République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITE M'HAMED BOUGARA-BOUMERDES

Faculté des **S**ciences de l'**I**ngénieur

Département d'Ingénierie des Systèmes Electriques

Mémoire de Master

Présenté par :

M^lle^ Djaoudi Meryem

M^lle^ Bouguerra Amina

En vue de l'obtention du diplôme de **Master** en Électronique

Option : Électronique des Systèmes Embarqués

Thème :

Conception et développement d'un système intelligent pour la détection et la mesure de vitesse des véhicules

Président	**:**	Mr Harrar Khaled (grade MCA)
Examinateurs :		Mme Guerbai Yasmine (grade MCB)
Examinateurs :		Mr Smaani Billel (grade MCB)
Encadreur	**:**	Mr Himeur Yacine (grade maitre de recherche CDTA)
Co-encadreur :		Mlle Mahdi Ismahan (grade MAA)

Promotion Juin 2018

Dédicace

Chaleureusement, je dédie ce modeste travail

À mes chers parents, pour tous leurs sacrifices, leur amour, leur tendresse, leur soutien et leurs prières tout au long de mes études,

À mes sœurs pour leurs encouragements permanents, et leur soutien moral,

À mes frères pour leur appui et leur encouragement,

À mon futur mari ***Samir***

À toute ma famille pour leur soutien tout au long de mon parcours universitaire,

Que ce travail soit l'accomplissement de vos vœux tant allégués, et les fruits de votre soutien infaillible,

Merci d'être toujours là pour moi.

Djaoudi Meryem

Chaleureusement, je dédie ce modeste travail

À mes chers parents, pour tous leurs sacrifices, leur amour, leur tendresse, leur soutien et leurs prières tout au long de mes études,

À mes sœurs pour leurs encouragements permanents, et leur soutien moral,

À mes frères pour leur appui et leur encouragement,

À mon futur mari ***Hakim***

À toute ma famille pour leur soutien tout au long de mon parcours universitaire,

Que ce travail soit l'accomplissement de vos vœux tant allégués, et les fruits de votre soutien infaillible,

Merci d'être toujours là pour moi.

Bouguerra Amina

Remerciements

Tout d'abord, nous remercions le Dieu, notre créateur de nous avoir donné les forces, la volonté et le courage afin d'accomplir ce travail modeste.

Nous adressons le grand remerciement à notre encadreur **Mr Himeur Yacine** *qui a proposé le thème de ce mémoire, et notre Co-encadreur* **Mlle Mahdi ismahan,** *pour ses conseils et ses dirigés du début à la fin de ce travail.*

Nous tenons également à remercier, Messieurs les Membres de jury pour l'honneur qu'ils nous ont fait en acceptant de siéger à notre soutenance, tout particulièrement :

Mr Harrar Khaled *pour nous avoir fait l'honneur de présiderle jury de cette mémoire. Nous souhaitons exprimer notre gratitude à* **Mme Guerbai Yasmine** *et* **Mr Smaani Billel** *pour avoir faire de lecteur notre mémoire, aller l'examiner et ils peuvent évaluer cette mémoire. Nous vous remercions pour l'intérêt que vous avez porté à ce travail et pour vos précieux conseils et remarques.*

Finalement, nous tenons à exprimer notre profonde gratitude à nos familles qui nous ont toujours soutenues et à tout ce qui participe de réaliser ce mémoire. Ainsi que l'ensemble des enseignants qui ont contribué à notre formation.

Liste des abréviations

CDT: Calculateur de densité de trafic

CNN: Convolutional Neural Networks

DOM: Détection d'objets en mouvement

eLU: exponential Linear Unit

GTI: Grand Tourisme Injection

HOG: Histogramme of Oriented Gradients

HTML: HyperText Markup Language

IA: Intelligence Artificielle

LReLU: Leaky Rectified Linear Unit

PMC: Perceptron multicouches

PReLU: Parameteric Rectified Linear Unit

RBF: Radial Basic Function

ReLU: (Rectified Linear Unit)

RNA : Réseaux de Neurones Artificielles

R.V.B : Rouge Vert Bleu

SGML: Standard Generalized Markup Language

VO: Vision par Ordinateur

XML: eXtended Markup Language

Table des figures

Liste des tableaux

Table des matières

مـلخـص

أصبحت أنظمة كشف السيارات وتحديد سرعتها شائعة بشكل متزايد في مجال رؤية الكمبيوتر. مراحل التحويل الأساسية المطبقة في هذه الأنظمة تعتمد أساسا على إستخلاص خصائص الصور إنطلاقا من تتابعات الفيديو ، ثم تصنيف هذه الصور. من بين مناهج الذكاء الاصطناعي المستعملة في هذا المجال نجد الشبكات العصبية التلافيفية المتخصصة في التعرف على الأشكال .في عملنا المنجز قمنا بإستعمال هذا النوع من الشبكات بهدف تصنيف الصور، بناءً على نموذج متعدد الطبقات.

كلمات مفتاحية: الكشف، التصنيف، الشبكات العصبية، التعلم العميق

Résumé

Les systèmes de détection et d'identification de véhicules et de leurs vitesses sont devenus de plus en plus commun à la communauté de vision par ordinateur. Les étapes de transformation principales impliquées dans ces systèmes sont basées essentiellement sur l'extraction des caractéristiques des images à partir des séquences vidéos, et de procéder par la suite à la classification de ces images. Parmi les approches de l'intelligence artificielle utilisées dans ce domaine, les réseaux de neurones convolutionnels qui sont spécialisés dans la reconnaissance de forme. Dans notre travail, nous avions utilisé ce type de réseaux pour la classification des images, basé sur un modèle multicouches.

Mots Clés : Détection, Classification, Réseaux de Neurones, Apprentissage Profondeur

Abstract

The vehicle detection and identification systems have become increasingly common to the computer vision community. The key processing steps implied in these systems are based primarily on the images characteristics extraction from the video sequences, and to proceed thereafter to the classification of these images. Among the artificial intelligence approaches used in this field, the neuronal convolutional networks which are specialized in the pattern recognition. In our work, we had used this kind of networks for the images classification, based on multi-layer model.

Key Words: Detection, Classification, Neural Networks, Deep Learning

Introduction Générale

La détection de véhicules permet des applications diverses dans les systèmes de transport intelligent et de surveillance automatique telle que le suivi, le recensement, le contrôle de véhicules et de leurs vitesses. Ce type de reconnaissance est un défi dans le domaine de la vision par ordinateur. En effet, discriminer des objets parfois très semblables est très complexe. A la fin des années 80 un type de réseau particulier qui s'appelle le réseau de neurones convolutionnel a été développé. Ce dernier est une forme particulière du réseau neurone multicouche dont l'architecture des connexions est inspirée de celle du cortex visuel des mammifères.

Plusieurs records en reconnaissance d'image ont été battus par des réseaux de neurones convolutionnels. La diminution des taux d'erreurs était telle qu'une véritable révolution dont la majorité des équipes de recherche en parole et en vision ont abandonné leurs méthodes préférées et sont passées aux réseaux de neurones convolutionnels et autres réseaux neurones. L'industrie d'Internet a immédiatement saisi l'opportunité et a commencé à investir massivement dans des équipes de recherche et développements en apprentissage profond.

L'objectif principal de notre travail est de reconnaître les classes d'objets de type «véhicule», à l'aide des réseaux de neurones convolutionnels sur une base de données d'images extraites de plusieurs séquences vidéo des caméras de surveillance du trafic routier. Notre approche consiste à développer une application informatique, qui est censée implémentée sur un système intelligent, pour la mesure et le contrôle de vitesse des véhicules en temps réel. Ceci se fait en utilisant les techniques de traitement d'images basées sur les réseaux de neurones convolutionnels (Deep Convolutional Networks) pour la classification des informations.

Notre travail sera divisé en deux parties : la première partie sera consacrée à la détection des véhicules en utilisant les méthodes Deep Learning via les réseaux de neurones. Quant à la deuxième partie à la mesure de vitesse en temps réel tout en sauvegardant les pixels des différentes positions dans chaque séquence vidéo, tout en marquant les changements constatés dans ces positions en utilisant un classificateur de véhicules prédéfini, et par la suite les convertir en paramètres d'objets réels.

Notre mémoire est organisé en trois chapitres comme suit :

Dans le premier chapitre, nous allons présenter quelques travaux de recherche dans le domaine de traitement d'image pour le contrôle du trafic routier, chose qui est devenue primordiale, afin de minimiser les accidents de circulation. Nous allons par la suite présenter notre approche qui traite la même thématique de recherche, et qui repose sur l'utilisation de l'une des méthodes de l'Intelligence Artificielle : les réseaux de neurones artificiels.

Le second chapitre sera consacré à la présentation des différentes approches de l'Intelligence Artificielle, de la vision par ordinateur ainsi que les réseaux de neurones convolutionnels, et leurs intérêts dans le domaine de la classification des images pour la détection des véhicules.

Dans le troisième et dernier chapitre, notre approche de détection des véhicules et de leurs vitesses sera présentée et implémentée via une application informatique. Pour cela, le modèle de réseaux en couches sera adopté en présentant son architecture. Divers outils de programmation utilisés, tels que le langage Python et les librairies OpenCV, seront présentés. Une présentation des résultats obtenus via l'application conçue sera éventuellement faite.

Nous terminerons par une conclusion générale.

Chapitre I : L'État de l'art

I.1 Introduction

Depuis longtemps les chercheurs font une course pour le développement des systèmes de détection des véhicules et leur vitesse, afin d'améliorer les systèmes de surveillance routière pour éliminer les problèmes tels que les accidents de circulation, par l'utilisation des approches de l'Intelligence Artificielle (IA). Dans ce qui suit, nous allons citer quelques travaux dans la même thématique de recherche que la nôtre.

I.2 Présentation des travaux de recherche

I.2.1 Utilisation des réseaux de neurones pour la détection des véhicules par Darcy Bullok et al

L'objectif de ce travail est de développer un nouveau système de détection de véhicules basé sur un simple réseau de neurones de propagation [1]. La technique proposée par les auteurs ne repose pas sur le développement ou l'étalonnage de gabarits rigides, mais plutôt sur la reconnaissance de formes des véhicules parmi plusieurs exemples de véhicules. Ceci offre un avantage significatif par rapport au traitement d'images classique, puisque le réseau de neurones peut s'adapter à différentes perspectives de caméra, conditions d'éclairage, etc.

Une autre technique a été envisagée par les auteurs, c'était l'utilisation de plusieurs sorties binaires séquentielles indiquant si un véhicule est ou non dans un segment particulier de l'image. Cependant, des études sur les représentations de données ont montré que l'apprentissage des réseaux de neurones pour produire ces sorties binaires est difficile en présence de bruit. Par conséquent, une représentation redondante a été utilisée pour améliorer la représentation de sortie binaire séquentielle.

Le modèle mathématique du réseau de neurone utilisé dans ce travail est basé sur une classe particulière de réseaux appelée « **propagation avant** », Le réseau est composé de **n** unités de traitement d'entrée, et **p** unités de traitement cachées et **m** unités de traitement de sortie. Chaque unité de traitement peut recevoir des entrées, transmettre des sorties et effectuer des calculs. Les unités de traitement d'entrée sont les récepteurs de la carte de pixel de la zone de détection. Chaque tuile **Ti** dans l'image d'entrée est représentée comme un vecteur de valeurs dans la plage [0... 255]. Chaque unité de traitement d'entrée émet une valeur de sortie normalisée :

$$\boldsymbol{ti} = (\boldsymbol{Ti}, -\mathbf{128}) \,/\, \mathbf{128.0} \;\; \boldsymbol{Vi} = \mathbf{1}, \ldots, \boldsymbol{n}, \qquad \text{(I. 1)}$$

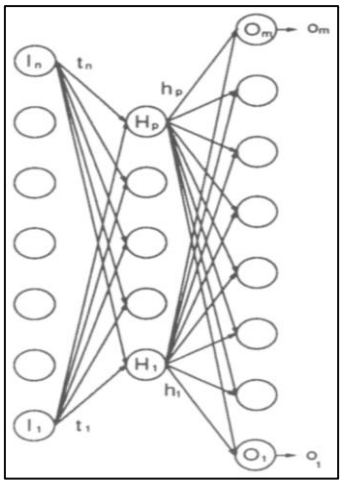

Fig I. 1.Modèle de suivi de réseau de neurones

Chaque sortie des unités de traitement d'entrée est connectée à un récepteur d'entrée sur toutes les unités de traitement cachées. Ces unités de traitement émettent également une valeur de sortie normalisée :

$$\boldsymbol{h_k = f\left(\sum_{i=1} w_{j,k} * t_j + \beta_k\right), \forall_k = 1, \ldots\ldots, p,} \qquad \text{(I. 2)}$$

Pour que le réseau de neurones apprendre à suivre les véhicules d'abord un humain montre plusieurs exemples d'images, l'apprentissage est fait selon deux étapes : la construction de l'ensemble d'apprentissages et l'ajustement des poids et des biais de connexion, afin que le réseau de neurones apprenne à cartographier les images en mosaïque dans des vecteurs de localisation. Un ensemble de données d'apprentissage peut être assemblé et il est composé à la fois du vecteur d'image et d'un vecteur de localisation sortie. Ces données sont utilisées pour apprendre au réseau comment cartographier l'image en vecteur mosaïque au vecteur de localisation. L'ensemble d'apprentissages est construit par la collection de plusieurs dizaines d'instantanés de zone de détection et par l'utilisation d'une fenêtre graphique, qui permet à l'opérateur d'identifier les instantanés dans lesquels se trouvent des véhicules et de spécifier l'emplacement des véhicules dans ces images.

La technique d'entraînement utilisée pour établir les poids de connexion $(Wj, kp\ w, J)$ et les biais $(\beta k, \beta j)$, est une procédure de modification des poids décent en boucle fermée connue sous le nom de Règle généralisée Delta. La méthode d'acquisition de poids fournit une technique pour modifier itérativement les gains et les préjugés de sorte que le réseau neural apprend la correspondance générale entre les vecteurs d'image en mosaïque et les vecteurs de localisation. La procédure de formation en boucle fermée est réalisée par l' utilisation des centaines de cartes de

pixels classées pour apprendre au réseau neural à produire la courbe de forme gaussienne pour varier les conditions d'éclairage, la forme des véhicules et l'emplacement des véhicules.

Le modèle de détection proposé par les auteurs peut être calibré par un opérateur "montrant" le système plusieurs exemples d'images, contenant des véhicules et identifiant leurs emplacements. Ceci élimine le besoin d'un opérateur de terrain d'avoir une compréhension profonde du traitement d'images. Ainsi, les réseaux de neurones peuvent mieux performer, que les techniques de traitement d'images traditionnelles, les conditions d'éclairage extérieur non structuré habituellement rencontrées par les systèmes de détection de véhicules basés sur l'image. Un autre avantage, c'est de réduire l'effort de formation et améliorer la fiabilité du modèle de suivi du réseau de neurones.

I.2.2 Utilisation des réseaux de neurones pour l'identification des véhicules et la densité du trafic par le traitement des vidéos de trafic par Celil Ozkurt et Fatih Camci

L'objectif visé dans ce travail est l'estimation automatique de densité du trafic ainsi qu'est l'identification des véhicules par l'utilisation des réseaux de neurones, à partir des vidéos obtenues de l'une des caméras de circulation utilisées par ISBAK (Société de Gestion de Trafic d'Istanbul). Ce modèle a été appliqué sur 1000 images qui durent 100 secondes (10 images chaque seconde). Il est composé de trois sous modèles qui permettent de faire : la Détection d'Objet en Mouvement (MOD), l'Identification de véhicule (IV) et le Calculateur de Densité de Trafic (CDT) [2].

a. Détection d'objets en mouvement (DOM)

La détection d'objet en mouvement est appliquée distinctement à chaque image et réalisée en deux étapes : dans la première étape, l'arrière-plan est soustrait de l'image en cours, la matrice de différence est appliquée à un seuil, les niveaux de gris supérieurs et inférieurs au seuil sont mis à jour en tant que 1 et 0 respectivement, ce qui conduit les objets en mouvement à être représentés sous la forme de pixels blancs en utilisant les formules. (I. 3) et (I. 4) :

$$\boldsymbol{D_{i,j} = C_{i,j} - B_{i,j}} \qquad \textit{(I. 3)}$$

D : Matrice de différence avec n ligne et m colonne

C : Matrice de trame actuelle avec n ligne et m colonne

B : Matrice de fond avec n ligne et m colonne

$$\boldsymbol{D_{i,j} = \begin{cases} 1 \; si \; D_{i,j} > th \\ 0 \; si \; D_{i,j} \leq th \end{cases}} \qquad \textit{(I. 4)}$$

Dans la deuxième étape, une fois que la matrice binaire montrant la différence entre la trame actuelle et l'arrière-plan est obtenue, elle sera analysée afin de détecter l'objet en mouvement.

Dans le calcul de l'arrière-plan dynamique, l'arrière-plan est mis à jour pour chaque image. Dans l'algorithme utilisé par auteurs, les niveaux de gris de pixel sont stockés en tant que données de séries temporelles et la valeur médiane des données de séries temporelles est mise à jour sur la base du nouveau niveau de gris dans la trame actuelle.

b. Identification des Véhicules

Dans ce modèle, ils ont utilisé la classe « **propagation avant** » de réseau de neurones, ce modèle se compose de 14 couches d'entrée et de 4 couches de sortie. Les couches d'entrée sont les 14 propriétés d'objet identifiées dans MOD et représentées comme suit :

Tableau I. 1.Propriétés à utiliser pour classer un véhicule

1	P1n	Nombre de pixels couverts pour l'objet. Nombre de uns et de zéros entourés de ceux-ci
2	P2n	Nombre de couverts de l'objet
3,4	cx, cy	Centre de la zone
5,6,7,8	E1,E2,E3,E4	Les coordonnées des quatre bords du plus petit rectangle pouvant couvrir l'objet
9	Dx	La distance entre les indices les plus petits et les plus élevés sur l'axe des x
10	Dy	La distance entre les indices les plus petits et les plus élevés sur l'axe des y
11	O	Angle entre l'axe de l'objet et l'axe des x
12	R	Le taux du nombre d'un nombre de pixel dans le plus petit rectangle couvre l'objet
13	Ec	Excentricité de la plus petite ellipse qui recouvre l'objet
14	Di	Diamètre de la plus petite ellipse qui recouvre l'objet

Les couches de sortie sont des nœuds de valeurs binaires et représentent chacun un type de véhicule (grand, petit et moyen véhicule, et non-véhicule). En d'autres termes, ces nœuds de sortie se font concurrence pour représenter l'entrée donnée. Une seule couche cachée est utilisée, qui comprend 25 nœuds. Il est optimisé par l'utilisation de la méthode d'erreur d'essai.

c. Calculateur de densité de trafic (CDT)

Dans les deux premiers sous-modèles (MOD et VI), les trames sont traitées individuellement. Dans ce sous-modèle, toutes les images vidéo pour une période de temps donnée sont traitées ensemble afin de calculer le nombre de véhicules qui ont traversé la route pendant une période donnée. Les trames successives représentent la scène de la route en millisecondes l'une après l'autre. Ainsi, les mêmes véhicules seront vus dans des cadres successifs. Dans ce sous-modèle, le nombre de véhicules qui sont passés par la route pendant une période donnée est compté en utilisant l'emplacement du véhicule dans des trames successives. La densité du trafic est calculée par le nombre de véhicules au cours du temps. Le nombre de véhicules est calculé selon l'algorithme suivant :

```
V_i is calculated based on
AllVech=0
For (k=1 to N) // N is the last frame
   For (j=1 to Detk) //  Detk is the number vehicles detected in frame k
      AllVech=AllVech+1
      Center[AllVech]=c[j]

V_i = 0
Isclose=false;
 while(is AllVech empty)
    select vehicle from AllVech
    [x y] = get the coordinates of selected vehicle
    remove the selected vehicle from AllVech
    for(go back 1 to H number of frames)
       [tempx tempy] =get the coordinates of vehicles in each frame
       if(is tempx and tempy close enough to x and y)
            Isclose = true;
    if(not Isclose)
        Isclose=false;
        V_i = V_i + 1
        Store the selected vehicle for future comparisons
```

Fig I. 2.Programme de comptage de véhicules

I.3 Présentation de l'approche proposée

Notre approche consiste à développer et à concevoir un système intelligent pour la mesure et le contrôle de vitesse des véhicules dans des séquences vidéo, ce qui est l'une des technologies clé au domaine de vision-surveillance dans un trafic routier. Notre travail vise à développer une application software qui sera embarqué par la suite sur des smartphones, pour la détection et la mesure de la vitesse des véhicules en temps réel, en utilisant les techniques de traitement d'images

basées sur les réseaux de neurones convolutionnels (Deep Convolutional Networks) pour l'extraction des informations.

Notre travail sera divisé en deux parties :

La détection des véhicules : en utilisant les méthodes d'apprentissage profondeur (Deep Learning) via les réseaux de neurones. Les modèles de réseaux de neurones convolutionnels seront employés afin d'identifier les contours des véhicules et par la suite les classifier.

La mesure de vitesse en temps réel : en sauvegardant les pixels des différentes positions dans chaque séquence vidéo, et marquer les changements constatés dans ces positions en utilisant un classifieur des véhicule prédéfini et par la suite les convertir en paramètres d'objets réels. La distance de Manhattan est calculée entre les centres de surface des véhicules, la valeur en pixel résultante sera convertie de l'image vers l'objet.

I.4 Conclusion

Dans ce premier chapitre, nous avions présenté quelques travaux de recherche dans le domaine de traitement d'images, et plus précisément la télésurveillance pour le contrôle du trafic routier, chose qui est devenue primordiale afin de minimiser les accidents de circulation.

Nous avions, par la suite, présenté notre approche qui traite la même thématique de recherche, et qui repose sur l'utilisation de l'une des méthodes de l'Intelligence Artificielle : les réseaux de neurones artificiels.

Dans le chapitre suivant, nous allons détailler les méthodes utilisées dans l'Intelligence Artificielle, pour le traitement d'images pour arriver au concept de réseaux de neurones.

Chapitre II : L'approche de l'intelligence artificielle

II.1 Introduction

L'intelligence artificielle (IA) a pour but de faire face à des machines que l'homme arrive à faire lui-même par son intelligence. Les méthodes et techniques mises en œuvre sont variées, pluridisciplinaires et souvent complémentaires : modèles symboliques fondés sur des connaissances, réseaux neuromimètiques, modèles stochastiques,… etc. L'IA a le potentiel à donner une naissance à la nouvelle technologie dans divers domaines tels que l'aide à la décision, le diagnostic, l'interprétation des signaux, la reconnaissance de la parole, le traitement du langage écrit, la robotique, l'interprétation des images et la vision par ordinateur, ainsi que pour les systèmes de détection des véhicules et leurs vitesses. Ces derniers sont devenus, de plus en plus, importants dans la surveillance routière grâce aux approches de l'IA telles que les réseaux de neurones qui ont démontrées de meilleures performances dans ces systèmes et aussi la vision par ordinateur et le traitement d'images.

II.2 L'Intelligence Artificielle

Le terme « intelligence artificiel » (« IA » ou « AI » en anglais pour Artificial Intelligence) a été introduit par **John McCarthy**, c'est la simulation du processus d'intelligence humaine par des machines, en particulier par des systèmes informatiques. Ces processus comprennent l'apprentissage (l'acquisition d'informations et les règles d'utilisation de l'information), le raisonnement (en utilisant les règles pour parvenir à des conclusions approximatives ou définies) et l'auto-correction [3].

II.2.1 Les approches de l'intelligence artificielle

Dès le début de l'IA dans les années 1950, deux grandes approches ont été adoptées par les chercheurs pour concevoir des machines « intelligentes » [4]:

a. L'approche Making a mind :

Que l'on peut qualifier d'IA symbolique, consiste à doter un système d'IA de mécanismes de raisonnement capables de manipuler les données symboliques qui constituent les connaissances d'un domaine. Cette approche fait appel aux modèles et méthodes de la logique. Elle a donné lieu aux systèmes à bases de connaissances.

b. L'approche Modeling the brain :

Que l'on peut qualifier d'IA connexionniste, consiste à s'inspirer du fonctionnement du cortex cérébral. L'entité de base est un modèle formel du neurone, un système étant formé par l'interconnexion d'un grand nombre de tels « neurones ». Cette approche a donné lieu aux réseaux neuromimétiques actuels.

II.2.2 Domaines de l'intelligence artificielle

L'intelligence artificielle occupe plusieurs domaines tels que :

a. Systèmes experts :

L'IA peut réaliser la même tâche qu'une personne humaine, notamment dans la robotique (Ex : logiciel MYCIN en 1974 qui donne des diagnostiques) [5].

b. Calcul formel (opposé au calcul numérique) :

Cela permet d'exécuter de nombreuses formules symboliques (Ex: MATHEMATICA) [6].

c. Représentation des connaissances :

On entend par la représentation symbolique de la connaissance pour que le logiciel soit capable de la manipuler [6].

d. Simulation du raisonnement humain :

Tenter de mettre au point des logiques qui formalisent le mode de raisonnement (logiques modales, floues, temporelles, etc.) [6].

e. Traitement du langage et résolution de problèmes :

Dans les logiciels de traduction automatique, même s'ils ne sont pas tous performants, car c'est une tâche très délicate. Quant à la résolution de problèmes, on trouve souvent l'IA dans les jeux vidéo [5].

f. La reconnaissance :

Qu'elle soit de la parole, de l'écriture ou du visage, l'IA a fait de nombreux progrès dans ces actions [5].

g. L'apprentissage :

Pour que IA puisse faire son devoir, elle doit apprendre les notions données par

L'Homme [5].

h. Domaines de travail :

On trouve l'IA dans divers domaines tels que les banques qui sont dotées des systèmes d'évaluation aux risques de vols de crédits [5].

II.2.3 Avantages et inconvénients de L'IA

a. Avantage :

- Réalisation de toutes les tâches ménagères, (ménage, cuisine, jardinage...), dans le futur l'IA pourra sûrement avoir une apparence humaine pour le contact social [7].

- Limitation des erreurs de calcul grâce aux algorithmes, en utilisant des ordinateurs pour résoudre ces calculs, cela est plus rapide et plus efficace [7].

- Déplacement des véhicules de manière autonome à l'aide des caméras et des capteurs répartis sur ceux-ci [7].

- Reproduction des membres de l'Homme tels que les prothèses qui pourraient servir aux personnes ayant perdu un membre de pouvoir retrouver une vie normale [7].

- Réalisation des tâches pénibles ou dangereuses sur l'Homme sans présentation d'aucune contrainte physique (besoin de nourriture, repos...), ce qui fait qu'elle serait toujours en activité et au service de l'Homme [7].

b. Inconvénients :

- La création de l'IA nécessite des coûts énormes, car sont des machines très complexes, de plus, leur réparation et leur entretien exigent des coûts énormes [8].

- L'IA est considérée comme un don de la nature, les machines n'ont pas d'émotions ni de valeurs morales, elles exécutent ce qui est programmé. Elles ne peuvent pas prendre de décisions si elles rencontrent une situation qui ne leur est pas familière. Elles ne fonctionnent pas correctement ou se décomposent dans de telles situations [8].

- Contrairement aux humains, l'IA ne peut pas être améliorée avec l'expérience. Avec le temps, cela peut conduire à l'usure. Elle stocke beaucoup de données, mais la façon dont elles peuvent être consultées et utilisées est très différente de l'intelligence humaine. Les machines sont incapables de modifier leurs réponses aux environnements changeants [8].

- Le remplacement des humains par des machines peut entraîner un chômage à grande échelle et qui présente un phénomène socialement indésirable [8].

II.3 Vision par Ordinateur (VO)

La vision par ordinateur est une branche de l'intelligence artificielle dont le but est de permettre à une machine de comprendre ce qu'elle « voit » lorsqu'on la connecte à une ou plusieurs caméras [9]. La VO a pour objectif de la construction d'un capteur visuel capable de remplir la fonction de l'œil humain dans un système artificiel, elle doit construire un ensemble matériel et logiciel s'intégrant dans une IA.

Le traitement d'images et vision sont des disciplines relativement jeunes mais qui évoluent rapidement. Elles sont en plein expansion et donnent lieu chaque année à une profusion de travaux, académiques, technologiques et industriels. D'autre part, l'engouement pour ces disciplines s'explique par la multiplication permanente d'applications et d'enjeux industriels dans des domaines aussi variés que : médecine, télécommunications, automobile.

II.3.1 Traitement d'images

Le traitement d'images numériques est un ensemble d'approches, de méthodes, de techniques et d'outils dont l'ambition est de résoudre la majorité des problèmes qui peuvent se présenter dans une image et aussi pour rendre cette opération possible, plus simple, plus efficace et plus agréable, et d'améliorer l'aspect visuel de l'image et d'en extraire des informations jugées [10].

II.3.1.1 Caractéristiques d'une image numérique

a. Dimension :

C'est la taille de l'image et qui se présente sous forme de matrice dont les éléments sont des valeurs numériques représentatives des intensités lumineuses (pixels). Le nombre de lignes de cette matrice multiplié par le nombre de colonnes nous donne le nombre total de pixels dans une image [11].

b. Résolution :

C'est la clarté ou la finesse de détails atteinte par un moniteur ou une imprimante dans la production d'images. Sur les moniteurs d'ordinateurs, la résolution est exprimée en nombre de pixels par unité de mesure (pouce ou centimètre). On utilise aussi le mot résolution pour désigner le nombre total de pixels affichables horizontalement ou verticalement sur un moniteur, plus grand est ce nombre, meilleure est la résolution [11].

c. Bruit :

Un bruit (parasite) dans une image est considéré comme un phénomène de brusque variation de l'intensité d'un pixel par rapport à ses voisins, il provient de l'éclairage des dispositifs optiques et électroniques du capteur [11].

d. Histogramme :

L'histogramme des niveaux de gris ou des couleurs d'une image est une fonction qui donne la fréquence d'apparition de chaque niveau de gris (couleur) dans l'image. Il permet de donner un grand nombre d'informations sur la distribution des niveaux de gris (couleur) et de voir entre quelles bornes est repartie la majorité des niveaux de gris (couleur) dans le cas d'une image trop claire ou d'une image trop foncée. Il peut être utilisé pour améliorer la qualité d'une image (Rehaussement d'image) en introduisant quelques modifications, pour pouvoir extraire les informations utiles de celle-ci. Pour diminuer l'erreur de quantification, pour comparer deux images obtenues sous des éclairages différents, ou encore pour mesurer certaines propriétés sur une image, on modifie souvent l'histogramme correspondant [11].

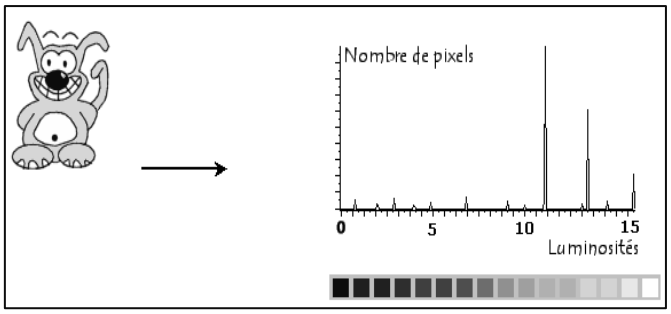

Fig II. 1.Histogramme et palette associés à une image

e. Luminance :

C'est le degré de luminosité des points de l'image. Elle est définie aussi comme étant le quotient de l'intensité lumineuse d'une surface par l'aire apparente de cette surface [11]. Une bonne luminance se caractérise par :

- Des images lumineuses (brillantes)
- Un bon contraste : il faut éviter les images où la gamme de contraste tend vers le blanc ou le noir ; ces images entraînent des pertes de détails dans les zones sombres ou lumineuses.
- L'absence de parasites.

f. Contraste :

C'est l'opposition marquée entre deux régions d'une image, plus précisément entre les régions sombres et les régions claires de cette image. Le contraste est défini en fonction des luminances de deux zones d'images [11].

II.3.1.2 Images à niveaux de gris

C'est la valeur de l'intensité lumineuse en un point, la couleur du pixel peut prendre des valeurs allant du noir au blanc en passant par un nombre fini de niveaux intermédiaires. Donc, pour représenter les images à niveaux de gris, on peut attribuer à chaque pixel de l'image une valeur correspondant à la quantité de lumière renvoyée. Cette valeur peut être comprise par exemple entre 0 et 255. Chaque pixel n'est donc plus représenté par un bit, mais par un octet. Pour cela, il faut que le matériel utilisé pour afficher l'image soit capable de produire les différents niveaux de gris correspondant. Le nombre de niveaux de gris dépend du nombre de bits utilisés pour décrire la « couleur » de chaque pixel de l'image. Plus ce nombre est important, plus les niveaux possibles sont nombreux [11].

II.3.1.3 Images en couleurs

La représentation des couleurs s'effectue de la même manière que les images monochromes avec cependant quelques particularités. On peut représenter les couleurs à l'aide de leurs composantes primaires, les systèmes émettant de la lumière (écrans d'ordinateurs,...) sont basés sur le principe de la synthèse additive : les couleurs sont composées d'un mélange de rouge, vert et bleu (modèle R.V.B.) [11].

II.3.2 Extraction de contours

La détection de contour est une étape préliminaire à de nombreuses applications de l'analyse d'images. Les contours constituent en effet des indices riches, au même titre que les points d'intérêts, pour toute interprétation ultérieure de l'image. Les contours dans une image proviennent : discontinuités de la fonction de réflectance (texture, ombre), et des discontinuités de profondeur (bords de l'objet) ; et sont caractérisés par des discontinuités de la fonction d'intensité dans les images [11].

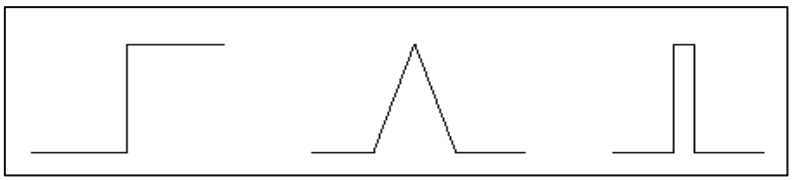

Fig II. 2.Différents types de contours : marche, toit et pointe

Le principe classique de la détection de contours repose sur l'étude des dérivées de la fonction d'intensité dans l'image : les extrema locaux du gradient de la fonction d'intensité et les passages par zéro du laplacien.

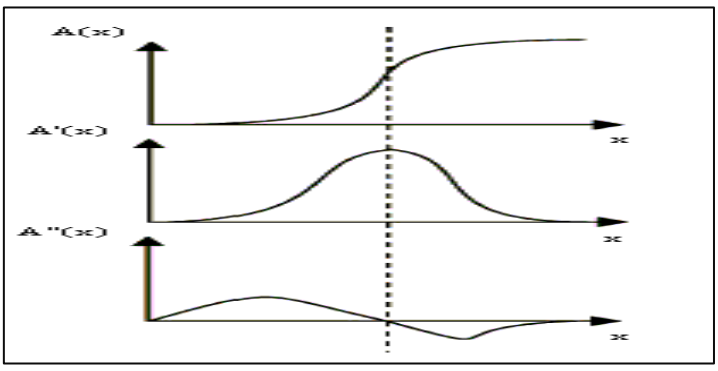

Fig II. 3.Cas bidimensionnel

II.3.3 Filtres pour la détection des contours

La mise en évidence des contours peut se faire notamment par différentiation de l'image, grâce à des filtres détecteurs de Roberts, Sobel ou prewitt [12].

II.3.3.1 Le filtre de Roberts

Le filtre de Roberts est une approche discrète de la dérivée de pas 1 d'une fonction : le gradient de cette fonction. Si $I(x, y)$ représente un pixel dans une image, alors les amplitudes des gradients en x et en y peuvent s'écrire respectivement :

$$Gx = I(x+1, y) - I(x, y) \quad \text{(II. 1)}$$

$$Gy = I(x, y+1) - I(x, y) \quad \text{(II. 2)}$$

Cela revient à convoluer l'image avec les deux filtres :

$Rx = [-1\ 1]$ Et $Ry = trans([-1\ 1])$

L'amplitude du gradient peut être calculée de plusieurs façons :

$$G1(x, y) = sqrt(Gx\text{^}2\ Gy\text{^}2) \quad \text{(II. 3)}$$

$$G2(x, y) = max(abs(Gx), abs(Gy)) \quad \text{(II. 4)}$$

$$Ou : G3(x, y) = abs(Gx) + abs(Gy) \quad \text{(II. 5)}$$

Et la direction du gradient est donne par :

$$D(x, y) = Arctan(Gy\ /\ Gx) \quad \text{(II. 6)}$$

Or, le bruit peut aussi être une brusque variation locale des niveaux de gris, ces filtres sont très sensibles au bruit, car ils accentuent par dérivation le bruit présent dans l'image.

II.3.3.2 Les filtres de Prwitt et de Sobel

Les filtres de Prewitt et de Sobel sont aussi des opérateurs de dérivation, mais on y a introduit un opérateur de lissage. L'image est convoluer avec les masques suivants (prewitt c=1, sobel c=2). L'amplitude peut être calculée de la même manière que pour les filtres de Roberts. Ces filtres sont moins sensibles au bruit que ce de Robert, car le fait d'introduire un moyen local sur le domaine couvert par le masque diminue leur sensibilité. Le filtre Sobel donne une meilleure estimation que celui de Prewitt.

II.3.4 La détection et suivi des objets en mouvement dans une scène vidéo

II.3.4.1 La détection de l'objet

La détection de l'objet comprend la détection d'objets et la reconnaissance de formes dans le cadre d'une séquence vidéo. Un mécanisme de détection d'objet est nécessaire dans n'importe quel procédé de suivi soit dans chaque trame ou lorsque l'objet apparaît en premiers dans la vidéo. Le suivi d'objets dans les séquences vidéo est un sujet très important et à diverses applications dans la compression vidéo, la surveillance, la technologie robotique, etc. Dans de nombreuses applications, l'accent est mis sur le suivi des objets en mouvement [13].

II.3.4.2 Analyse du mouvement dans une séquence d'images

L'analyse de mouvement dans une séquence d'images doit être posée comme un problème d'estimation et de segmentation, puisqu'il s'agit d'appréhender des informations partielles observables et discontinues. C'est un problème particulièrement difficile, mais dont la résolution est cruciale pour la plupart des taches en analyse de scène dynamique. Ce problème peut se présenter en fait sous plusieurs variantes, suivant que l'objectif prioritaire se trouve être l'obtention d'une mesure dense ou paramétrique du mouvement, d'une partition de l'image en régions, ou l'extraction d'entités pertinentes. L'analyse du mouvement dans une séquence d'images repose soit sur la détection du mouvement soit sur la segmentation basée mouvement soit sur la poursuite de mouvement [14].

II.3.4.3 Détection de mouvement

La détection du mouvement dans une séquence d'images consiste à distinguer les zones fixes et mobiles d'une scène. Comme le mouvement d'objets induit des différences temporelles entre images, sa détection s'appuie sur l'étude des variations temporelles de la fonction de luminance : si le capteur est fixe et que l'éclairement de la scène varie peu, alors nous supposons que toute variation temporelle de l'intensité est liée au mouvement d'un objet ou à du bruit [14].

a. Détection de mouvement (Motion detection) :

Il indique uniquement que l'on parle d'une méthode qui a pour objet de trouver en quels points de l'image un mouvement a eu lieu. Un algorithme ayant cet objectif fournit en sortie une variable quantitative (quantité de mouvement) ou qualitative (booléenne) pour tout pixel de chaque image d'entrée.

b. Estimation du mouvement (Motion estimation) :

Cette notion inclut la précédente en y ajoutant la contrainte que le résultat fourni doit être quantitatif. Ce terme est surtout utilisé par les auteurs travaillant sur l'estimation du flux optique ; dans ce cas, le résultat de l'estimation est un champ de vecteurs à deux dimensions représentant la projection sur le plan de l'image du mouvement réel tridimensionnel ayant lieu dans la scène.

c. Modélisation de l'arrière-plan (Background Modeling) :

Cette catégorie regroupe toutes les méthodes de détection de mouvement qui consistent à créer un modèle de l'arrière-plan de la scène filmée (sans aucun objet mobile). Ce modèle peut être une image créée à partir des pixels observes à différents instants de la séquence vidéo.

d. Soustraction de l'arrière-plan (Background substraction) :

La soustraction de l'arrière-plan est l'opération qui suit logiquement la modélisation de l'arrière-plan afin d'obtenir une détection de mouvement. Si le modèle de l'arrière-plan est une image, une différence en valeur absolue entre ce modèle et l'image courante est effectuée afin d'obtenir une détection de mouvement.

e. Segmentation de/par le mouvement (Motion based segmentation) :

Cette tâche va au-delà de la détection de mouvement puisqu'il s'agit de segmenter chaque image en régions qui présentent une homogénéité du mouvement apparent. Cette opération est généralement réalisée à partir d'une estimation du flot optique ou des dérivées spatio-temporelles de l'intensité lumineuse. La détection du mouvement contient les méthodes de différences d'images (approche directe), de corrélation ou de recherche dans l'espace des paramètres. Les méthodes basses niveau exploitent la comparaison pixel à pixel, ou petite région à petite région entre deux images consécutives d'une séquence. Elles permettent de déterminer les régions de variations de l'image dans le temps. Elles nécessitent soit une caméra fixe, soit un recalage préalable dans le cas d'un observateur mobile, afin de détecter uniquement les zones de mouvement dans la scène. Elles sont généralement limitées aux mouvements d'objets rigides et au cas de petits déplacements.

II.3.5 Définition d'un objet vidéo

Un objet vidéo est défini par une forme, une texture et un mouvement (rigide ou non rigide). Cependant, la notion d'objet vidéo est beaucoup plus descriptive qu'une simple région, un objet vidéo peut par exemple être un modèle 3D. La notion d'objet ne fait pas forcément référence à un objet du monde réel. En effet, dans le domaine de la vidéo, un objet n'est pas nécessairement un objet d'une scène 3D, mais plutôt le résultat de l'analyse de la projection d'un monde 3D sur un plan. Ainsi, un objet vidéo est défini comme une région de la vidéo conforme à un modèle. On peut par exemple avoir pour modèle : un modèle de mouvement ou un modèle d'objet physique [14].

II.3.6 Suivi d'objets dans la vidéo

Le suivi d'objets quelconques dans une séquence vidéo réelle est une tâche très délicate. Surtout quand ces objets sont non-rigides, le fond de la scène n'est pas fixe et dans le cas de plusieurs objets mobiles dans la même scène. Lorsqu'il s'agit de suivre des objets particuliers comme le visage, la main, le bras. La tâche est plus simple, car le modèle et les contraintes de

variation de ces objets sont connus a priori. D'autres critères de type géométrique et/ou statique peuvent aussi être introduits dans l'identification de ces objets [14].

II.3.6.1 Mesures de la distance des objets dans une séquence vidéo

II.3.6.1.1 Fonctions de similarité utilisées

Un système de classification d'images calcule la similarité visuelle entre le descripteur de l'image requête et les descripteurs des images de la base. Différents types de mesure de similarité sont présentés dans la littérature. Un de ces types correspond aux distances géométriques entre vecteurs. Dans ce cas, on parle de distances, car ces mesures ont la propriété de respecter les axiomes des espaces métriques. Un espace métrique E se définit comme un ensemble non vide doté d'une application d, appelée distance, de E x E dans R^+, vérifiant les axiomes suivants : $\forall\, x, y, z \in E$:

1. $d(x,y) = 0 \rightarrow x = y$ ***identit***é.
2. $d(x,y) = d(x,y) \rightarrow$ ***symétrie***
3. $d(x,y) + d(y,z) \geq d(x,z) \rightarrow$ ***in*égal*ité triangulaire***

a. La distance de Manhattan (L_1) :

L'approche la plus simple pour mesurer la similarité entre deux images correspond aux distances de Minkowski (Fondateur des deux distances (L_1) et (L_2)). Cette distance L_r est définie par :

$$\boldsymbol{L_r(I_1, I_2) = [\sum_{i=1}^{n} |I_1(i) - I_2(i)|^r]^{\frac{1}{r}}} \qquad \text{(II. 7)}$$

Où r ≥ 1 est le facteur de Minkowski et n la dimension de l'espace caractéristique. Les métriques de Minkowski représentent un bon compromis entre efficacité et performance. Pour cette famille de distances, plus le paramètre $\boldsymbol{r}$ augmente, plus la distance L_r aura tendance à favoriser les grandes différences entre coordonnées. Ces distances sont rapides à calculer et simples à implémenter, par contre leur calcul est réalisé en considérant que chaque composante du vecteur apporte la même contribution à la distance. Pour $r = 1$, on obtient la distance de Manhattan :

$$\boldsymbol{L_1(I_1, I_2) = [\sum_{i=1}^{n} |I_1(i) - I_2(i)|]} \qquad \text{(II. 8)}$$

Cette norme est aussi connue sous le nom city-block est plus appropriée pour mesurer la similarité entre les données multivariées ; elle est moins sensible au bruit coloré que la distance euclidienne.

b. La distance Euclidienne (L_2) :

Pour $r = 2$, on obtient la distance Euclidienne :

$$L_2(I_1, I_2) = \sqrt{\sum_{i=1}^{n}(I_1(i) - I_2(i))^2} \qquad \text{(II. 9)}$$

La distance euclidienne est invariable aux translations et aux rotations des données dans l'espace des attributs et couramment utilisée dans des espaces à 2 ou 3 dimensions, cette métrique donne de bons résultats si l'ensemble des données présente des classes compactes et isolées.

c. La métrique de corrélation :

Cette distance prend en considération la corrélation entre les données ; de plus, elle n'est pas dépendante de l'échelle de données. Elle est ainsi définie par :

$$Dist_{corr}(I_1, I_2) = (I_1 - I_2)\sum^{-1}(I_1 - I_2)^T \qquad \textit{(II. 10)}$$

Où $\sum$ est la matrice covariance entre l'ensemble des descripteurs d'images. La corrélation tient compte de la distribution statistique des données dans l'espace, c'est ce qui la différencie des autres distances.

II.4 Les réseaux de neurones

II.4.1 Neurone biologique

Un neurone est une cellule du système nerveux, capable de communiquer et de traiter des informations. On le trouve dans le cerveau, mais aussi dans la moelle épinière et les nerfs optiques. Le neurone aux caractéristiques d'une cellule, il est composé d'un corps cellulaire contenant un noyau, un cytoplasme, une membrane plasmique… Mais lui seul possède des axones et des dendrites [15].

II.4.1.1 Propriétés de neurone biologique

Le neurone biologique a plusieurs propriétés qui sont citées ci-dessous :

- Le diamètre de la cellule est compris entre 0.01 et 0.05mm. [16]

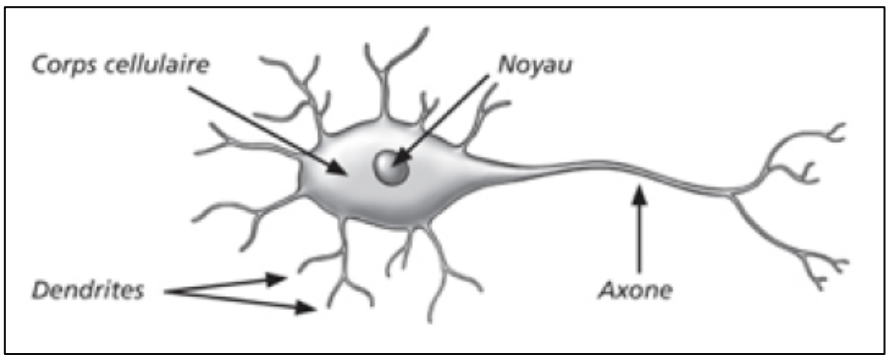

Fig II. 4.structure du neurone biologique

La dendrite reçoit les signaux des autres neurones en entrée.

Le corps cellulaire (Soma) additionne tous les signaux entrants pour générer une valeur d'entrée.

Axone :quand la somme atteint un certain seuil, le neurone se déclenche et le signal est transmis aux autres neurones.

Synapses qui est le point d'interconnexion d'un neurone avec les autres, le niveau du signal transmis dépend de la force (poids synaptique) des connections.

De manière générale, un **réseau de neurones** est un réseau interconnecté de milliard de neurones et de trillions d'interconnexions entre eux. [17]

II.4.2 Les réseaux de neurones artificiels

Un réseau de neurones artificiel RNA (ou ANN en anglais pour **Artificial Neural Network**) est un réseau de neurones inspiré par la biologie, et configuré pour exécuter des actions spécifiques bien définies. Les RNA sont des réseaux fortement connectés de processeur élémentaire (neurones) fonctionnant en parallèle [18]. Le RNA est un ensemble de neurones formels interconnectés, il est basé sur un modèle de neurone simplifié et il peut reproduire certaines fonctions du cerveau comme la reconnaissance des formes, le traitement de langage naturel, la mémorisation associative ou bien l'apprentissage. Mais ces neurones formels n'ont pas toutes les capacités des neurones biologiques. [15]

II.4.2.1 Définition du neurone artificiel

L'élément de base d'un réseau de neurones est le neurone artificiel, aussi appelé neurone formel, est un élément qui est fortement inspiré par le système nerveux biologique [19]. Il est utilisé dans le fonctionnement de l'IA et copie le fonctionnement du neurone biologique [20]. On peut représenter un neurone artificiel de cette façon : chaque neurone reçoit un nombre d'entrées des neurones amont à chacune de ces entrées est associé un poids (la force de la connexion), chaque neurone à une sortie unique qui se ramifie pour alimenter des neurones en aval [21].

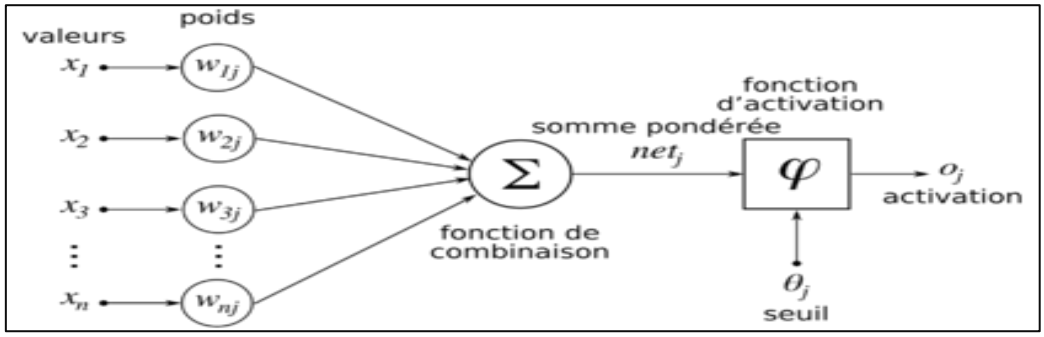

Fig II. 5.fonctionnement de base d'un neurone formel

Avec :

- $(X1, X2 \ldots . Xn)$ sont les entrées.
- $(W1j, W2j, \ldots, Wnj)$ sont les poids synaptiques du neurone j.
- $(Netj)$ est la sortie de l'unité de sommation

Pour un nombre compris entre $j = 1$ et un nombre quelconque n, le neurone formel va calculer la somme de ses entrées $(x1, \ldots, xn)$, pondérées par les poids synaptiques $(w1, \ldots, wn)$, et la comparer à son seuil Θ. Si le résultat est supérieur au seuil, alors la valeur renvoyée est 1, sinon la valeur renvoyée est 0 [19].

Un réseau de neurones artificiels ressemble au cerveau humain de 2 façons principales :

Un réseau de neurones artificiel acquiert la connaissance grâce à l'apprentissage, tout comme pour les réseaux biologiques.

La connaissance d'un réseau de neurones artificiel est stockée dans les connexions entre les neurones avec une valeur/force appelée "poids synaptique" (weight) tout comme pour les réseaux biologiques.

II.4.3 Architecture des réseaux de neurones

On distingue deux types d'architecture : les réseaux de neurones **non bouclés** et les réseaux de neurones **bouclés** (ou **récurrents**).

II.4.3.1 Les réseaux de neurones non bouclés

Un réseau de neurones non bouclé est représenté graphiquement par un ensemble de neurones interconnectés, l'information circulant des entrées vers les sorties sans retour en arrière. Le graphe d'un réseau non bouclé est acyclique. Il convient d'insister sur le fait que le temps ne joue aucun rôle fonctionnel dans un réseau de neurones non bouclé : si les entrées sont constantes, les sorties le sont également. Le temps nécessaire pour le calcul de la fonction réalisée par chaque neurone est négligeable et on peut considérer ce calcul comme instantané, ils sont souvent appelés « réseaux statiques ». Ils sont utilisés en classification de reconnaissance des formes (caractère, parole, ...) et en prédiction. La sortie de ce type de réseau est une fonction algébrique non linéaire de ses entrées et de ses paramètres [18]. L'architecture la plus utilisée est celle du réseau à couches « percéptron »

a. Percéptron monocouche :

C'est un réseau simple il se compose d'une seule couche d'entrée et d'une seule couche de sortie. Le percéptron est le premier modèle de réseau de neurones avec algorithme d'apprentissage

créé par Frank Rosenblatt. Il est calqué à la base sur le système visuel, il a été conçu dans le but de reconnaissance de formes.

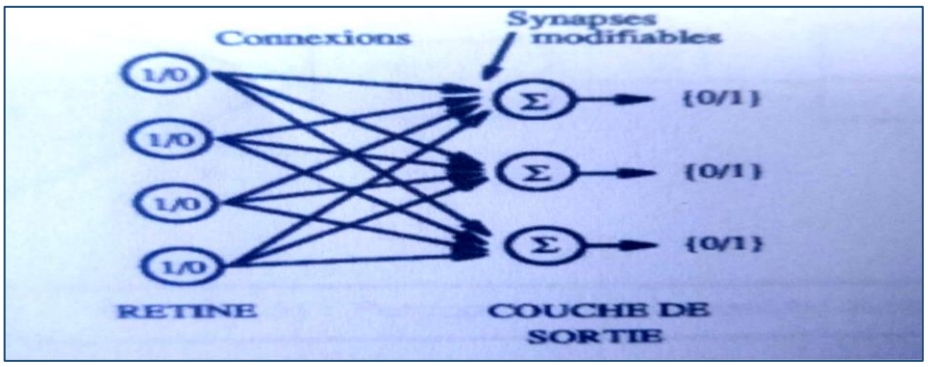

Fig II. 6.le percéptron classique

Les cellules de la première couche répondent en oui /non. La réponse 'oui' correspond à la valeur '1' et la réponse 'non' correspond à la valeur '0' à la sortie du neurone. Les cellules d'entrées sont reliées aux cellules de sortie grâce à des synapses d'intensités variables. L'apprentissage du percéptron s'effectue en modifiant l'intensité de ces synapses. Il suit généralement un apprentissage supervisé selon la règle de correction de l'erreur (ou selon la règle de Hebb). Cependant, il peut aussi être utilisé pour faire de la classification et pour résoudre des opérations logiques simples [22]

b. Perceptron multicouches 'PMC' :

Noté aussi MLP pour Multi Layer Perceptron, est un réseau de neurones qui est défini par un ensemble de neurones interconnectés afin de permettre l'approximation des rapports non linéaires arbitraire d'entrées–sorties. Cela constitue une extension du modèle de percéptron. Dans les réseaux multicouches, les neurones sont disposés en couches successives. C'est-à-dire chaque neurone dans une couche est connecté à tous les neurones de la couche précédente et de la couche suivante (excepté pour les couches d'entrée et de sortie) et il n'y a pas de connexion entre les cellules d'une même couche [22].

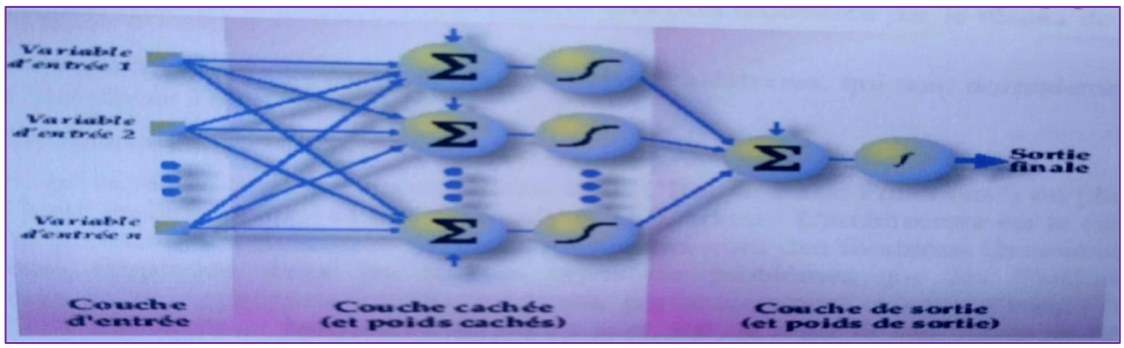

Fig II. 7.Réseau de neurones multicouches

Le réseau de neurones applique une transformation non linéaire à chacun des neurones des couches cachées et les valeurs résultantes sont combinées de façon linéaire pour obtenir la valeur prédite.

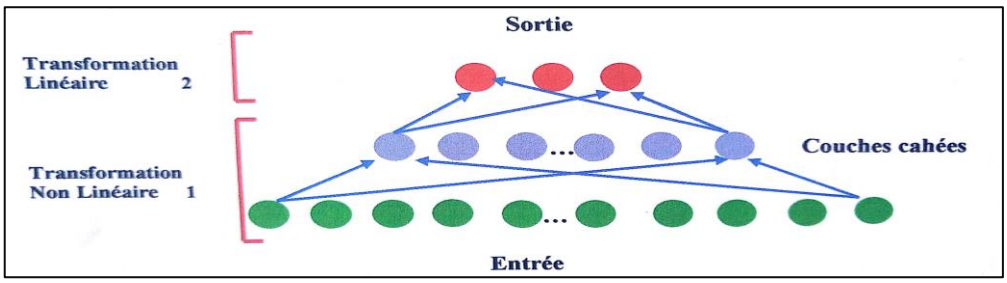

Fig II. 8.Rôle des couches caches d'un MLP

Les fonctions d'activation dans les neurones des couches cachées sont nécessaires pour introduire la non-linéarité dans les réseaux. Elles sont principalement les fonctions sigmoïdes, par contre la couche de sortie se compose normalement des neurones linéaire qui calculent seulement une somme pondérée de toutes ses entrées [10]. Il peut résoudre des problèmes non linéairement séparables et des problèmes logiques plus compliqués, il suit aussi un apprentissage supervisé selon la règle de correction d'erreur. Cette architecture est également appelée réseau à deux couches puisqu'il y a deux couches de poids ajustables : celle qui relie les entrées aux neurones cachés, et celle qui relie les neurones cachés aux neurones de sortie. Le choix du nombre de neurones cachés est très important : plus ce nombre est important, plus le nombre de degrés de liberté est élevé et plus la fonction modélisée par le réseau de neurones peut être complexe. Initialement, tous les poids peuvent avoir des valeurs aléatoires, qui sont normalement les très petits avants de commencer l'apprentissage.

c. Réseaux à fonction radiale :

Nommés aussi RBF pour Radial Basic Function proposé par J.Moody et C.Darken. L'architecture est la même que pour les PMC, cependant les fonctions de base utilisées sont les fonctions Gaussiennes. Les RBF sont employés dans les mêmes types de problèmes que les PMC à savoir en classification et en approximation de fonction. L'apprentissage le plus utilisé pour les RBF est le mode hybride, et les règles sont les règles de correction des erreurs [20].

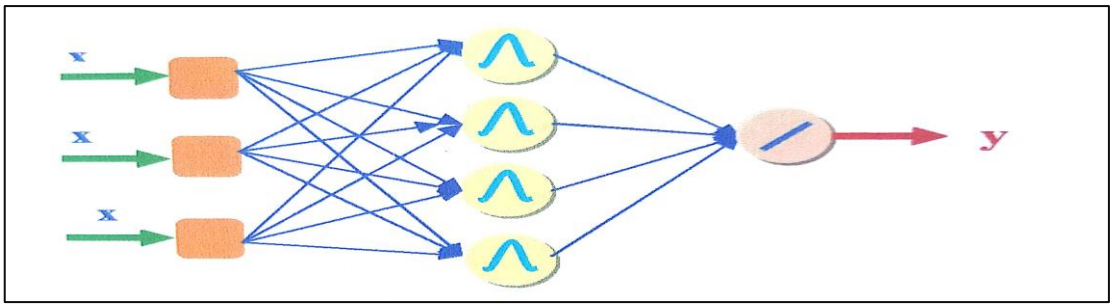

Fig II. 9.les réseaux à fonction radiale

II.4.3.2 Les réseaux de neurones bouclés

Appelés aussi réseaux récurrents, sont des réseaux dans lesquels il y a un retour en arrière de l'information FEED-BACK [22], c'est-à-dire qu'il y a la présence d'au moins une boucle de rétroaction. Par conséquent, le graphe des réseaux bouclés est cyclique. La sortie d'un neurone du réseau peut donc en fonction d'elle-même, cela n'est évidemment concevable que si la notion de temps est explicitement prise en considération. Ainsi, à chaque connexion d'un réseau de neurones bouclé est attaché. Un réseau de neurones bouclé peut avoir une topologie de connexion quelconque, il est donc un système dynamique [18].

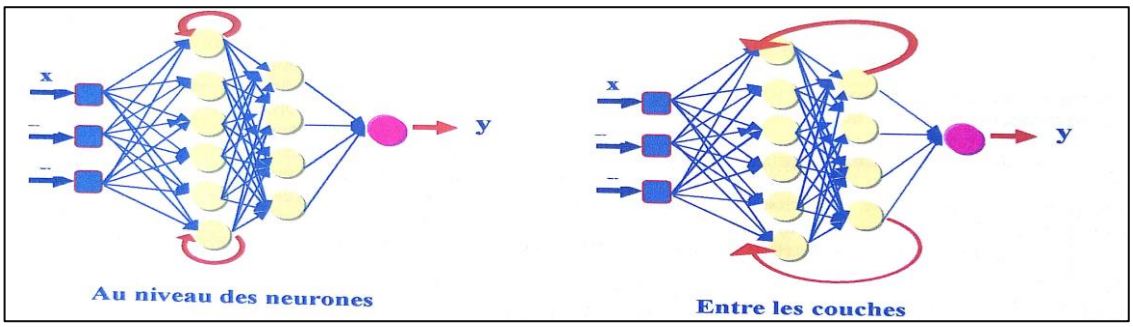

Fig II. 10.Réseaux de neurones bouclés

II.4.4 L'apprentissage des réseaux de neurones

La recherche des algorithmes d'apprentissage statistiques a mené à d'importantes découvertes au cours des deux dernières décennies, qui ont changé la manière de concevoir le problème de rendre les ordinateurs plus intelligents. Un agent est intelligent parce qu'il a des connaissances opérationnelles qui lui permettent d'effectuer certaines taches ou de répondre à certaines questions sur un certain domaine. L'objectif de l'apprentissage est de fournir une méthode au réseau, afin qu'il puisse ajuster ses paramètres lorsqu'on lui présente des exemples à traiter [23].

On distingue trois types d'apprentissage : supervisé, non supervisé et hybride. En effet, les réseaux de neurones ont été appliqués avec succès à l'apprentissage de tâches de classification et

d'approximation de fonctions. L'apprentissage est le processus d'adaptation des paramètres d'un système pour donner une réponse désirée à une entrée ou stimulation quelconque [23].

II.4.4.1 Apprentissage supervisé

Dans ce type d'apprentissage, on fournit au réseau de neurones la donnée à traiter, mais aussi la sortie attendue. Le réseau effectue une évaluation de la donnée, puis compare la valeur obtenue avec la valeur désirée, il va ensuite modifier ses paramètres internes afin de minimiser l'erreur constatée. L'apprentissage supervisé consiste à calculer les coefficients de telle manière que les sorties du réseau de neurones soient aussi proches que possible des sorties désirées, qui peuvent être : la classe d'appartenance de la forme que l'on veut classer, la valeur de la fonction que l'on veut approcher ou de la sortie du processus que l'on veut modéliser, ou encore la sortie souhaitée du processus à commander.

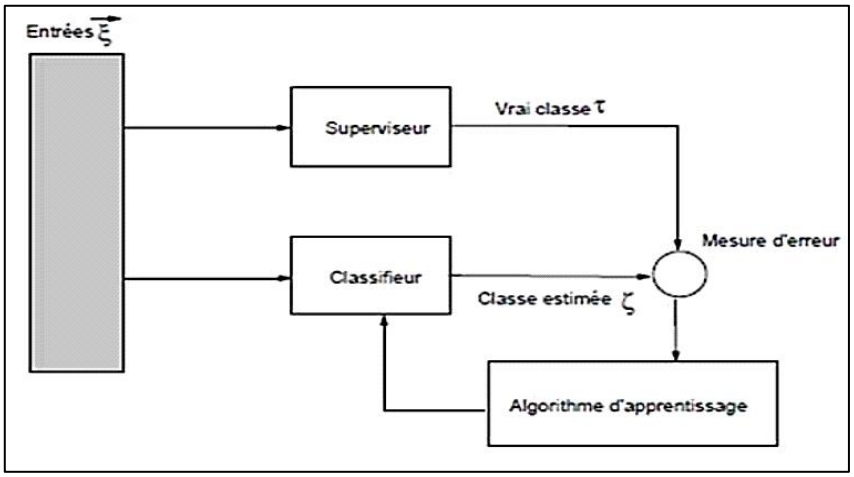

Fig II. 11.réseaux à apprentissage supervisé

La plupart des algorithmes d'apprentissage des réseaux de neurones sont des algorithmes d'optimisation : ils cherchent à minimiser, par des méthodes d'optimisation non linéaire, une fonction de coût, qui constitue une mesure de l'écart entre les réponses réelles du réseau et ses réponses désirées. Cette optimisation se fait de manière itérative, en modifiant les poids en fonction du gradient de la fonction de coût : le gradient est estimé par une méthode spécifique aux réseaux de neurones, dite méthode de rétro propagation, puis il est utilisé par l'algorithme d'optimisation. Les poids sont initialisés aléatoirement avant l'apprentissage, puis modifiés itérativement, jusqu'à obtention d'un compromis satisfaisant entre la précision de l'approximation sur l'ensemble d'apprentissage et la précision de l'approximation sur un ensemble de validation, distinct du précédent. Contrairement à des affirmations maintes fois répétées, l'apprentissage des réseaux de neurones n'est pas spécialement lent : il existe des algorithmes d'optimisation non linéaire

extrêmement rapides qui permettent de faire des développements industriels sur de simples PC. Le but fondamental de l'apprentissage est de bien généraliser à de nouveaux cas.

II.4.4.2 Apprentissage non-supervisé

Dans ce type d'apprentissage, aucune information n'est fournie au réseau en plus des données à apprendre. Celui-ci est amené à découvrir la structure sous-jacente des données afin de les organiser en clusters. L'apprentissage non-supervisé correspond au cas où aucune cible n'est prédéterminée. Ainsi, l'ensemble d'entraînement ne contient que des entrées et ne définit pas explicitement la nature de la fonction f qui doit être retournée par l'algorithme d'apprentissage. C'est plutôt l'utilisateur qui doit spécifier le problème à résoudre. Pour ce problème, f doit fournir une estimation de la fonction de densité ou de probabilité de la distribution ayant généré les éléments d'entrées. Pour le problème de la classification, les cartes auto-organisatrices de Kohnen sont utilisées dans les réseaux de neurones artificiels.

Dans l'apprentissage non supervisé, le réseau modifie ses paramètres en tenant compte seulement des informations locales. Ces méthodes n'ont pas besoin de sorties désirées préétablies. Les réseaux utilisant cette technique sont appelés réseaux à dynamique autonome et sont considérés comme des détecteurs de régularité, car le réseau apprend en détectant les régularités dans la structure des motifs d'entrée et produit la sortie la plus satisfaisante.

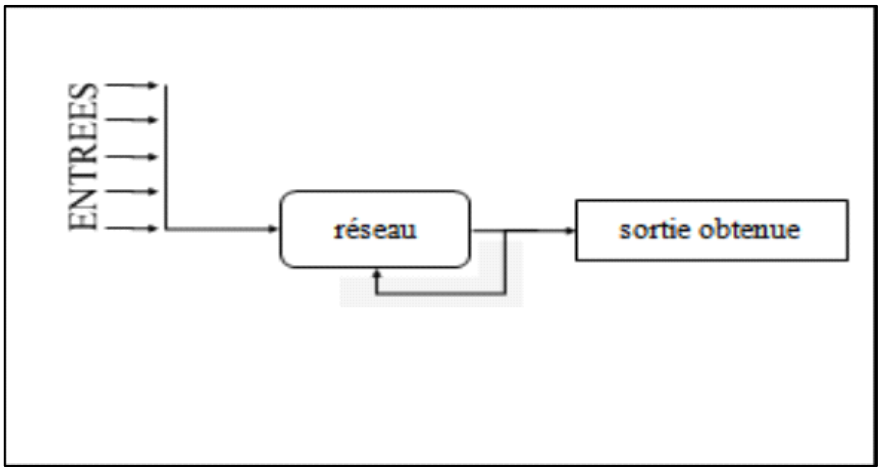

Fig II. 12. Réseau à apprentissage non supervisé

II.4.4.3 Apprentissage hybride

Ce type d'apprentissage est plus rare et encore mal exploré, cette approche combine des méthodes numériques (réseaux de neurones, algorithmes génétiques,…etc.) et des méthodes symboliques. Certains auteurs utilisent le terme d'apprentissage hybride pour parler d'un couplage 'supervisé / non-supervisé' ; dans ce cas, il s'agit d'un réseau qui met en parallèle ou en série un réseau entraîné en mode supervisé et un autre en mode non-supervisé.

II.4.5 Règles d'apprentissage des réseaux de neurones artificiels

Règle d'apprentissage ou processus d'apprentissage est une méthode ou une logique mathématique. Il améliore les performances du réseau de neurones artificiel et applique cette règle sur le réseau. Ainsi, les règles d'apprentissage actualisent les poids et les niveaux de polarisation d'un réseau lorsqu'un réseau simule dans un environnement de données spécifique [24].

L'application de la règle d'apprentissage est un processus itératif. Il aide un réseau de neurones à apprendre des conditions existantes et à améliorer ses performances [24].

II.4.5.1 Règle d'apprentissage Hebbian

La règle Hebbian était la première règle d'apprentissage du réseau neuronal non supervisé. Utilisée pour identifier, comment améliorer le poids des nœuds d'un réseau. La règle d'apprentissage Hebb suppose que si deux neurones voisins sont activés et désactivés en même temps, le poids reliant ces neurones devrait augmenter. Pour les neurones opérant dans la phase opposée, le poids entre eux devrait diminuer. S'il n'y a pas de corrélation de signal, le poids ne devra pas changer. Lorsque les entrées des deux nœuds sont positives ou négatives, un fort poids positif existe entre les nœuds. Si l'entrée d'un nœud est positive et négative pour les autres, un fort poids négatif existe entre les nœuds.

Au début, les valeurs de tous les poids sont mises à zéro, cette règle d'apprentissage peut être utilisée pour les fonctions d'activation logicielle et matérielle. Puisque les réponses souhaitées des neurones ne sont pas utilisées dans la procédure d'apprentissage, il s'agit de la règle d'apprentissage non supervisée. Les valeurs absolues des poids sont généralement proportionnelles au temps d'apprentissage, ce qui n'est pas souhaitable. La règle d'apprentissage Hebbian décrit la formule comme suit :

$$\mathbf{W_{ij}} = \mathbf{X_i} * \mathbf{X_j} \qquad \text{(II. 11)}$$

II.4.5.2 Règle d'apprentissage Percéptron

Chaque connexion dans un réseau de neurones a un poids associé, qui change au cours de l'apprentissage. Selon celui-ci, un exemple d'apprentissage supervisé, le réseau commence son apprentissage en attribuant une valeur aléatoire à chaque poids. Calculer la valeur de sortie sur la base d'un ensemble d'enregistrements pour lesquels nous pouvons connaître la valeur de sortie attendue, c'est l'exemple d'apprentissage qui indique la définition complète. En conséquence, il est appelé un échantillon d'apprentissage. Le réseau compare ensuite la valeur de sortie calculée avec la valeur attendue, puis calcule une fonction d'erreur $\in$, qui peut être la somme des carrés des erreurs survenues pour chaque individu dans l'échantillon d'apprentissage :

$$\sum_i \sum_j (E_{ij} - O_{ij})^2 \qquad \text{(II. 12)}$$

La première sommation est effectuée sur les individus de l'ensemble d'apprentissages et la deuxième sommation sera effectuée sur les unités de sortie. $Eij\ et\ Oij$ Sont les valeurs attendues et obtenues de la $\mathbf{j}^{\text{ème}}$ unité pour le $\mathbf{i}^{\text{éme}}$ individu. Le réseau ajuste ensuite les poids des différentes unités, en vérifiant chaque fois si la fonction d'erreur a augmenté ou diminué. Comme dans une régression classique, il s'agit de résoudre un problème de moindres carrés. Depuis l'attribution des poids des nœuds en fonction des utilisateurs, il s'agit d'un exemple d'apprentissage supervisé.

II.4.5.3 Règle d'apprentissage Delta

Développé par Widrow et Hoff, la règle du delta est l'une des règles d'apprentissage les plus courantes. Cela dépend de l'apprentissage supervisé. Cette règle stipule que la modification du poids sympatrique d'un nœud est égale à la multiplication de l'erreur et de l'entrée. Sous forme mathématique, la règle delta est la suivante :

$$\Delta W = \eta(t - Y)X_i \qquad \text{(II. 13)}$$

Pour un vecteur d'entrée donné, il sera comparé avec le vecteur de sortie. Si la différence est zéro, aucun apprentissage n'a lieu ; sinon, ses poids seront ajustés pour réduire cette différence. Le changement de poids de $\mathbf{u_i}$ en $\mathbf{u_j}$ est $\mathbf{dw_{ij}} = r * a_i * e_j$ Où : $\mathbf{r}$ est le taux d'apprentissage, $\mathbf{a_i}$ représenté l'activation de $\mathbf{u_i}$, et $\mathbf{e_j}$ est la différence entre la sortie attendue et la sortie réelle de $\mathbf{u_j}$. Si l'ensemble de modèles d'entrée forme un ensemble indépendant, alors des associations arbitraires sont apprises en utilisant la règle delta. Pour les réseaux avec des fonctions d'activation linéaires et sans unités cachées, l'erreur au carré par rapport au graphique de poids est un paraboloïde dans l'espace n. Puisque la constante de proportionnalité est négative, le graphe d'une telle fonction est concave vers le haut et a la plus petite valeur. Le sommet de ce paraboloïde représente le point où il réduit l'erreur. Le vecteur de poids correspondant à ce point est le vecteur de poids idéal. Nous pouvons utiliser la règle d'apprentissage delta avec une unité de sortie unique et plusieurs unités de sortie. Tout en appliquant la règle delta, nous supposons que l'erreur peut être mesurée directement. Le but de l'application de la règle delta est de réduire la différence entre la sortie réelle et la sortie attendue qui est l'erreur.

II.4.5.4 Règle d'apprentissage de corrélation

Basée sur un principe similaire à la règle d'apprentissage Hebbian, il suppose que les poids entre les neurones répondeurs devraient être plus positifs, et les poids entre les neurones ayant une réaction opposée devraient être plus négatifs. Contrairement à la règle de Hebbian, la règle de corrélation est l'apprentissage supervisé. La réponse $\mathbf{o_j}$, la réponse désirée, $\mathbf{d_j}$ est utilisée pour le

calcul du changement de poids. Sous forme mathématique, la règle d'apprentissage par corrélation est la suivante :

$$\Delta W_{ij} = \eta X_i d_j \qquad (II.\ 14)$$

Où $\mathbf{d_j}$ est la valeur désirée du signal de sortie. Cet algorithme d'apprentissage commence généralement avec l'initialisation des poids à zéro. Depuis l'attribution du poids souhaité par les utilisateurs, la règle d'apprentissage par corrélation est un exemple d'apprentissage supervisé.

II.4.5.5 Règle d'apprentissage Outstar

Nous utilisons la règle d'apprentissage OutStar lorsque nous supposons que des nœuds ou des neurones d'un réseau sont disposés dans une couche. Ici, les poids connectés à un certain nœud devraient être égaux aux sorties désirées pour les neurones connectés à travers ces poids. La règle de départ produit la réponse souhaitée t pour la couche de n nœuds. Ce type d'apprentissage est appliqué à tous les nœuds d'une couche particulière. La mise à jour des poids pour les nœuds sont comme dans les réseaux neurones de Kohonen. Sous forme mathématique, l'apprentissage de l'OutStar est exprimé comme suit :

$$W_{jk} = \begin{cases} \eta(y_k - w_{jk}) & \textit{Si le noeud j gagne la compétition} \\ 0 & \textit{Si le noeud j perd la compétition} \end{cases} \qquad (II.\ 15)$$

C'est une procédure de formation supervisée, car les résultats souhaités doivent être connus.

II.5 L'apprentissage en profondeur (Deep Learning)

Le Deep Learning est une méthode de **Machine Learning** qui consiste à enseigner à des ordinateurs. L'entraînement des modèles s'effectue via un vaste ensemble de données labellisées et d'architectures de réseaux de neurones qui contiennent de nombreuses couches. [25]

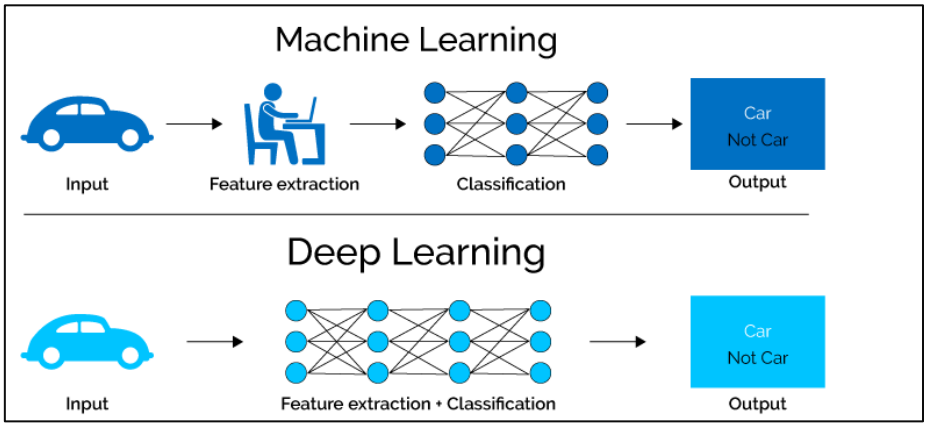

Fig II. 13.Le principe de Deep Learning

L'apprentissage en profondeur permet aux modèles de calcul composés de plusieurs couches de traitement d'apprendre des représentations de données avec plusieurs niveaux d'abstraction. L'apprentissage en profondeur découvre une structure complexe dans de grands ensembles de données en utilisant l'algorithme de propagation arrière pour indiquer comment une machine devrait modifier ses paramètres internes utilisés pour calculer la représentation dans chaque couche à partir de la représentation dans la couche précédente. Les réseaux convolutives profonds ont apporté des avancées décisives dans le traitement des images, de la vidéo, de la parole et de l'audio, alors que les réseaux récurrents ont éclairé des données séquentielles telles que le texte et la parole [26].

II.5.1 Les réseaux de neurones convolutifs profonds

Le premier réseau de neurones convolutives (CNN) a été introduit à la fin des années 80 par LeCun. C'est le premier réseau de neurones pour la reconnaissance d'images. Il permettait la reconnaissance de chiffres manuscrits [27].

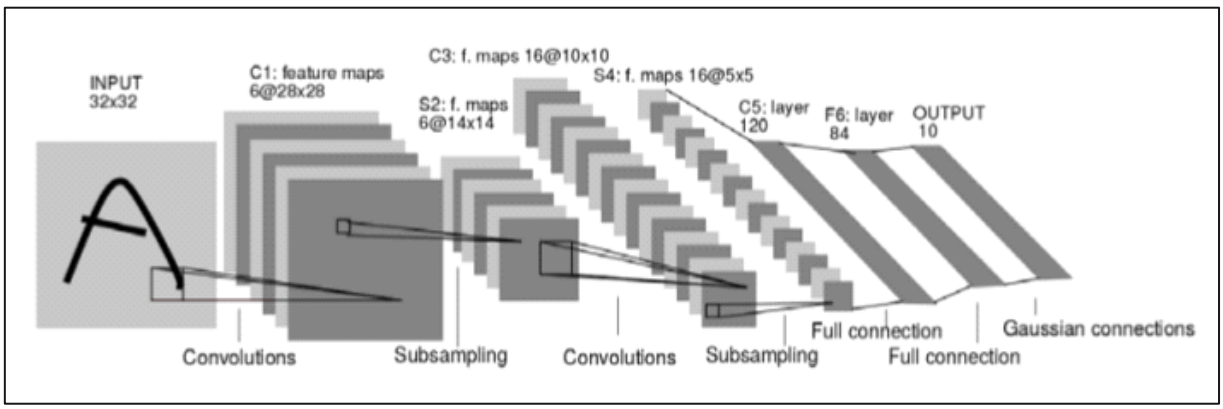

Fig II. 14.Petit réseau convolutifs pour la reconnaissance de chiffres manuscrits

L'idée est de passer l'image dans une succession de filtres convolutives apportant une description réduite et pertinente de l'image. Ces caractéristiques sont, par la suite, envoyées à un percéptron multicouche composé de **couches cachées** et d'**une couche de sortie** complètement connectées permettant la classification du chiffre présent dans l'image. Les filtres de convolution et les couches complètement connectée sont appris simultanément. Les CNN sont un type particulier de réseaux de neurones applicables facilement à des images pour capter spatialement de l'information. Les CNN peuvent être vus comme un assemblage de modules en série permettant l'extraction de caractéristiques de manière hiérarchique à partir des pixels d'une image [27].

II.5.1.1 Différents modules d'un réseau de neurones convolutifs

Les différents modules utilisés dans les CNN sont : les convolutions, l'agglomération (pooling), les fonctions d'activation, le dropout, la Batchnormalization et les fonctions d'erreur classiques utilisées pour l'apprentissage.

a. La convolution :

La pièce maîtresse d'un CNN est la couche convolutive. La sortie en résultant est appelée carte de caractéristiques. Une couche convolutive est constituée de plusieurs filtres (ou noyaux) de convolution à appliquer sur une matrice d'entrée (une image ou une carte de caractéristique précédente).

Une convolution est définie comme une opération mathématique décrivant une règle pour la fusion de deux ensembles d'informations. Il est important à la fois en physique et en mathématiques et définit un pont entre le domaine temporel et le domaine fréquentiel grâce à l'utilisation de transformer de Fourier. L'opération de convolution (Fig II.15) est connue sous le nom de détecteur de caractéristiques d'un CNN. L'entrée d'une convolution peut être une donnée brute ou une sortie de carte d'entités d'une autre convolution [28].

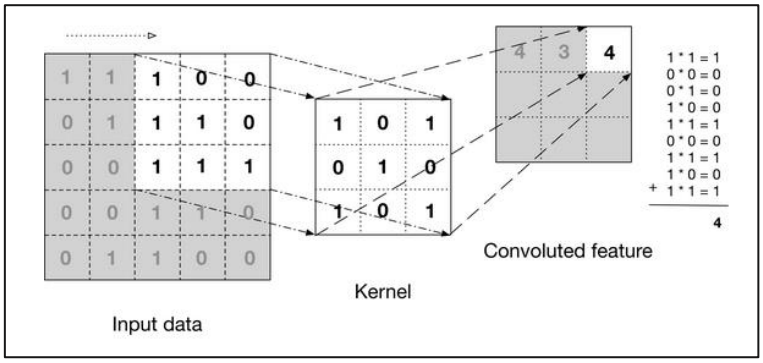

Fig II. 15.Opération de convolution

b. Le pooling :

Les couches pooling (ou agglomérations) sont généralement insérées entre des couches convolutives successives pour réduire progressivement la taille (largeur et hauteur) de la représentation des données, et contrôler le sur-ajustement. La couche pooling fonctionne indépendamment sur chaque tranche de profondeur de l'entrée [28].

La couche pooling permet de rajouter de l'invariance spatiale lors de l'extraction de caractéristiques tout en réduisant la dimension des entrées. Elle peut être de différentes natures, mais les types de pooling les plus utilisés sont le **Max Pooling** qui renvoie l'élément maximum sur une fenêtre de calcul (Fig II.16) et l'**Average pooling** qui renvoie la moyenne des éléments sur une fenêtre de calcul.

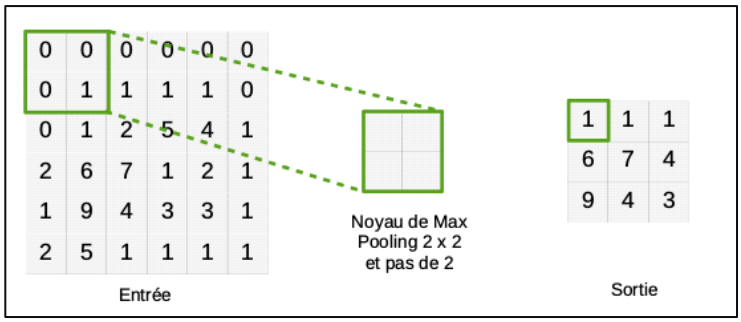

Fig II. 16.Illustration du Max Pooling (le noyau du pooling est de taille 2 × 2 et appliqué tous les deux pixels (stride = 2). Le max des quatre éléments sur une fenêtre de l'entrée est gardé)

c. Les fonctions d'activation :

La fonction d'activation ou fonction de transfert est un nœud qu'on ajoute à la sortie de n'importe quel réseau de neurones, il peut également être attaché entre deux réseaux de neurones. Il est utilisé pour déterminer la sortie du réseau de neurones comme oui ou non, il mappe les valeurs résultantes entre 0 à 1 ou -1 à 1 etc. (selon la fonction) [29]. Il existe différentes fonctions d'activation permettant la non-linéarité dans les différentes couches des CNN :

- **Sigmoid or Logistic Activation Function :** La fonction sigmoïde utilisée notamment dans le percéptron multicouches original

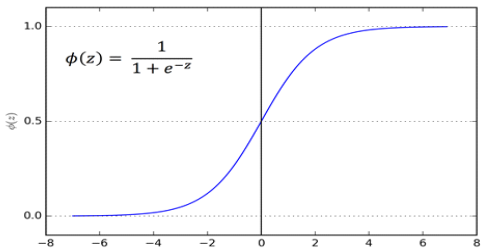

Fig II. 17.Fonction sigmoid

- Tanh ou tangente hyperbolique Fonction d'activation

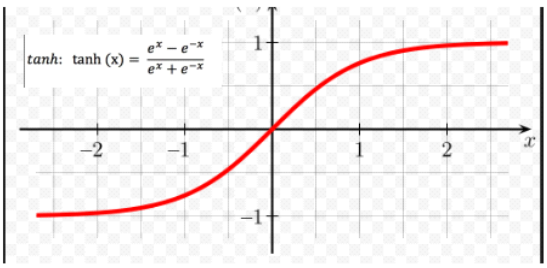

Fig II. 18.Fonction Tanh

- **Fonction d'activation unité linéaire rectifiée ReLU (Rectified Linear Unit) :** c'est la fonction d'activation non linéaire la plus simple et la plus utilisée dans presque tous les réseaux de neurones convolutionnels ou d'apprentissage en profondeur. Si l'entrée obtenue est positive, la dérivée est juste 1, donc il n'y

a pas l'effet de compression qu'on peut rencontrer sur les erreurs propagées en arrière de la fonction sigmoïde. La recherche a montré que les ReLU entraînent une formation beaucoup plus rapide pour les grands réseaux. La plupart des frameworks tels que TensorFlow et TFLearn simplifient l'utilisation des ReLU sur les couches cachées, nous n'avons donc pas besoin de les implémenter nous-même [30]. Il existe d'autres fonctions d'activation de la même famille que les ReLU comme les LReLU, les PReLU et les eLU.

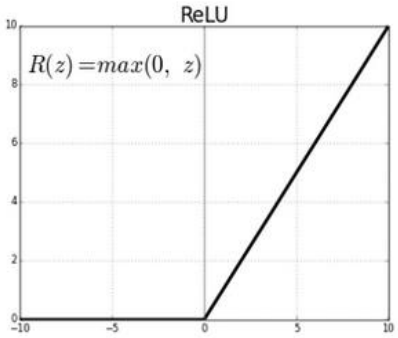

Fig II. 19.Fonction ReLU

Le ReLU dans la figure II.16 est à moitié rectifié (à partir du bas). R (z) est nul lorsque z est inférieur à zéro et R(z) est égal à z lorsque z est supérieur ou égal à zéro. Le problème, c'est que toutes les valeurs négatives deviennent immédiatement nulles, ce qui diminue la capacité du modèle à s'adapter ou à s'entraîner correctement à partir des données. Cela signifie que toute entrée négative donnée à la fonction d'activation ReLU transforme la valeur en zéro immédiatement dans le graphique, ce qui affecte le graphique résultant en ne mappant pas les valeurs négatives de manière appropriée [29].

d. Le dropout :

Pour éviter le sur-apprentissage (overfitting), la couche de dropout a été introduite [Srivastava 2014]. C'est une technique qui aborde ces deux problèmes : elle empêche le sur-ajustement et fournit un moyen de combiner de manière exponentielle plusieurs réseaux neurones de différentes architectures efficacement. Le terme «dropout» fait référence à des unités de décrochage (cachées et visibles) dans un réseau de neurones. En supprimant une unité, nous voulons dire la retirer temporairement du réseau, ainsi que toutes ses connexions entrantes et sortantes, comme le montre la Figure II.20 [31] .

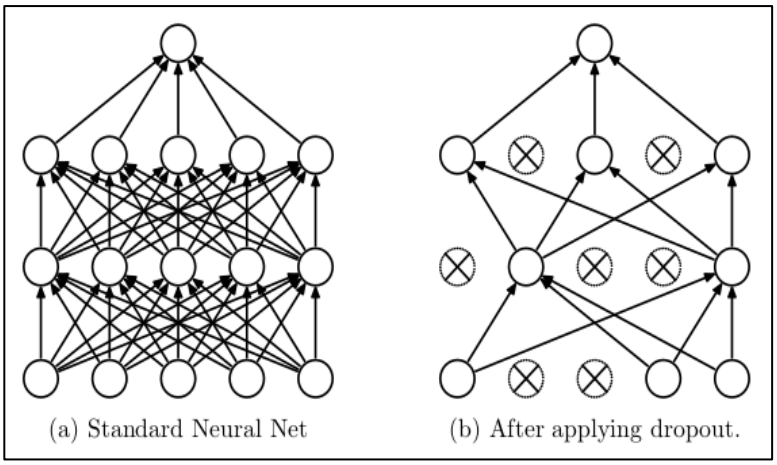

Fig II. 20.Dropout Neural Net Model - (a) Un réseau de neurones standard avec 2 couches cachées. (b) Exemple d'un réseau aminci produit en appliquant une suppression sur le réseau sur la gauche. Les unités en croix ont été supprimées.

e. La Batchnormalization :

Cette technique a été introduite par Ioffe en 2015, afin d'apprendre les CNN de manière plus rapide et efficace. C'est une technique qui permet d'améliorer les performances et la stabilité des réseaux de neurones, et qui permet également aux architectures d'apprentissage en profondeur plus sophistiquées de fonctionner. L'idée de la Batchnormalization est de normaliser les entrées de chaque couche afin que les distributions de celles-ci soient de moyenne nulle et de variance unitaire. Durant l'apprentissage, les couches de Batchnormalization apprennent des paramètres (un facteur d'échelle et un biais) permettant d'ajuster cette normalisation : ces paramètres permettent d'appliquer une transformation sur la distribution normalisée : si le réseau considère que la distribution normalisée n'est pas adaptée pour une couche donnée, il apprend les paramètres permettant de l'ajuster.

f. Les fonctions de perte :

Il existe plusieurs fonctions de perte (ou fonctions objectif) utilisables pour l'apprentissage des réseaux de neurones. Ces fonctions sont dépendantes de la tâche que le réseau doit effectuer (classification, régression...).

- **La fonction de perte Softmax :** utilisée pour l'optimisation du réseau de classification d'images, elle permet la maximisation de la probabilité qu'à une entrée d'appartenir à une classe plutôt qu'à une autre.
- **La fonction de perte par entropie croisée sigmoïde :** permettant une régression sur des probabilités.

- **La fonction de perte euclidienne :** utilisée pour des problématiques de régression sur des valeurs réelles.
- **La fonction de perte L1 lisse :** introduite par Girshick en 2015 qui permet une meilleure optimisation pour les problèmes de régression.

II.6 Conclusion

Les approches de l'IA offrent un très haut avantage aux systèmes de surveillance routière par l'utilisation de l'apprentissage en profondeur, via des réseaux de neurones. L'utilisation de la VO et le traitement d'images pour la détection des véhicules ainsi que leurs vitesses, a donné une naissance à de nouvelles technologies.

Chapitre III : Implémentation et discussion des résultats

III.1 Introduction

La détection des véhicules est basée sur l'extraction des caractéristiques des images dans des séquences vidéo suivies du classement de ces images selon plusieurs critères, ceci en utilisant les réseaux de neurones convolutionnels. En outre, la détection de vitesse est réalisée par la mesure de la distance entre la caméra et le véhicule afin de calibrer la vitesse du véhicule dans la séquence. La distance de Manhattan sera calculée entre les centres de surface des véhicules, la valeur en pixel résultante sera convertie de l'image vers l'objet.

III.2 Détection de véhicules

L'identification des véhicules se fait en temps réel à l'aide des réseaux de neurones**,** l'ensemble de données est composé d'images extraites de la base de données d'images de véhicules GTI. Des exemples vont être extraits de la vidéo de test, l'ensemble de données est étiqueté avec deux classes : les voitures seront étiquetées avec **1**, alors que les non-voitures ont une étiquette de **0**, comme illustré dans la figure suivante :

Fig III. 1.Classement des captures : véhicule 1, non-véhicule 0

L'ensemble de données sera ensuite subdivisé en un ensemble d'apprentissage 90%, et un ensemble de validation 10%, de la distribution suivante. Nous pouvons voir que l'ensemble de données est bien équilibré, ce qui est important pour l'entraînement du réseau de neurones (Fig III.2).

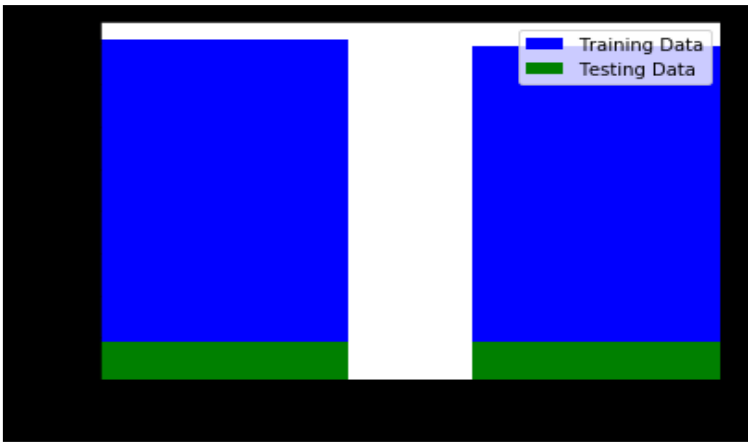

Fig III. 2.Ensemble de données d'apprentissage (en bleu) et de validation (en vert)

Chaque image de la vidéo sera introduite dans le réseau de neurones, qui sera utilisé pour détecter les véhicules partout dans le cadre, ce dernier produit une image comme une carte de chaleur virtuelle. Les boîtes de délimitation seront ensuite dessinées sur les positions chaudes comme on peut le voir ci-dessous :

Fig III. 3.Véhicules détectés en utilisant le réseau de neurones

Pour chaque source de chaleur détectée, une boîte de délimitation unique peut être dessinée qui identifie essentiellement les véhicules dans les images (Fig III.4).

Fig III. 4.Détection des sources de chaleur

Les paramètres du réseau de neurones (poids) seront par la suite gelés et stockés pour une utilisation future, car le réseau de neurones doit être entraîné qu'une seule fois.

III.3 Détection de vitesse

La vidéo sera lue à l'aide du package vidéo Scikit, toutes les images vidéo avec les véhicules détectés ayant des boîtes englobantes dessinées à l'aide du classificateur Cascade seront dans un fichier de sortie. Par la suite, les images vidéo seront itérées plus d'une fois et leurs coordonnées ou valeurs de pixels absolues des positions des véhicules détectés seront obtenues, en utilisant le classificateur HOG afin de suivre la région d'intérêt des véhicules présents dans chaque trame. Une fois que les régions d'intérêt de tous les véhicules dans les cadres sont obtenues, les points médians des boîtes de délimitation entourant les véhicules seront obtenus, ce qui est considéré comme la "position" du véhicule qui sera la clé pour de futurs cadres. Ce processus sera répété pour tous les véhicules afin de créer une liste de valeurs intermédiaires mises à jour à chaque image.

Afin de déterminer la vitesse de n'importe quel objet, les paramètres de base qui sont nécessaires, sont : la distance et le temps. Dans notre travail, nous allons effectuer tous les calculs dans le plan image, puis nous allons les projeter dans le plan objet. Dans ce cas, la distance parcourue par un véhicule particulier dans le plan de l'image correspond aux différences relatives des pixels à diverses positions du véhicule dans les trames suivantes. Le temps serait le taux auquel les images étaient traitées, ce que nous supposions être 30 images par seconde. Le temps est calculé par la relation suivante :

$$\boldsymbol{T = \frac{N_{pixels}}{fps}} \qquad \text{(III. 1)}$$

T : Temps.

N : Nombre de pixels progressé par véhicule.

fps : Images par seconde.

La distance **d** est calculée par la relation suivante :

$$\boldsymbol{d = \sqrt{mid_1{}^2 + mid_2{}^2}} \qquad \text{(III. 2)}$$

Où mid1 et mid2 sont les positions du véhicule dans les images 1 et 2 respectivement. Si la différence est inférieure à 10 pixels, la voiture est supposée stationnaire et est négligée lors de la mise à jour des valeurs.

Tous les calculs seront effectués dans le plan image, puis sont projetés dans le plan objet. Nous convertissons les valeurs de pixels obtenues en **mm / sec** qui est représentatif du plan de l'objet, en utilisant la distance de l'objet de la caméra dans la formule suivante :

$$d_{ext} = \frac{L_f * h_{Robj} * h_{im}}{h_{obj} * h_C} \qquad \text{(III. 3)}$$

L_f : Longueur focale (mm)

h_{Robj} : Hauteur réelle de l'objet (mm)

h_{im} : Hauteur de l'image (pixels)

h_{obj} : Hauteur de l'objet (pixels)

h_C : Hauteur du capteur (mm)

Cependant, le classificateur entraîné est incapable de détecter toutes les voitures dans chaque trame et ainsi les valeurs de vitesse obtenues ne sont pas toujours précises et sujettes à dévier de la valeur d'origine en fonction des hypothèses suivantes sur les paramètres. La hauteur de l'objet dans le plan de l'objet est supposée être de 4 mètres en général se rapprochant des valeurs médianes pour toutes les voitures qui ne sont pas les mêmes dans tous les cas. La distance entre la caméra et la route est approximée ce qui n'est pas précis en raison de contraintes pratiques (Fig III.5).

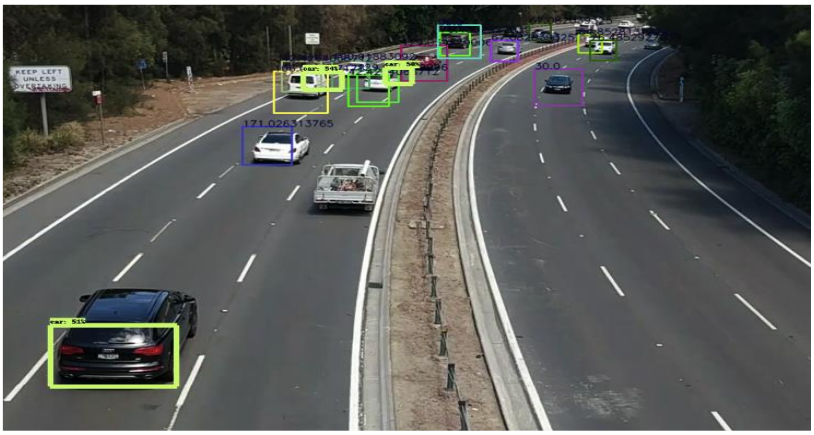

Fig III. 5.Les voitures détectées et leurs vitesses

III.4 Outils & langages de programmation

III.4.1 Langage XML

Le langage XML (eXtended Markup Language) est un format général de documents orienté texte. Il est aussi bien utilisé pour le stockage de documents que pour la transmission de données entre applications. Sa simplicité, sa flexibilité et ses possibilités d'extension ont permis de

l'adapter à de multiples domaines allant des données géographiques au dessin vectoriel en passant par les échanges commerciaux.

De nombreuses technologies se sont développés autour de XML et enrichissent ainsi son environnement. Le langage XML dérive de SGML (Standard Generalized Markup Language) et de HTML (HyperText Markup Language). Comme ces derniers, il s'agit d'un langage orienté texte et formé de balises qui permettent d'organiser les données de manière structurée.

Un document XML est simplement une suite de caractères respectant quelques règles. Il peut être stocké dans un fichier et/ou manipulé par des logiciels en utilisant un codage de caractères. Ce codage précise comment traduire chaque caractère en une suite d'octets réellement stockés ou manipulés.

Comme un document XML est seulement du texte, il peut être écrit comme l'exemple ci-dessous. Écrit dans un fichier « bibliography.xml ». Ce document représente une bibliographie de livres sur XML. Ce dernier contient une liste de livres avec pour chaque livre, 1e titre, l'auteur, l'éditeur, l'année de parution, le numéro ISBN et éventuellement une URL.

III.4.1.1 Base de données XML

Une base de données XML est réalisée pour le stockage de données structurées en langage XML. On peut la caractériser par le fait qu'elle offre à l'utilisateur une vision logique des données en accord avec le modèle XML (organisation hiérarchique des informations...), de la même façon que les bases de données relationnelles présentent une vision logique des données conforme au modèle relationnel (organisation des informations sous forme de tables). Cette vision logique conforme au modèle XML permet d'envisager aisément d'utiliser les standards définis autour de XML (XQuery, XPath, XUpdate) pour accéder et traiter les données de la base.

III.4.1.2 Avantages de XML

L'avantage de l'utilisation des bases de données XML réside dans le fait qu'elles manipulent directement le format XML, ce qui permet de mettre rapidement en œuvre des services web d'accès à de telles bases :

- le stockage des fichiers XML se fait directement. (pas besoin de définir et d'écrire des règles de transformations XML ? relationnel / relationnel ? XML)

- de même, la restitution des documents est immédiate alors qu'il faut le reconstituer à partir des tables d'une base de données relationnelle.

- la publication des résultats de requêtes est très aisée à partir des bases de données XML.

III.4.2 Langage Python

Développé à l'origine par Guido van Rossum en 1989, Python est un langage de programmation de haut niveau interprété (il n'y a pas d'étape de compilation) et orienté objet avec une sémantique dynamique. C'est un langage simple, facile à apprendre et permet une bonne réduction du coût de la maintenance des codes. Les bibliothèques (packages) python encouragent la modularité et la réutilisabilité des codes. Python et ses bibliothèques sont disponibles (en source ou en binaires) sans charges pour la majorité des plateformes et peuvent être redistribués gratuitement [32].

III.4.2.1 Caractéristiques de python

- Python est un langage qui continue à évoluer, soutenu par une communauté d'utilisateurs enthousiastes et responsables, dont la plupart sont des supporters du logiciel libre. Parallèlement à l'interpréteur principal, écrit en ***C*** et maintenu par le créateur du langage, un deuxième interpréteur, **Jython**, écrit en **Java**, est en cours de développement
- Python est dynamiquement typé, tout objet manipulable par le programmeur possède un type bien défini à l'exécution, qui n'a pas besoin d'être déclaré à l'avance.
- Python est dynamique, l'interpréteur peut évaluer des chaînes de caractères représentant des expressions ou des instructions Python. Il est aussi orthogonal, d'où un petit nombre de concepts suffit à engendrer des constructions très riches. Python est réflectif, car il supporte la méta-programmation, par exemple la capacité pour un objet de se rajouter ou de s'enlever des attributs ou des méthodes, ou même de changer de classe en cours d'exécution. Un grand nombre d'outils de développement, comme le debugger ou le profiler, sont implantés en Python lui-même.
- Python gère ses ressources (mémoire, descripteurs de fichiers...) sans intervention du programmeur, par un mécanisme de comptage de références (proche, mais différent, d'un ramasse-miettes).
- Python est gratuit, mais on peut l'utiliser sans restriction dans des projets commerciaux.

III.4.2.2 Avantages de python

- La syntaxe de Python est très simple et, combinée à des types de données évolués (listes, dictionnaires,...), conduit à des programmes à la fois très compacts et très lisibles. A fonctionnalités égales, un programme Python (abondamment commenté et

présenté selon les canons standards) est souvent de 3 à 5 fois plus court qu'un programme *C* ou *C++* (ou même *Java*) équivalent, ce qui représente en général un temps de développement de 5 à 10 fois plus court et une facilité de maintenance largement accrue.

- Python convient aussi bien à des scripts d'une dizaine de lignes qu'à des projets complexes de plusieurs dizaines de milliers de lignes.
- Python est portable, non seulement sur les différentes variantes d'Unix, mais aussi sur les OS propriétaires : MacOS, BeOS, NeXTStep, MS-DOS et les différentes variantes de Windows. Un nouveau compilateur, baptisé JPython, est écrit en Java et génère du bytecode Java.
- Python intègre, comme Java ou les versions récentes de C++, un système d'exceptions, qui permettent de simplifier considérablement la gestion des erreurs.

III.5 L'interface Homme-Machine (IHM)

L'interface graphique désigne la manière dont est présenté un logiciel à l'écran pour l'utilisateur. C'est le positionnement des éléments : menus, boutons, fonctionnalités dans la fenêtre. Une interface graphique bien conçue est ergonomique et intuitive : faite pour que l'utilisateur la comprenne tout de suite.

L'interface graphique est le langage d'échange entre l'humain et la machine. L'ordinateur affiche à l'écran les éléments que nous comprenons et que nous interprétons. Chaque système dispose de sa propre interface graphique :

- le tableau de bord dans les voitures
- le distributeur de billet à la banque
- le téléviseur
- l'ordinateur
- la tablette
- le smartphone

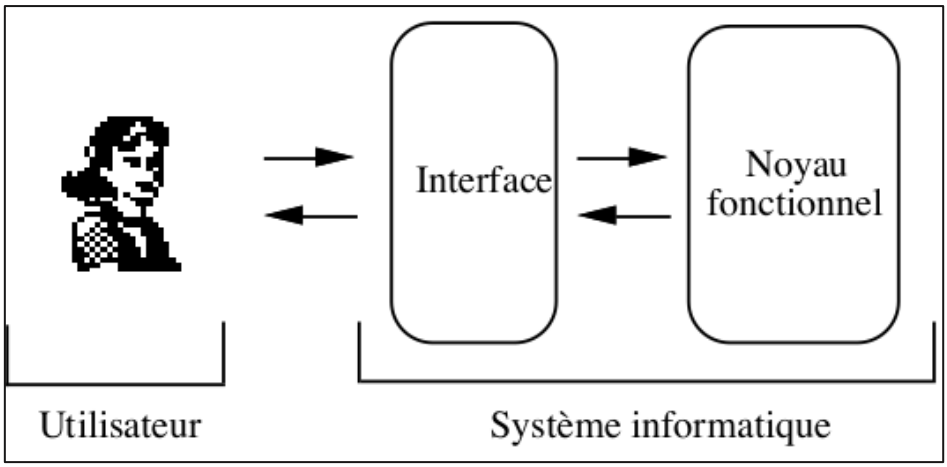

Fig III. 6.Un système informatique et son utilisateur.

III.6 Interface graphique en python (Tkinter)

Tkinter est une boite à outils pour réaliser des interfaces graphiques en Python. Elle repose sur un autre langage de script : le Tcl (Tool Command Language), en utilisant la bibliothèque Tk de ce langage. Tkinter est l'abréviation de «Tk interface».

On dispose de plusieurs éléments graphiques appelés «widgets», certains sont destinés à en contenir d'autres, c'est notamment le cas des fenêtres. La règle étant que tout widget doit être contenu dans un autre, excepté un widget spécial, qui est la fenêtre racine.

Pour créer une interface graphique avec Tkinter, il y a deux choses à faire : créer une fenêtre racine et lancer la boucle principale via la méthode **mainloop ( )**. L'appel à cette méthode bloque l'exécution de l'appelant. Tkinter est alors à l'écoute des événements provenant de l'utilisateur, tels que des clics de souris.

```
from tkinter import *
root = Tk() # Création de la fenêtre racine
root.mainloop() # Lancement de la boucle principale
```

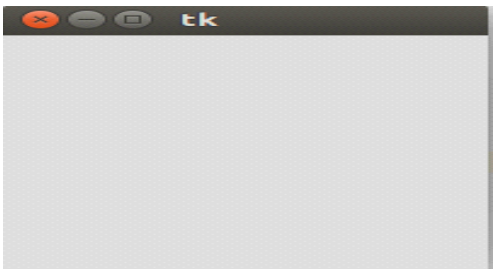

Fig III. 7.Fenêtre racine créée par la méthode **mainloop**

Pour créer un label avec le texte « Hello », on procède comme suit :

```
from tkinter import *
root = Tk() # Création de la fenêtre racine
label = Label(root, text='Hello')
root.mainloop() # Lancement de la boucle principale
```

Lorsqu'on exécute notre code, notre label ne s'affiche pas, il faut positionner notre label après l'avoir créé. Pour positionner un widget, Tkinter propose trois gestionnaires de position. Chacun permet de placer les widgets avec une philosophie différente.

Tableau III. 1.Les fonctions de la gestionnaire de position

Gestionnaire de Position	Description
Pack	Le conteneur est coupé en deux zones. L'une de ces zones occupée par le widget et l'autre la zone libre, La zone libre sera de nouveau coupée en deux au positionnement du prochain widget.
Place	Positionne le widget au pixel près. Ce gestionnaire est difficile à utiliser dans la pratique
grid	Découpe le conteneur en grille et place le widget dans une cellule de cette grille.

Pour placer un widget, il faut appeler les méthodes **pack(), place()** ou **grid()** du widget.

a. La fonction pack :

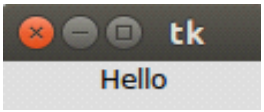

Fig III. 8.Fenêtre racine munie d'un label avec la méthode **pack**

```
from tkinter import *
root = Tk() # Création de la fenêtre racine
label = Label(root, text='Hello')
label.pack()
root.mainloop() # Lancement de la boucle principale
```

b. La fonction grid :

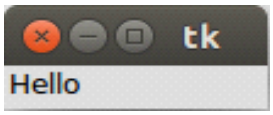

Fig III. 9.Fenêtre racine munie d'un label avec la méthode **grid**

```
from tkinter import *
root = Tk() # Création de la fenêtre racine
label = Label(root, text='Hello')
label.grid()
root.mainloop() # Lancement de la boucle principale
```

Pour la création des boutons, on utilise le widget **Button**, qui prend en paramètre un texte à afficher et une fonction à exécuter.

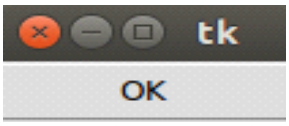

Fig III. 10.Fenêtre racine munie d'un bouton avec la méthode Button

```
from tkinter import *
def Hello():
print("hello!")
root = Tk()
button = Button(root,text = "OK", width=10, height=1,command=Hello)
button.pack()
root.mainloop()
```

Tkinter propose une dizaine de widgets, ceux-ci peuvent prendre en paramètre les options suivantes :

Tableau III. 2.La description des paramètres des options standards d'un widget

Options standard	**Description**
Foreground	La couleur du texte du widget.
Activeforeground	La couleur du texte lorsque le curseur est sur le widget.
Background	Couleur de fond.
Activebackground	Couleur de fond lorsque le curseur est sur le widget (valable uniquement pour les widgets cliquables).
Padx	Marge horizontale entre les bords du widget et son contenu.
Pady	Marge verticale entre les bords du widget et son contenu.
Width	Largeur du widget en taille de police.
Height	Hauteur du widget en taille de police.

Il est possible de créer une liste dans laquelle l'utilisateur peut faire un ou plusieurs choix à l'aide du widget **Listbox.** Pour remplir la liste avec des valeurs, il faut utiliser la méthode **insert(),** Cette méthode prend en premier paramètre l'endroit où l'on souhaite insérer l'élément, puis le ou les éléments à insérer.

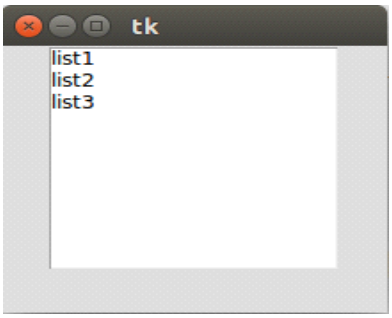

Fig III. 11.Fenêtre racine munie d'une liste

```
from tkinter import *

root = Tk()
lb = Listbox(root)
lb.insert(1, "list1")
lb.insert(2, "list2")
lb.insert(3, "list3")
lb.pack()
root.mainloop()
```

Pour lire une image, il faut utiliser le module **PIL** et le widget **image**

Fig III. 12.Fenêtre racine munie d'une image

```
from tkinter import *

from PIL import Image, ImageTk
root = Tk()
image = Image.open("index.jpeg")
photo = ImageTk.PhotoImage(image)
Label(root, image=photo).pack()
root.mainloop()
```

III.7 Implémentation de l'application

III.7.1 Architecture du Réseau de Neurones utilisé

Le modèle que nous présentons (Fig III.13, Tableau III. 3) est composé de 3 couches de convolution et d'une couche de maxpooling. L'image en entrée est de taille (64*64) pixels, l'image passe d'abord à la première couche de convolution. Cette couche est composée de 128 filtres de taille (3*3) pixels, la fonction d'activation ReLU est utilisée, cette dernière force les neurones à retourner des valeurs positives, après cette convolution 128 images caractérisées (features maps) seront créés de taille (64*64) pixels. Ces dernières seront fournies en entrée de la deuxième couche de convolution qui est composée aussi de 128 filtres. La fonction d'activation ReLU est appliquée sur cette couche. À la sortie de cette couche, nous aurons 128 feature maps de taille (64*64) pixels.

Par la suite, les 128 feature maps qui sont obtenus seront données en entrée de la troisième couche de convolution qui est composée aussi de 128 filtres. La fonction d'activation ReLU est appliquée sur cette couche. Le Maxpooling sera appliqué ensuite pour réduire la taille de l'image. À la sortie de cette couche, nous aurons 128 feature maps de taille 8*8.

Après ces trois couches de convolution, nous utilisons un réseau de neurones composé de deux couches complètement connectées (fully connected). La première couche est composée de 128 neurones où la fonction d'activation utilisée est le ReLU, la deuxième couche utilise la fonction hyperbolique tangent (tanh).

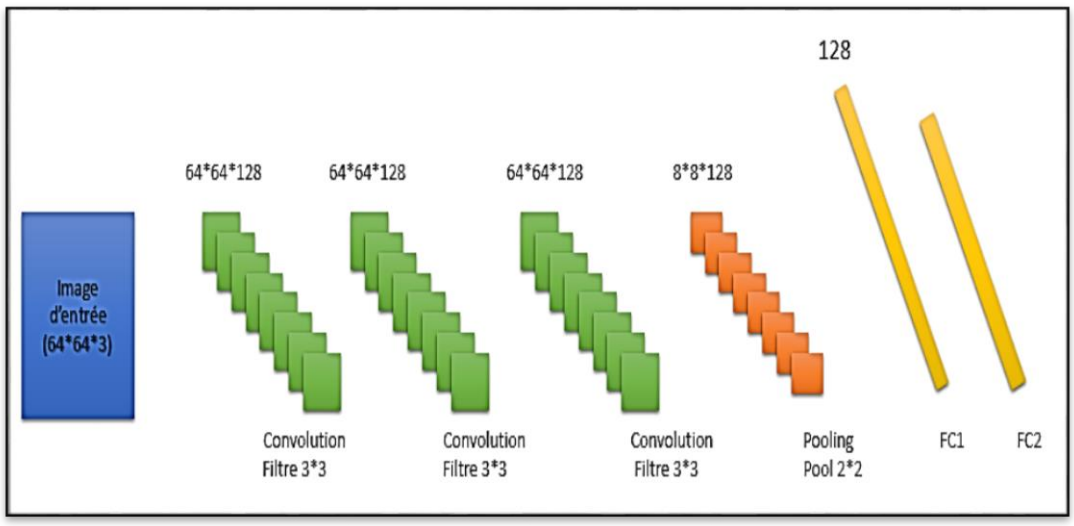

Fig III. 13.Architecture du modèle en couche du Réseau de neurones

Tableau III. 3.L'architecture complète du réseau de neurones

Layer (type)	Output Shape	Param #
lambda_1 (Lambda)	(None, 64, 64, 3)	0
conv1 (Conv2D)	(None, 64, 64, 128)	3584
dropout_1 (Dropout)	(None, 64, 64, 128)	0
conv2 (Conv2D)	(None, 64, 64, 128)	147584
dropout_2 (Dropout)	(None, 64, 64, 128)	0
conv3 (Conv2D)	(None, 64, 64, 128)	147584
max_pooling2d_1 (MaxPooling2	(None, 8, 8, 128)	0
dropout_3 (Dropout)	(None, 8, 8, 128)	0
dense1 (Conv2D)	(None, 1, 1, 128)	1048704
dropout_4 (Dropout)	(None, 1, 1, 128)	0
dense2 (Conv2D)	(None, 1, 1, 1)	129

Total params: 1,347,585
Trainable params: 1,347,585
Non-trainable params: 0

III.7.2 Les différentes vidéos utilisées pour la détection

Afin de tester la détection des véhicules et des vitesses, les échantillons de vidéos ont été collectés à travers différents endroits. Nous avons constaté pour la première vidéo (Fig III.14) que les boites englobantes sont dessinées sur les véhicules (bonne identification des véhicules) et les vitesses sont détectées sur chaque véhicule, par contre pour la deuxième vidéo (Fig III. 15), les boites englobantes sont dessinées sur tout type d'objets : les véhicules, et même sur les panneaux, les arbres,…Etc, ainsi que leurs vitesses qui sont calculées. Ce problème persiste surtout lorsqu'il s'agit des caméras qui bougent tout le temps à cause de mal fixation, ou encore en mauvais temps et en présence de vent.

Fig III. 14.La 1er vidéo avec un camera stable

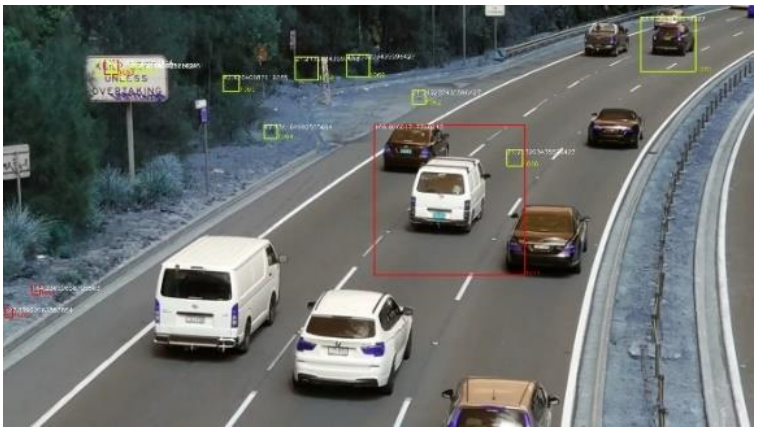

Fig III. 15.la 2ème vidéo avec un camera instable

III.7.3 Extraits du code source

Le code source de notre application commence par l'importation des paquets utilisés :

```
1: import skvideo.io
2: import cv2
3: import numpy as np
4: from collections import deque
5: from PIL import Image, ImageTk
6: from tkinter import *
7 : import tkinter.ttk
```

1 : Scikit-video est conçu pour un traitement vidéo facile en utilisant Python. Il est modélisé dans l'esprit d'autres scikits réussis tels que **scikit-learn** et **scikit-image**.

2 et **3 : OpenCV** (pour Open Computer Vision), c'est une bibliothèque graphique libre, initialement développée par Intel, spécialisée dans le traitement d'images en temps réel. En Python, le traitement d'image en utilisant **OpenCV** est implémenté en utilisant les modules **cv2** et **numpy,** pour accéder à la fonctionnalité de traitement d'image.

4 : Deque peut être implémenté en python en utilisant le module **"collections"**. Deque est préférable à la liste dans les cas où nous avons besoin d'opérations **append** et **pop** plus rapides des deux extrémités du conteneur, il fournit une complexité temporelle O (1) pour les opérations **append** et **pop** par rapport à **list** qui fournit une complexité temporelle O (n)

5 : Python Imaging Library (ou **PIL**) c'est une bibliothèque de traitement d'image pour le langage de programmation Python. Elle permet d'ouvrir, de manipuler, et de sauvegarder différents formats de fichiers graphiques.

6 et **7 : Tkinter** (Tk interface) c'est un module intégré à la bibliothèque standard de Python, bien qu'il ne soit pas maintenu directement par les développeurs de Python. Il offre un moyen de

créer des interfaces graphiques via Python. Tkinter a l'avantage d'être disponible par défaut, sans nécessiter une installation supplémentaire.

```
8: cap = skvideo.io.vread('video1.avi')
```

8 : skvideo.io.vread permet de charger n'importe quelle vidéo image par image. Cette fonction ne doit être utilisée que pour les petites vidéos.

```
9 : car_cascade = cv2.CascadeClassifier('cars.xml')
```

9 : la classe CascadeClassifier est utilisée pour détecter les objets dans un flux vidéo, et de charger un fichier de classificateur de type .XML.

```
10 : with open("out.csv", "w") as o:pass
```

10 : cette instruction permet d'écrire dans un fichier CSV. Si le fichier existe, elle l'efface et commence à y écrire à partir de la première ligne. Sinon, si le fichier n'existe pas, il sera créé et les données y sont écrites.

```
11 : centroids_list = deque([]) initialisation du deque
12 : cv2.imshow('video-original', image) afficher les images dans une fenêtre
```

```
13: gray_image = cv2.cvtColor(cap[i], cv2.COLOR_BGR2GRAY)
```

13 : Nous utilisons la fonction **cv2.cvtColor(input_image, flag)** pour la conversion des couleurs, où **flag** détermine le type de conversion.

Pour la conversion BGR → Gray, nous utilisons les flags cv2.COLOR_BGR2GRAY.

De même pour BGR → HSV, nous utilisons l'indicateur cv2.COLOR_BGR2HSV.

```
14: cars = car_cascade.detectMultiScale(gray_image, 1.1, 1)
```

14 : detectMultiScale (image, scaleFactor, minNeighbors) il s'agit d'une fonction générale de détection d'objets

Image : La première entrée est l'image en niveau de gris.

ScaleFactor : Cette fonction compense une fausse perception de taille qui se produit lorsqu'un visage semble être plus grand que l'autre simplement parce qu'il est plus proche de la caméra.

minNeighbors : Il s'agit d'un algorithme de détection qui utilise une fenêtre mobile pour détecter des objets, il le fait en définissant combien d'objets se trouvent près de l'actuel avant de pouvoir déclarer la face trouvée.

```
15: X = abs(float(centroid_data[2][0] + centroid_data[2][2]) / 2 - float(xA + xB) / 2)
16: Y = abs(float(centroid_data[2][1] + centroid_data[2][3]) / 2 - float(yA + yB) / 2)
```

15 et **16 :** Vérifier la proximité en utilisant la distance de Manhattan

```
17 : centroids_list[idx][6].append(np.sqrt(X ** 2 + Y ** 2) * 30)
```

17 : La fonction **append()** est utilisée pour insérer la valeur dans son argument à l'extrémité droite de deque.

$$Vitesse = sqrt(X ** 2 + Y ** 2) * 30 \qquad \text{(III. 4)}$$

```
18 : cv2.rectangle(image, (xA, yA), (xB, yB), (0, 255, 200), 2)
```

18 : La fonction **cv2.rectangle()** est utilisée pour dessiner un rectangle, nous avons besoin du coin supérieur gauche et du coin inférieur droit du rectangle. **cv2.rectangle (**Les paramètres ici sont l'image, la coordonnée supérieure gauche, la coordonnée inférieure droite, la couleur, l'épaisseur de ligne).

```
19 : cv2.putText(image, str(j), (xB, yB), 0, 0.5, (0, 255, 200), 1, cv2.LINE_AA)
20 : cv2.putText(image, (str(centroids_list[idx][6][-1])), (xA, (yA + 10)), 0, 0.5,(255, 255, 255), 1,
cv2.LINE_AA)
```

19et **20 :** La fonction **cv2.putText()** est utilisée pour créer un texte dans l'image **cv2, putText** (l'image sur laquelle est dessinée, le texte à écrire, coordonnées du point de départ du texte, police à utiliser, taille de police, couleur du texte, épaisseur du texte, le type de ligne utilisé)

```
21: with open("out.csv", "a") as o: o.write(str(j) + ": " +"Distance:" + " " + str(Z) +" "+"Speed:" + " " + str(centroids_list[idx][6][-1]) +  "\n")
```

21 : Cette instruction est utilisée pour écrire des lignes dans un fichier préexistant. Il est différent du mode écriture, car il n'efface pas les lignes existantes du fichier CSV, il ajoute simplement les lignes au-dessous d'elles.

$$Z = Distance = sqrt(X ** 2 + Y ** 2) \qquad \text{(III. 5)}$$

```
22: centroids_list.appendleft([count, [0], (xA, yA, xB, yB), 1, 5, "unlocked", [0], car_count])
```

22 : La fonction **appendleft()** est utilisée pour insérer la valeur dans son argument à l'extrémité gauche du deque

```
23 : if cv2.waitKey(33) == 27: break
```

23 : Attendre que la touche **Echap** arrête **cv2.waitKey ():** ceci est une fonction de liaison du clavier, qui prend un argument (x) le temps en millisecondes. La fonction retarde (x) millisecondes tout événement du clavier. Si (0) est pressé, il attend indéfiniment une frappe, si une autre touche est pressée, le programme continue.

❖ Les instructions utilisées pour l'interface graphique sont :

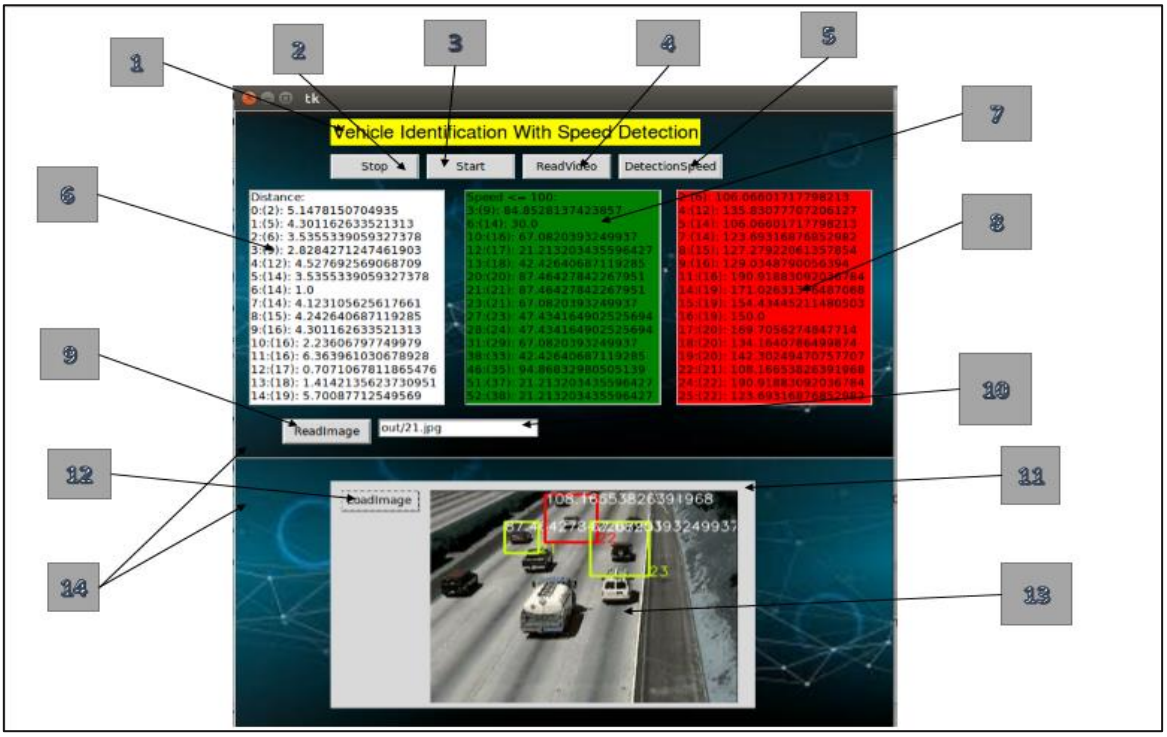

Fig III. 16.L'interface graphique de l'application

1. Titre de l'interface graphique :

```
titre = Label(app,text = "Vehicle Identification With Speed Detection", font =("Helvetica",15), foreground='Black', bg='Yellow').place(x=100,y=10)
```

2. Bouton **"stop"** : lorsqu'on clique sur ce bouton, l'exécution s'arrête

```
Button(app,text="Stop",width=8,height=1,command=stop).place(x=100, y=50)
```

3. Bouton **"Start"** : lorsqu'on clique sur ce bouton, l'exécution se lance

```
Button(app,text="Start",width=8,height=1,command=start).place(x=200, y=50)
```

4. Bouton **"ReadVideo"** : lorsqu'on clique sur ce bouton, la lecture de la vidéo commence.

```
Button(app,text="ReadVideo",width=8,height=1,command=ReadVideo).place(x=300,y=50)
```

5. Bouton **"DetectionSpeed"** : lorsqu'on clique sur ce bouton, la détection commence (Fig III.17).

```
Button(app,text="DetectionSpeed",width=10,height=1,command=DetectionSpeed).place(x=400,
y=50)
```

Fig III. 17.Vidéo originale et Détection de vitesse

6. Liste des distances de véhicules en mouvement :

```
lb2 = Listbox(app, width=25, height=16)
lb2.insert(0,"Distance:")
lb2.insert(j+1,str(j)+":"+"("+str(count)+")"+":"+" "+ str(Z))
lb2.place(x=15, y=90)
```

7. Liste (vitesse < = 100) des véhicules en mouvement :

```
lb = Listbox(app,width=25, height=16,bg='Green')
lb.insert(0,"Speed <= 100:")
lb.insert(j+1,str(j)+":"+"("+str(count)+")"+ ":"+" "+str(centroids_list[idx][6][-1]))
lb.place(x=240, y=90)
```

8. Liste (vitesse > 100) des véhicules en mouvement :

```
lb1 = Listbox(app,width=25, height=16,bg='red')
lb1.insert(0,"Speed > 100:")
lb1.insert(j+1,str(j)+":"+"("+str(count)+")"+ ":"+" "+str(centroids_list[idx][6][-1]))
lb1.place(x=460, y=90)
```

9. Bouton **"ReadImage"** : lorsqu'on clique sur ce bouton, la lecture de l'image commence.

```
Button(app, text="ReadImage", width=8, height=1, command=Rimage).place(x=50, y=350)
```

10. Le nom d'image :

```
entry_name = Entry(app)
entry_name.place(x=150, y=350)
```

11. Création un widget Frame en tant qu'enfant **« c »** d'un widget parent **« app »:**

```
c = tkinter.ttk.Frame(app, padding=(5, 5, 12, 0))
c.place(x=100, y=420)
```

12. Bouton **"LoadImage"** : lorsqu'on clique sur ce bouton, l'image s'affiche

```
btn = tkinter.ttk.Button(c, text="LoadImage", command=showImage)
btn.grid(column=0, row=0, sticky=N, pady=5, padx=5)
```

13. Afficher l'image :

```
image_tk = ImageTk.PhotoImage(Image.open(entry_name.get()))
def showImage():
   lbl['image'] = image_tk
lbl = tkinter.ttk.Label(c)
lbl.grid(column=1, row=0, sticky=N, pady=5, padx=5)
```

14. Afficher l'image de l'arrière-plan :

```
image = Image.open("im.png")
photo1 = ImageTk.PhotoImage(image)
Label(app, image=photo1).place(x=0,y=310)
photo = ImageTk.PhotoImage(image)
Label(app, image=photo).place(x=0,y=0)
```

III.8 Conclusion

Dans ce dernier chapitre, l'approche de détection des véhicules et de leurs vitesses a été présentée. Elle a été implémentée en utilisant l'une des méthodes de l'intelligence artificielle et qui est les Réseaux de Neurones Convolutionnels. Pour cela, nous avions opté pour le modèle de réseaux en couches dont son architecture a été présentée. Sur cette base, une application informatique a été développée en utilisant divers outils de programmation tels que le langage Python et les librairies OpenCV.

Conclusion Générale

L'objectif principal de ce projet de fin d'études est la reconnaissance de véhicules ainsi que la détection de leurs vitesses. Nous avions discuté des notions fondamentales des réseaux de neurones en générale et des réseaux de neurones convolutionnels en particulier. Nous avions ainsi présenté les différents types de couches utilisées dans la classification : la couche convolutionnelle, la couche de rectification, la couche de pooling et la couche fully connected.

Les paramètres du réseau sont difficiles à définir a priori. C'est pour cette raison que nous avions choisi une architecture bien déterminée afin d'obtenir de meilleurs résultats en terme de précision et de minimisation d'erreurs.

Les résultats obtenus ont montré que plus la profondeur du réseau de neurones est importante plus les résultats obtenus sont meilleurs en terme de précision et de minimisation d'erreurs.

Comme dernière étape, nous avions implémentée notre modèle neuronal via une application informatique en utilisant divers outils de programmation tels que XML, le langage Python et les librairies OpenCV.

Comme perspectives ; nous comptons :

- Réviser l'installation des caméras afin d'éliminer le problème de l'instabilité en présence du vent, les mouvements indésirables, la grêle, pluies orageuses ; ce qui empêche la précisions des captures
- Améliorer l'algorithme qui permet de capturer des vidéos en obscurité (durant la nuit ; dans les tunnels, …) pour identification des véhicules et leurs vitesses avec le changement des conditions d'éclairage.
- La détection des plaques de véhicules, afin d'ajouter plus de paramètres sur ces derniers pour séparer les véhicules de types différents ayant la même largeur et la même longueur.
- Améliorer l'architecture du RNA en augmentant sa profondeur ; en ajoutant plus de couches cachées afin d'avoir de meilleurs résultats en terme de précision et diminution d'erreurs
- Equiper les caméras d'un système d'alimentation de secours (panneaux solaires, batteries rechargeables ; …) pour prévoir le problème de coupure de courant
- Implémenter notre application sur des équipements mobiles tels que les smartphones

Bibliographie

[1] D. G. J. J. &. H. C. Bullock, «A neural network for image-based vehicle detection,» Transportation Research Part C: Emerging Technologies, vol. 1, n° 13, pp. 235-247, 1993.

[2] C. e. C. F. OZKURT, «Automatic traffic density estimation and vehicle classification for traffic surveillance systems using neural networks,» Mathematical and Computational Applications, vol. 14, n°13 , pp. 187-196, 2009.

[3] M. Rouse, «AI (artificial intelligence),» Décembre 2016. [En ligne]. Available: https://searchenterpriseai.techtarget.com/definition/AI-Artificial-Intelligence. [Accès le 15 Mai 2018].

[4] J.-P. e. H. M.-C. HATON, Artist, L'intelligence artificielle. [Art]. Presses universitaires de France , 1989.

[5] A.-T. N. Axel Mounier, «TPE - Intelligence Artificielle,» 2013. [En ligne]. Available: http://intelart.e-monsite.com/pages/ii-l-utilisation-de-l-ia/domaines-d-applications.html. [Accès le Mai 2018].

[6] «Intelligence artificielle . fr,» 2012. [En ligne]. Available: http://www.intelligenceartificielle.fr/domaines_IA.php. [Accès le Mai 2018].

[7] B. Alexandra, «Comment l'intelligence artificielle s'est-elle développée et jusqu'où va-t-elle progresser ?,» 2016. [En ligne]. Available: http://www.tpeia.sitew.fr/Avantages_Inconvenients.B.htm#Avantages_Inconvenients.B. [Accès le avril 2018].

[8] B. K. Reddy, «Advantages and Disadvantages of Artificial Intelligence,» 2016. [En ligne]. Available: https://content.wisestep.com/advantages-disadvantages-artificial-intelligence/. [Accès le 13 Mai 2018].

[9] F. HACHEMI, La détection et suivi des objets en mouvement dans une scène vidéo en utilisant la bibliothèque OpenCV, Mémoire de fin d'études: Université Abou Bakr Belkaid– Tlemcen, 2016.

[10] K.Bayram, Les réseaux de neurones, Paris: Département INFRES ENST, 2004-2015.

[11] D. e. a. I.Bloch, Le traitement des images (tome 1), Paris: Département TSI - Télécom, 2005.

[12] M. BERGOUNIOUX, Quelques Méthodes de Filtrage en Traitement d'image, école CIMPA, 2011.

[13] Y. V. D. R. E. &. D. J. F. Wang, «Tracking moving objects in video sequences,» In Conference on Information Sciences and Systems, vol.2, pp. 24-29, 2000, March.

[14] M. E. D. J. A. H. E. D. Fatiha, Détection et Suivi d'Objets en Mouvement Dans Une Séquence d'Images, Mémoire de Magiste: Université Mohamed Boudiaf des sciences et de la technologie d'Oran)., 2012.

[15] Sïana, «L'intelligence Artificielle – TPE,» 2015. [En ligne]. Available: https://iatpe2015.wordpress.com/le-fonctionnement/le-reseau-de-neurones-artificiel/les-neurones-biologiques-et-artificiels/. [Accès le Avril 2018].

[16] N. SALHI, «Surveillance et diagnostic d'une chaîne de production par les réseaux de neurones artificiels,» chez Mémoire de Magister, Université de Boumerdes, 2008.

[17] T. Thelliez, «Cours, Intelligence artificielle, Technology,» 5 Mars 2018. [En ligne]. Available: https://rocketbootstrapper.com/fr/reseaux-de-neurones-artificielles/. [Accès le Avril 2018].

[18] M. CLERGUE, «Réseaux de neurones artificiels,» 2013.

[19] G. e. a. Dreyfus, «Réseaux de neurones-Méthodologie et applications,» 2002.

[20] M. PARIZEAU, Artist, Réseaux de neurones. [Art]. GIF-21140 et GIF-64326 2004, vol. 124., 2004.

[21] C. TOUZET, Les réseaux de neurones artificiels, introduction au connexionnisme, Collection de l'EERIE, 1992.

[22] M. PARIZEAU, Le perceptron multicouche et son algorithme de rétropropagation des erreurs, département de génie électrique et de génie informatique, Université de laval, 2004.

[23] K. Djamila, «Utilisation des réseaux de neurones comme outil du datamining : Génération de modèle comportemental d'un processusphysique à partir de données,» chez MEMOIRE DE MASTER, UNIVERSITE ABOU-BAKR BELKAID DE TLEMCEN, 2012.

[24] D. Team, «Machine Learning Tutorials,» 25 Juillet 2017. [En ligne]. Available: https://data-flair.training/blogs/learning-rules-in-neural-network. [Accès le Mai 2018].

[25] «mathworks,» What Is Deep Learning?, 2018. [En ligne]. Available: https://www.mathworks.com/discovery/deep-learning.html. [Accès le Avril 2018].

[26] Y. B. Y. e. H. G. LECUN, «Deep learning,» nature, vol. 521, n°17553, p. 436, 2015.

[27] F. CHABOT, Analyse fine 2D/3D de véhicules par réseaux de neurones profonds, Thèse de doctorat: Université Clermont Auvergne, 2017.

[28] J. &. G. A. Patterson, Deep Learning: A Practitioner's Approach, O'Reilly Media, Inc., 2017.

[29] S. SHARMA, «Activation Functions: Neural Networks,» 6 Septembre 2017. [En ligne]. Available: https://towardsdatascience.com/activation-functions-neural-networks 1cbd9f8d91d6. [Accès le Mai 2018].

[30] J. Yang, «ReLU and Softmax Activation Functions,» 11 Février 2017. [En ligne]. Available: https://github.com/Kulbear/deep-learning-nano-foundation/wiki/ReLU-and-Softmax Activatio-Functions. [Accès le Mai 2018].

[31] N. H. G. K. A. e. a. SRIVASTAVA, «Dropout: A simple way to prevent neural networks from overfitting,» The Journal of Machine Learning Research, vol. 15, n°11, pp. 1929-1958, 2014.

[32] «Programmation Python/Introduction,» 17 Janvier 2017. [En ligne]. Available: https://fr.wikibooks.org/wiki/Programmation_Python/Introduction. [Accès le Juin 2018].