Stockage d' une image en niveau de gris en mémoire

Alors si img est un IplImage* contenant une image en niveaux de gris dont les pixels sont des char, alors :

- img->imageData est l'adresse du pixel de coordonnées (0,0) , le premier pixel de l'image (img->imageData + 7) est l'adresse du pixel de coordonnées (7,0), le huitième pixel de la première ligne ;
- (img->imageData + img->widthStep) est l'adresse du pixel de coordonnées (0,1) , le premier pixel de la seconde ligne ;
- (img->imageData + (3 * img->widthStep) + 12) est l'adresse du pixel de coordonnées(12,3) , le treizième pixel de la quatrième ligne ;
- (img->imageData + img->imageSize) est l'adresse de l'octet se situant juste après le dernier pixel de l'image.

Dans le cas des images en niveaux de gris dont les pixels sont des char, il est très probable en effet que les champs width et widthStep soient égaux. Cela dit, il y a deux raisons majeures pour lesquelles on ne peut pas supposer que ce soit le cas. La première raison, c'est qu'OpenCV, en interne, peut décider que les lignes de l'image ne seront pas forcément contiguës en mémoire, mais qu'elles seront alignées sur un nombre d'octets précis, de façon à améliorer les performances. Quand c'est le cas, cet alignement de quelques octets est ajouté au champ widthStep alors que width sera toujours exactement le nombre de colonnes de l'image. La seconde raison, plus simple, est que les niveaux de gris ne sont pas toujours des Char.

2.1.5 La profondeur des images

Quand on parle d'« images 8, 16 ou 32 bits ».On parle de nombre de bits qui désigne la taille d'un pixel en mémoire. On appelle cela **la profondeur (depth) de l'image**.

En toute rigueur, la profondeur d'une image décrit le nombre de valeurs sur lequel l'intensité lumineuse est échantillonnée. Par exemple, sur une image en niveaux de gris 8 bits , l'intensité lumineuse sera échantillonnée sur 256 valeurs, avec 0 pour le noir et 255 pour le blanc, alors que sur une image 16 bits, l'intensité lumineuse pourra prendre 65536 valeurs différentes (0 pour le noir et 65535 pour le blanc), ce qui permet d'être 256 fois plus précis qu'en 8 bits, mais prend deux fois plus de place en mémoire…

Cette profondeur, dans OpenCV, est décrite par le champ depth de la structure IplImage. Ce champ contient une constante définie par OpenCVsous la forme suivante : IPL_DEPTH_[Nombre de bits][Type].

Le tableau suivant décrit les correspondances entre les types des pixels, leurs tailles, les ensembles de valeurs possibles et les constantes de profondeur dans OpenCV:

Type des pixels	Taille	Valeurs	Constante OpenCV
`unsigned char` (ou `uchar`)	8 bits	0, 1, ..., 255	`IPL_DEPTH_8U`
`signed char` (ou `schar`)	8 bits	-127, -126, ..., 128	`IPL_DEPTH_8S`
`unsigned short` (ou `ushort`)	16 bits	0, 1, ..., 65535	`IPL_DEPTH_16U`
`signed short` (ou `short`)	16 bits	-32767, ..., 32768	`IPL_DEPTH_16S`
`float`	32 bits	[0,1]	`IPL_DEPTH_32F`
`double`	64 bits	[0,1]	`IPL_DEPTH_64F`

Tableau 4 : les types des pixels, leurs tailles

2.2 Les images colorées

2.2.1 La représentation théorique des images

La luminance et la chrominance

Nous avons découvert dans la partie précédente la nature des images dans le cas le plus simple : celui d'images en niveaux de gris. Or nous savons très bien, le monde que nous voyons n'est pas gris mais coloré. Pour corriger cela, nous allons maintenant affiner notre représentation de la lumière de façon à y faire entrer la notion de couleur.

Les niveaux de gris dans une image représentent une intensité lumineuse en chaque pixel. On dit qu'ils décrivent **la luminance**, c'est-à-dire l'information relative à l'intensité de la lumière. Cela revient à considérer la lumière comme porteuse d'une seule information, descriptible par un seul nombre : le niveau de gris. Si l'on raisonne en termes d'ondes lumineuses, on peut considérer, grosso-modo, que cette information se traduit par l'amplitude de l'onde.

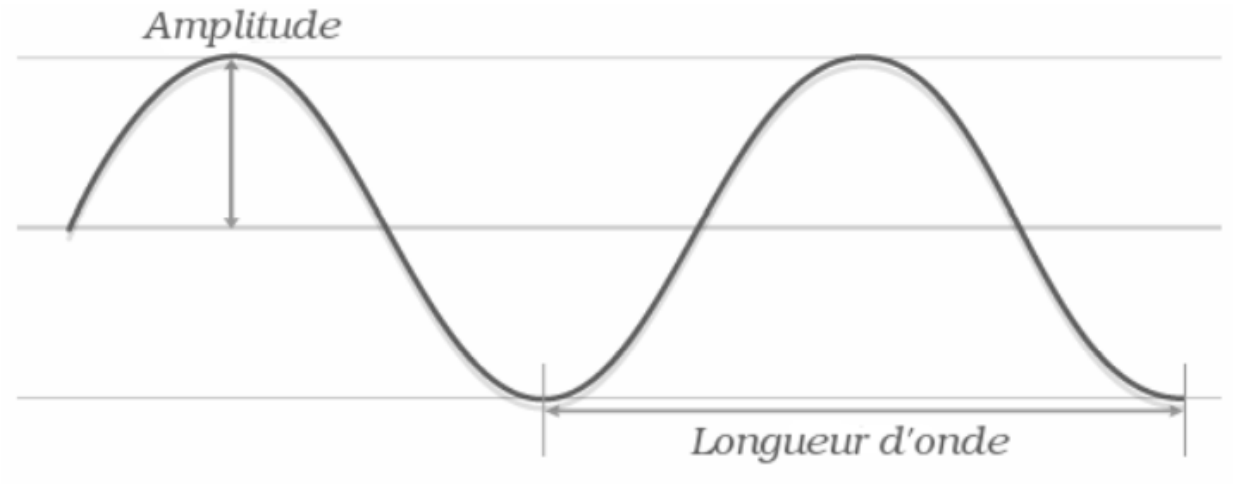


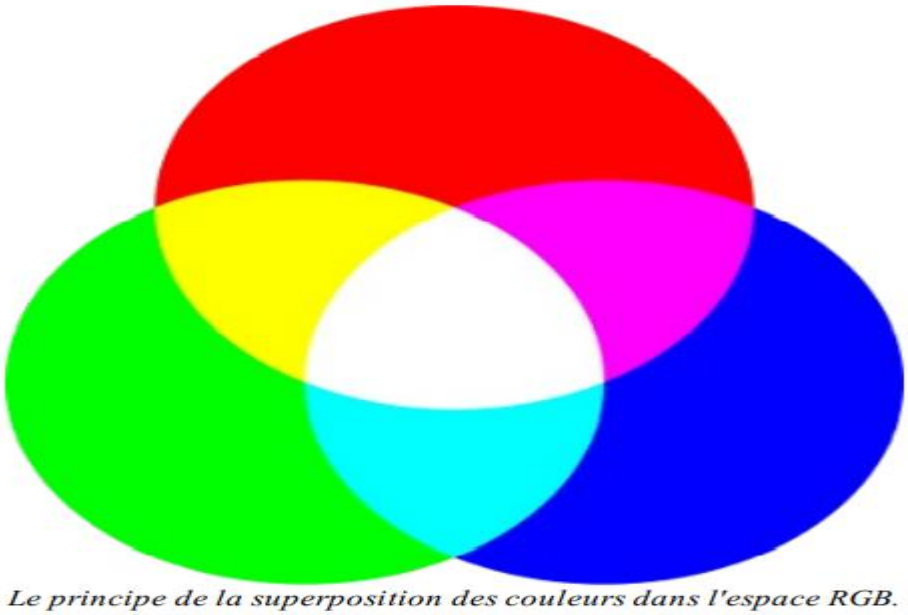
Figure 13 :Représentation d'une onde lumineuse

Seulement, dans le monde réel, la lumière ne se réduit pas à son intensité, tout comme une onde ne se décrit pas seulement par son amplitude, mais aussi par sa fréquence (ou sa longueur d'onde, mais ce sont deux mesures de la même information). On sait tres bien que la fréquence d'une onde lumineuse influence la couleur que nous percevons lorsque le rayon vient frapper notre rétine : du rouge au violet en passant par tous les tons de l'arc-en-ciel. Cette seconde information portée par la lumière est appelé **la chrominance.**

2.2.2 Les espaces de couleurs

Nous considérons maintenant la lumière comme porteuse de deux informations, nous n'allons pas travailler avec des pixels composés de deux nombres mais trois. Ceci est dû au fait que les rayons lumineux sont un petit peu plus complexes en réalité que de simples ondes, et qu'une seule dimension ne suffit pas à décrire efficacement l'information de chrominance, en particulier.

L'espace de couleurs RGB est perçue comme un savant dosage entre trois composantes de base : le rouge (red, R), le vert (green, G) et le bleu (blue, B).



Le principe de la superposition des couleurs dans l'espace RGB.

Figure 14 :Le principe de la superposition des couleurs dans l'espace RGB

Mathématiquement parlant, nous allons représenter une couleur dans l'espace RGB par un vecteur comprenant les niveaux de rouge, de vert et de bleu correspondants. Étant donné que nous travaillons pour l'instant sur des nombres tenant sur un octet, ces valeurs, de manière analogue aux niveaux de gris, seront comprises entre 0 et 255. Par exemple, on peut associer les couleurs suivantes à leurs représentations dans l'espace RGB :

Couleur	Représentation RGB
Rouge	(255,0,0)
Vert	(0,255,0)
Bleu	(0,0,255)
Noir	(0,0,0)
Blanc	(255,255,255)
Jaune	(255,255,0)
Gris 50%	(127,127,127)

Tableau 5 : association des couleurs à leurs représentations dans l'espace RGB

Pour passer d'un niveau de gris à sa valeur dans l'espace RGB, la conversion est évidente. Si ***f*gris = y**, alors son équivalent sera ***f* RGB =(y,y,y)** : il suffit de répliquer le niveau de gris sur les canaux rouge, vert et bleu pour obtenir le même ton.

Pour la transition inverse, en revanche, c'est un petit peu plus compliqué. Le niveau de gris est simplement la moyenne des niveaux de rouge, de vert et de bleu d'une couleur donnée. Bien que cela constitue une approximation acceptable de la luminance, cela revient aussi à négliger le fait que les gammes de fréquence correspondant aux tons rouge, vert et bleu de la lumière n'occupent pas la même place sur le spectre visible, donc que les trois composantes ne contribuent pas toutes de la même manière à la luminosité. En fait, la composante verte contribue beaucoup plus à la luminance que les deux autres.Ainsi, pour en tenir compte lors de l'extraction de la luminance, on affecte des poids aux composantes RGB. La formule classique de conversion est la suivante :

$$f\text{gris} = 0.299 * R + 0.587 * G + 0.114 * b$$

2.2.3 La representation des données

Les images en couleur diffèrent de celles en niveaux de gris par le fait qu'elles sont composées de plusieurs canaux. Une image en RGB sera composée de trois canaux: un canal rouge, un canal vert et un canal bleu. Cette information est accessible sur la structure IplImage grâce à son champ nChannels.

En réalité, cette histoire de canaux ne reflète pas vraiment ce qui se passe en mémoire : L'image en question ne sera pas composée de trois tableaux de dimensions identiques, mais d'un seul dont les pixels prennent trois fois plus de place. On dit que les canaux sont entrelacés. Une autre chose importante à savoir est que dans OpenCV, l'espace couleur par défaut n'est pas RGB, mais BGR. Le second se démarque du premier simplement par le fait que les canaux rouge et bleu sont inversés. Le schéma suivant résume la situation :

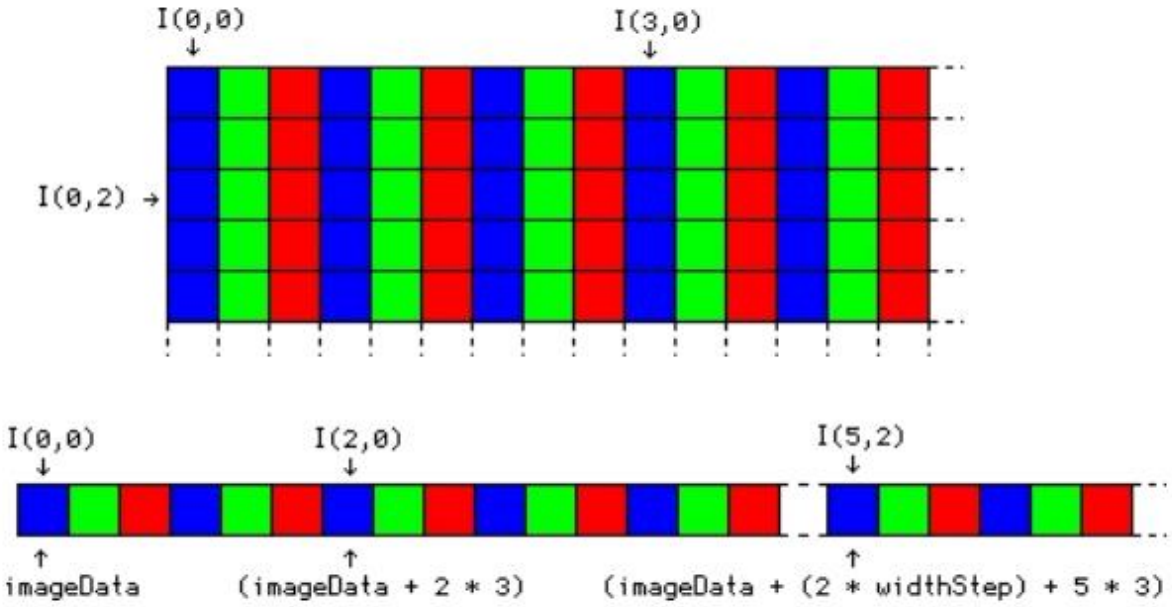


Tableau 6 :Stockage d'une image colorée en mémoire

Ce qui diffère ce tableau avec celui des images en niveaux de gris finalement est lorsque l'on veut accéder à l'adresse d'un pixel donné, c'est qu'il faut multiplier sa coordonnée par 3.

- (img->imageData + (2 * img->widthStep) + 12 * 3) est l'adresse de la composante bleue du pixel de coordonnées (12,2) ,
- (img->imageData + (2 * img->widthStep) + 12 * 3 + 1) est l'adresse de la composante verte de ce même pixel,
- (img->imageData + (2 * img->widthStep) + 12 * 3 + 2) est l'adresse de sa composante rouge.

Chapitre 3: Conception et réalisation de l’application de détection de la fatigue chez le conducteur

1. La reconnaissance faciale

La reconnaissance faciale est une catégorie de logiciels biométriques qui cartographie mathématiquement les traits du visage d'un individu et stocke les données sous forme d'empreinte faciale. Le logiciel utilise des algorithmes d'apprentissage en profondeur pour comparer une capture en direct ou une image numérique à l'empreinte faciale stockée afin de vérifier l'identité d'un individu.

1.1 Le fonctionnement de l’application faciale

Le logiciel identifie 80 points nodaux sur un visage humain. Dans ce contexte, les points nodaux sont des points finaux utilisés pour mesurer les variables du visage d’une personne, telles que la longueur ou la largeur du nez, la profondeur des orbites et la forme des pommettes. Le système fonctionne en capturant des données pour les points nodaux sur une image numérique du visage d'un individu et en stockant les données résultantes sous forme d'empreinte faciale. L'empreinte faciale est ensuite utilisée comme base de comparaison avec les données capturées à partir de visages dans une image ou une vidéo.

2 Conception

2.1 Diagramme de cas d’utilisation

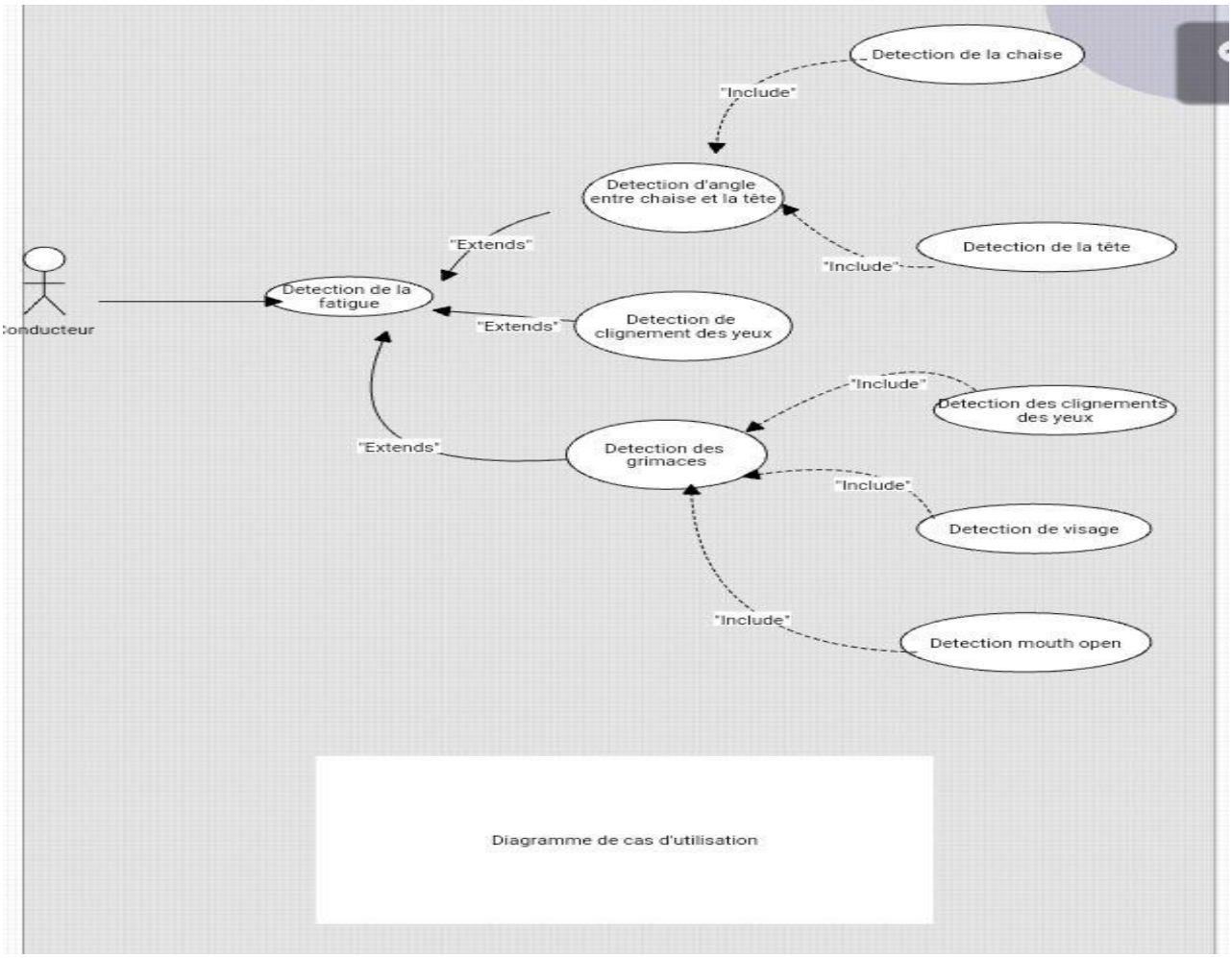


Figure 15: Diagramme de cas d'utilisation

2.2 Diagramme d’activité

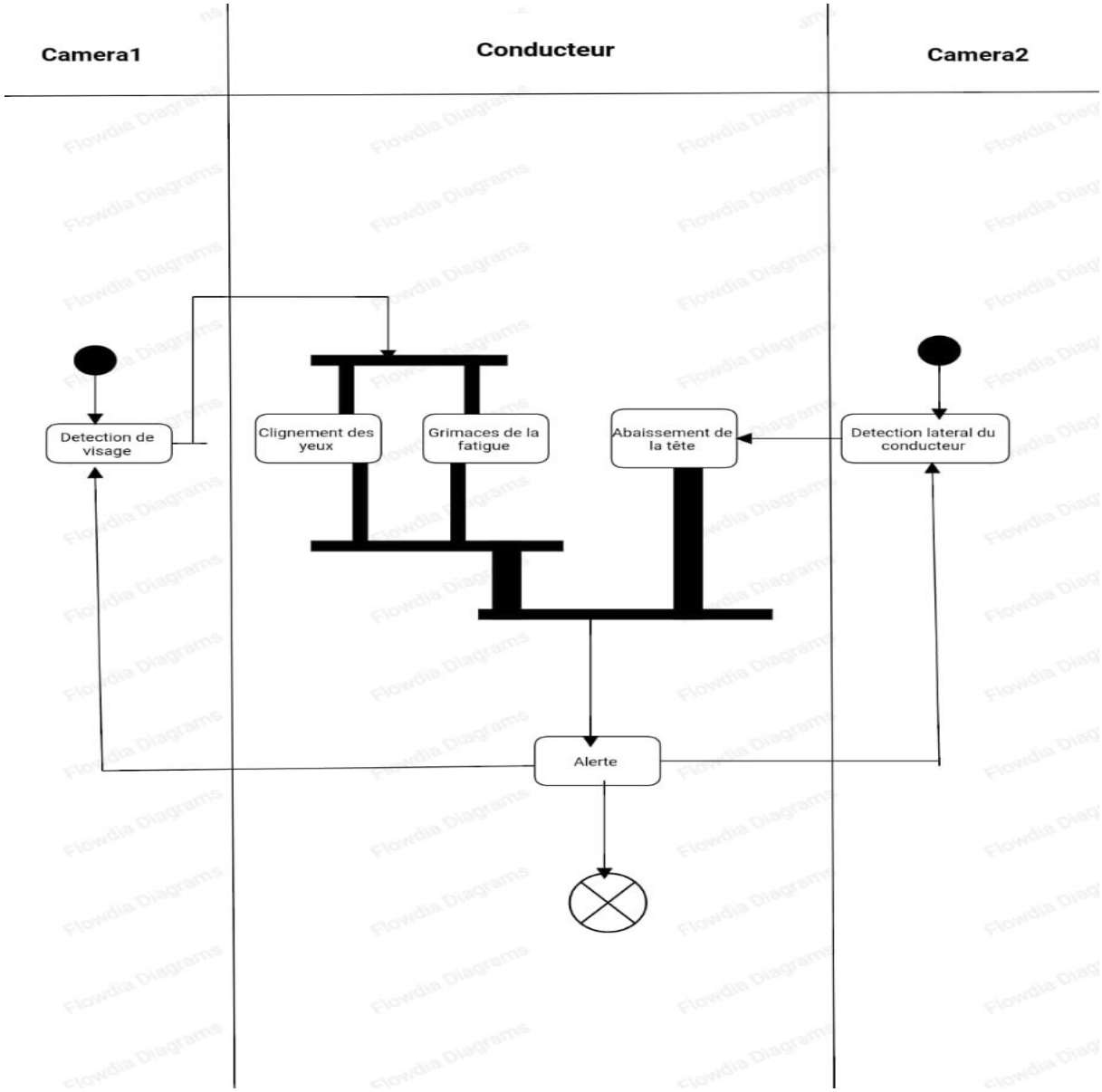


Figure 16: Diagramme d’activité

3 La pratique

Après avoir installé le matériel nécessaire on s’est servi de la reconnaissance faciale pour coder en Python trois parties de code, la première partie détecte le clignement des yeux du conducteur ,la deuxième détecte l’angle de sa tête par rapport à la chaise de la voiture et troisième qui soit la dernière détecte les grimaces de son visage (les grimaces de la somnolence).

3.1 Clignement des yeux

Conduire sans prendre des pauses lors des longs trajets peut entrainer la fatigue chez le conducteur et puis la somnolence. Le clignement des yeux est l'un des signes qui peuvent apparaitre chez le conducteur. Dans cette partie on va essayer de répondre à la question suivante Quand est ce que l'œil clignote ? Pour ensuite détecter quand les yeux de la personne en question devant la caméra clignotent et dans ce cas on va déclencher une alerte pour réveiller la personne et lui sauver la vie d'un accident éventuel qui pourrait être mortel.

3.1.1 Détection des yeux

On importe tout d'abord les bibliothèques Opencv, numpy ainsi que la bibliothèque dlib que nous utiliserons pour détecter les points de repère du visage. Nous créons ensuite un objet cap pour charger les images vidéo de la webcam.

```
import cv2
import numpy as np
import dlib
cap = cv2.VideoCapture(0)
```

Nous chargeons ensuite le détecteur de visage et le prédicteur de points de repère pour détecter les points de repère.

```
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")
```

Nous créons une fonction dont nous aurons besoin plus tard pour détecter le point médian.

Sur cette fonction, nous mettons simplement les coordonnées de deux points et renverrons le point moyen (les points au milieu entre les deux points).

Et après cela, nous pouvons exécuter la boucle while, prendre les images de la webcam et détecter le visage.

```
def midpoint(p1 ,p2):
return int((p1.x + p2.x)/2), int((p1.y + p2.y)/2)
while True:
_, frame = cap.read()
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
faces = detector(gray)
for face in faces:
#x, y = face.left(), face.top()
#x1, y1 = face.right(), face.bottom()
#cv2.rectangle(frame,(x,y),(x1,y1),(0,255,0),2)
```

En utilisant l'approche de détection des repères de visage, nous pouvons trouver 68 repères spécifiques du visage. Un index spécifique est attribué à chaque point.

Nous avons besoin de deux détectent séparément les deux yeux:

- Points de l'œil gauche: (36, 37, 38, 39, 40, 41)
- Points de l'œil droit: (42, 43, 44, 45, 46, 47)

```
landmarks = predictor(gray, face)
left_point = (landmarks.part(36).x, landmarks.part(36).y)
right_point = (landmarks.part(39).x, landmarks.part(39).y)
```

Une fois que nous connaissons la position exacte des yeux, nous pouvons commencer à travailler avec eux.

Créons deux lignes, l'une traversant l'œil horizontalement et l'autre verticalement. Nous en aurons besoin pour le prochain tutoriel lorsque nous apprendrons à détecter le clignotement des yeux.

Une fois que nous connaissons la position exacte des yeux, nous pouvons commencer à travailler avec eux.

Créons deux lignes, l'une traversant l'œil horizontalement et l'autre verticalement. Nous en aurons besoin pour lorsque nous allons détecter le clignotement des yeux.

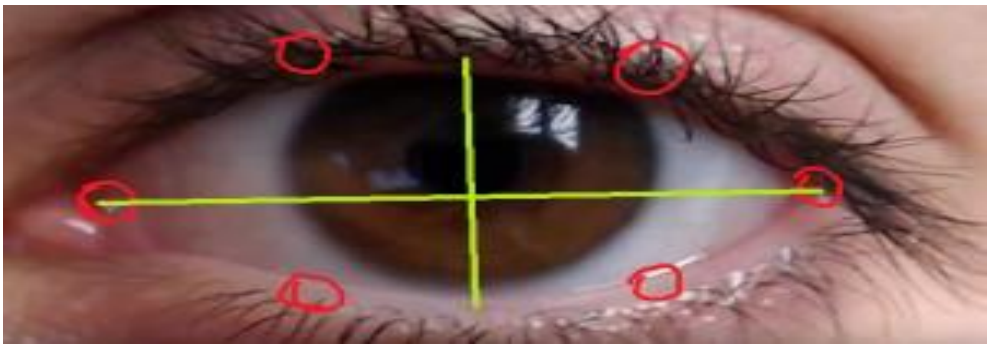


Figure 17: Détection des repères de l'œil

```
center_top = midpoint(landmarks.part(37), landmarks.part(38))
center_bottom = midpoint(landmarks.part(41), landmarks.part(40))
hor_line = cv2.line(frame, left_point, right_point, (0, 255, 0), 2)
ver_line = cv2.line(frame, center_top, center_bottom, (0, 255, 0), 2)
```

Et enfin, nous montrons tout à l'écran.

```
cv2.imshow("Frame", frame)
key = cv2.waitKey(1)
if key == 27:
break
cap.release()
cv2.destroyAllWindows()
```

3.1.2 Détection du clignement des yeux

Quelle explication possible pourrions-nous donner pour décrire le clignement des yeux?

Probablement plus d'une réponse sera la bonne.Un œil clignote lorsque:

- La paupière est fermée
- On ne voit plus le globe oculaire
- Les cils inférieurs et supérieurs se connectent ensemble.

Nous devons également tenir compte du fait que toutes ces actions doivent se produire pendant un court laps de temps (environ un clin d'œil prend 0,3 à 0,4 seconde) sinon il mesure que l'œil est juste fermé. Lorsque nous avons détecté l'œil, nous avons également détecté deux lignes: une ligne horizontale et une ligne verticale traversant l'œil.

Examinons-les de près car ils vont être au cœur de l'approche de détection clignotante.Voici à quoi ressemblent les lignes lorsque l'œil est ouvert et lorsqu'il est fermé :

Figure 18: Détection de l'œil fermé et ouvert

On voit clairement que la taille de la ligne horizontale est presque identique à l'œil fermé et à l'œil ouvert tandis que la ligne verticale est beaucoup plus longue à l'œil ouvert en comparaison avec l'œil fermé.

A l'œil fermé, la ligne verticale disparaît presque.

Nous prendrons alors la ligne horizontale comme point de référence, et à partir de là nous calculons le rapport par rapport à la ligne verticale.

Si le rapport descend en dessous d'un certain nombre, nous considérerons que l'œil est fermé, sinon ouvert.

Sur python, nous créons une fonction pour détecter le rapport de clignotement où nous insérons les points oculaires et les coordonnées du repère facial et nous obtiendrons le rapport entre ces deux lignes.

```
def get_blinking_ratio(eye_points, facial_landmarks):
left_point = (facial_landmarks.part(eye_points[0]).x, facial_landmarks.part(eye_points[0]).y)
right_point = (facial_landmarks.part(eye_points[3]).x, facial_landmarks.part(eye_points[3]).y)
center_top = midpoint(facial_landmarks.part(eye_points[1]), facial_landmarks.part(eye_points[2]))
center_bottom = midpoint(facial_landmarks.part(eye_points[5]), facial_landmarks.part(eye_points[4]))
hor_line = cv2.line(frame, left_point, right_point, (0, 255, 0), 2)
ver_line = cv2.line(frame, center_top, center_bottom, (0, 255, 0), 2)
hor_line_lenght = hypot((left_point[0] - right_point[0]), (left_point[1] - right_point[1]))
ver_line_lenght = hypot((center_top[0] - center_bottom[0]), (center_top[1] - center_bottom[1]))
ratio = hor_line_lenght / ver_line_lenght
return ratio
```

Nous utiliserons ensuite le nombre de rapport plus tard pour détecter et nous pourrons enfin définir quand l'œil clignote ou non.Dans ce cas, j'ai trouvé le rapport 5,7 comme le seuil le plus fiable, du moins pour mon œil.

```
landmarks = predictor(gray, face)
left_eye_ratio = get_blinking_ratio([36, 37, 38, 39, 40, 41], landmarks)
right_eye_ratio = get_blinking_ratio([42, 43, 44, 45, 46, 47], landmarks)
blinking_ratio = (left_eye_ratio + right_eye_ratio) / 2
if blinking_ratio > 5.7:
cv2.putText(frame, "BLINKING", (50, 150), font, 7, (255, 0, 0))
```

Le code donne le résultat suivant :

Figure 19:Résultat du code 1

3.2 Détection de l'angle entre la tête et la chaise de la voiture

3.2.1 Détection des objets

En vision par ordinateur on désigne par détection d'objet (ou classification d'objet) une méthode permettant de détecter la présence d'une instance (reconnaissance d'objet) ou d'une classe d'objets dans une image numérique. Une attention particulière est portée à la détection de visage et la détection de personne. Ces méthodes font souvent appel à l'apprentissage supervisé et ont des applications dans de multiples domaines, tels la recherche d'image par le contenu ou la vidéo surveillance.

Dans notre code on a détecter la tête du conducteur, la chaise et l'angle entre eux , la tête a en effet tendance à tomber en avant sous un angle de 15° à 20° quand la somnolence guette le conducteur. C'est à ce moment qu'il faut déclencher l'alerte.

```
VideoCapture cap(0);

CascadeClassifier profile("c:/p/opencv/data/haarcascades/haarcascade_profileface.xml");

while(1) {

    Mat frame, gray;

    cap.read(frame);

    pyrDown(frame,frame);

    cv::cvtColor(frame, gray, cv::COLOR_BGR2GRAY);

    std::vector<cv::Rect> faces_right,faces_left;

    std::vector<int> lvl_right,lvl_left;

    std::vector<double> weights_right,weights_left;

    // right face side

    profile.detectMultiScale(gray, faces_right, lvl_right, weights_right, 1.2, 1, 0, cv::Size(30, 30), Size(), true);

    // flip, and apply again for left one

    flip(gray,gray,1);

    profile.detectMultiScale(gray, faces_left, lvl_left, weights_left, 1.2, 1, 0, cv::Size(30, 30), Size(), true);

   float angle = 0; // formula from paper: a=-90*l+90*r ;)

    if (weights_right.size()>0 && weights_right[0]>0)

        // 4: heuristic scale factor, though i'm pretty sure, this is not linear ;)

        angle += 90 * weights_right[0] / 4;

    if (weights_left.size() && weights_left[0]>0)

        angle += -90 * weights_left[0] / 4;

    cout << angle << endl;

    imshow("F",frame);

    if (waitKey(10) > -1) break;

}
```

Le résultat

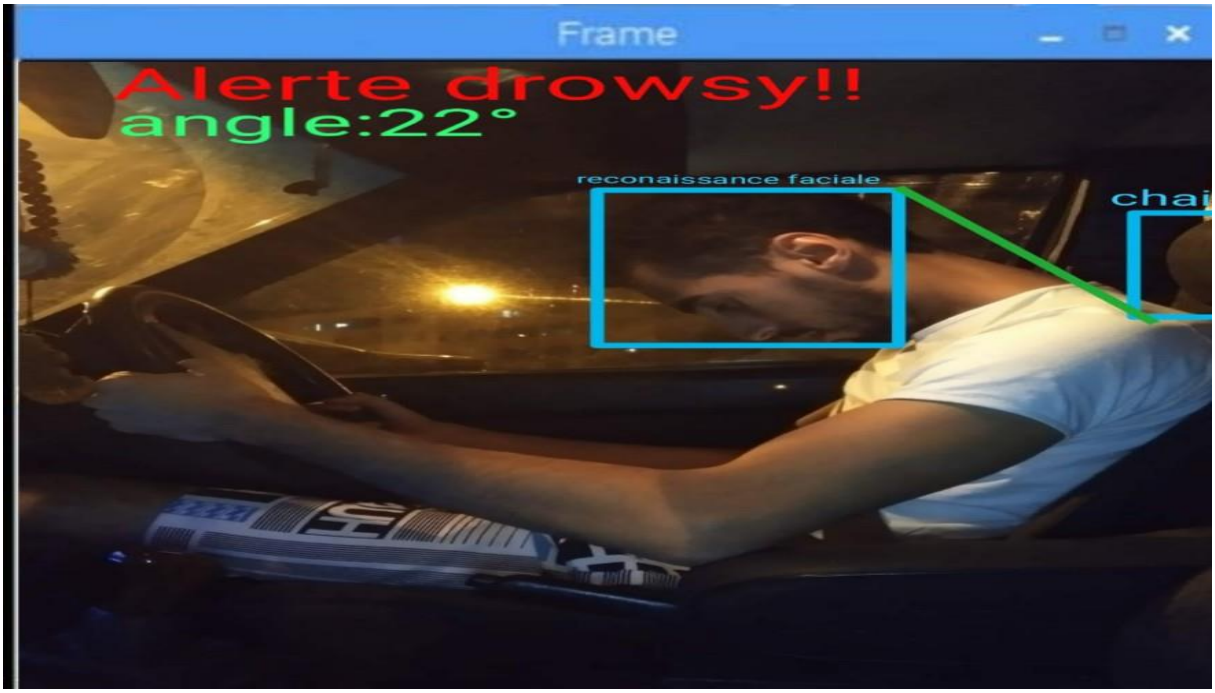


Figure 20 :Résultat du code 2

3.3 Détection des grimaces

On va premièrement charger une image du visage ,puis on code la détection du visage,pour finir par la détection des caractéristiques du visage (Le nez, la bouche et les yeux)

Etape 1 :Chargement et présentation d'une image

```
import cv2
# Lire l'image
img = cv2.imread("face.jpg")
# Afficher l'image
cv2.imshow(winname="Face", mat=img)
# Wait for a key press to exit
cv2.waitKey(delay=0)
# Fermer tous les windows
cv2.destroyAllWindows()
```

Étape 2: reconnaissance faciale

```
import cv2
import dlib
# Télecharger le detecteur
detector = dlib.get_frontal_face_detector()
# Lire l’’image
img = cv2.imread("face.jpg")
# Convertir l'image en niveau de gris
gray = cv2.cvtColor(src=img, code=cv2.COLOR_BGR2GRAY)
# Use detector to find landmarks
faces = detector(gray)
for face in faces:
    x1 = face.left() # left point
    y1 = face.top() # top point
    x2 = face.right() # right point
    y2 = face.bottom() # bottom point
  # Tracer un  rectangle
    cv2.rectangle(img=img, pt1=(x1, y1), pt2=(x2, y2), color=(0, 255, 0), thickness=4)
# Afficher l’’image
cv2.imshow(winname="Face", mat=img)
# Wait for a key press to exit
cv2.waitKey(delay=0)
# Close all windows
cv2.destroyAllWindows()
```

Étape 3: Identification des caractéristiques du visage

```
import cv2
import dlib
# Télecharger le detecteur
```

```
detector = dlib.get_frontal_face_detector()
# Télecharger le  predicteur
predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")
# Lire l'image
cap = cv2.VideoCapture(0)
while True:
    _, frame = cap.read()
    # Convertir l'image en niveau de gris
    gray = cv2.cvtColor(src=frame, code=cv2.COLOR_BGR2GRAY)
    # Utiliser le détecteur pour trouver des points de repère
    faces = detector(gray)
    for face in faces:
        x1 = face.left()  # left point
        y1 = face.top()  # top point
        x2 = face.right()  # right point
        y2 = face.bottom()  # bottom point
        # Créer un objet de repère
        landmarks = predictor(image=gray, box=face)
        # Boucler à travers tous les points
        for n in range(0, 68):
            x = landmarks.part(n).x
            y = landmarks.part(n).y


    # Tracer le cercle
            cv2.circle(img=frame, center=(x, y), radius=3, color=(0, 255, 0), thickness=-1)
    # show the image
    cv2.imshow(winname="Face", mat=frame)
    # Exit quand escape est pressé
    if cv2.waitKey(delay=1) == 27:
```

```
        break

# Lorsque tout est terminé, libérez les objets de capture vidéo et d'écriture vidéo

cap.release()

# Close all windows

cv2.destroyAllWindows()
```

Le code affiche finalement le résultat suivant :

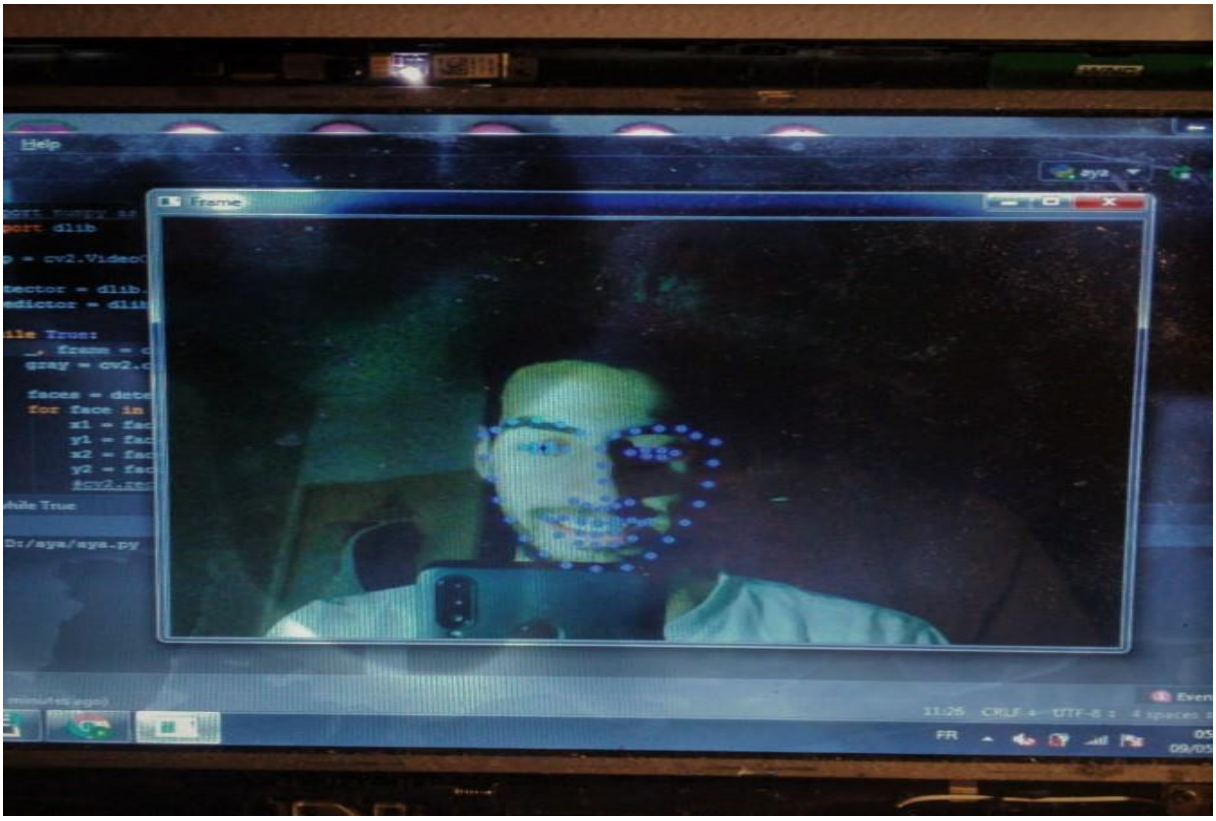


Figure 21 :Résultat du code 3

CONCLUSION GENERALE

L'Intelligence Artificielle évolue et progresse de jour en jour depuis le vingtième siècle dans le domaine de l'informatique mais aussi dans la robotique. Aujourd'hui cette intelligence n'a pas encore atteint son seuil de perfection c'est à dire qu'elle n'est pas capable de raisonner elle-même, de s'autogérer et d'éprouver des sentiments comme le ferait un humain. Cependant elle est tout de même capable de le remplacer dans plusieurs domaines tels que la médecine, le militaire, les jeux vidéo... ainsi que de simplifier sa vie quotidienne. Les nombreux progrès de cette intelligence posent de réels problèmes d'éthique surtout dans le domaine militaire car les robots créés sont certes semi-autonome (gérer par un humain) mais ils sont tout de même composer d'armes de guerres capables de tuer des hommes. Pour autant, nous avons découvert que cette intelligence artificielle comporte aussi de nombreux points positifs ainsi que de nombreux avantages capables d'améliorer la vie humaine pour ainsi la rendre plus facile. Petit à petit l'homme se fait remplacer par ce type d'intelligence, sa venue a supprimé un nombre consistant d'emplois. De nos jours l'Intelligence Artificielle est encore contrôlable mais jusqu'à quand ? L'intelligence Artificielle dépassera-t-elle l'Intelligence Humaine ? Aujourd'hui l'Intelligence Artificielle remplace l'homme dans certains domaines mais pas dans tous heureusement mais qu'elle sera la place de l'homme dans l’avenir, cette intelligence fera-t-elle mieux que lui ?

WEBOGRAPHIE

Apprentissage Supervisé. url : http://www.intellspot.com/unsupervised-vs-supervised-learning/

Apprentissage non supervisé. url :http://www.intellspot.com/unsupervised-vs-supervised-learning/

NumPy.url :https://fr.wikipedia.org/wiki/NumPy.

Python.url :https://fr.wikipedia.org/wiki/Python.

OpenCv.url :https://fr.wikipedia.org/wiki/Opencv.

Dlib.url :https://fr.wikipedia.org/wiki/NumPy.

https://www.pyimage.com/