République Algérienne démocratique et populaire
Ministère de l'enseignement supérieur et de la recherche scientifique
Université Saad Dahleb de Blida

Faculté des Sciences
Département de l'Informatique

Mémoire Présenté par :

Cheriet Sami M'hamed
&
Benabdelouahab Kussayla

En vue d'obtenir le diplôme de Master

Domaine : Mathématique et informatique

Filière : Informatique

Spécialité : TAL et SIR

Sujet : **Conception et réalisation d'un système de reconnaissance de plaque d'immatriculation en temps réel**

Encadré par : Mr. Abderrahmane Namane

Année Universitaire 2020-2021

Table des matières

Table de figures

Introduction Générale

Durant les dernières années le marché des ventes automobiles a connu une explosion en croissance ce qui submergé rendu le Traffic routier très important et presque impossible dans les heures de pointe, imaginez le flux important de véhicules à l'entrée des différents établissements (sociétés, entreprises, écoles, universités, hôpitaux, …), et si à chaque entrée on avait un agent qui doit prendre en note toute les plaques d'immatriculation, en prenant compte du risque d'erreurs où les conséquences vont être importantes dans la gestion des entrées : longue file d'attente, retard des employés, retard des travaux … ce qui implique le besoin d'automatiser l'accès de ces différents établissement (avec par exemple une barrière contrôlé par une caméra), ce qui permet un accès rapide et une fluidité dans les entrées.

D'où l'intérêt de notre projet intitulé (Reconnaissance de plaque d'immatriculation en temps réel). Ce sujet présente une grande importance et intérêt dans le contrôle des accès aux établissements ainsi que leurs sécurités.

Dans ce travail nous présentons un programme de reconnaissance des plaques d'immatriculation en temps réel. Pour aboutir à des résultats pertinents ce mémoire est établi sur trois chapitres :

- **Le premier chapitre** : nous présentons les généralités sur l'ANPR et l'analyse d'images.

- **Le deuxième chapitre :** est consacré au prétraitement, l'extraction des caractéristiques et les réseaux de neurones

- **Le troisième chapitre :** est consacré à l'implémentation et les résultats de notre travail.

1 Chapitre 1 : Généralités sur l’analyse d’images

1.1 Introduction :

Depuis le début du 20ème siècle, l'homme essaie de créer une machine capable de reproduire les taches qu'une personne effectue au quotidien, cependant la majorité de ces taches requirent la vision, qui est une faculté innée chez l'humain mais qui à tant fasciner les chercheurs, ce qui est donnée naissance à la vision par ordinateur.

Cette technologie ne cesse de se développer et a depuis touché plusieurs domaines, par exemple pour aider les humains dans les taches d'identification et de reconnaissance comme la reconnaissance automatique de plaques d'immatriculation qui avec la nette ascension des ventes automobiles est devenue un besoin fondamental, notre travail présente un système ANPR qui se base sur les techniques de reconnaissance de formes combiné avec la reconnaissance optique des caractères.

Dans ce chapitre nous présentons brièvement des généralités sur la reconnaissance automatique des plaques d'immatriculation, on couvrira aussi les images et le traitement d'images, la reconnaissance des formes ainsi que le système de reconnaissance optique des caractères.

1.2 Reconnaissance automatique de plaques d'immatriculation :

1.2.1 Historique :

L'ANPR a été inventé en 1976 à la Police Scientific Développent Branch en Grande-Bretagne.[5] Les systèmes prototypes fonctionnaient en 1979 et des contrats ont été attribués pour produire des systèmes industriels, d'abord chez EMI Electronics, puis chez Computer Recognition Systems (CRS, qui fait maintenant partie de Jenoptik) à Wokingham, au Royaume-Uni. Les premiers systèmes d'essai ont été déployés sur la route A1 et au tunnel de Dartford. La première arrestation par détection d'une voiture volée a eu

lieu en 1981.[6] Cependant, l'ANPR n'est pas devenu largement utilisé jusqu'à ce que de nouveaux développements dans des logiciels moins chers et plus faciles à utiliser aient été lancés au cours des années 1990. La collecte de données ANPR pour une utilisation future (c'est-à-dire pour résoudre des crimes alors non identifiés) a été documentée au début des années 2000.Le premier cas documenté d'ANPR utilisé pour aider à résoudre un meurtre s'est produit en novembre 2005, à Bradford, au Royaume-Uni, où l'ANPR a joué un rôle essentiel dans la localisation et la condamnation des assassins de Sharon Beshenivsky.[1]

1.2.2 Utilité et domaine d'applications de l'ANPR :

La reconnaissance automatique de plaques d'immatriculation est une technologie qui utilise un traitement d'image avancé et une reconnaissance optique de caractères sur les flux vidéo ou images pour lire la plaque minéralogique du véhicule cette technologie est maintenant d'une grande utilité dans plusieurs secteurs et domaines parmi eux :

- **Forces de l'ordre :** Prévention de la criminalité à l'aide de caméras ANPR installées sur les voitures de police mobiles et stationnaire.
- **Les routes à péages :** Capture la plaque d'immatriculation, le péage en vidéo et une facturation efficace avec presque aucune intrusion humaine.
- **Identification des infractions au code de la route :** Identifie les ruptures de signal, les accidents, les mauvais virages et les sorties de voie avec un système de reconnaissance de marque et de modèle de voiture
- **Surveillance des aires de stationnement :** Aide à la gestion efficace de l'occupation du stationnement et à l'identification des places de stationnement vacantes, du temps de stationnement, etc.
- **Smart parking :** Idéal pour les commerces de détail, les hôtels, les hôpitaux et toute entreprise qui cherche à offrir une expérience de stationnement libre. Ce qui fera l'objet de notre travail.

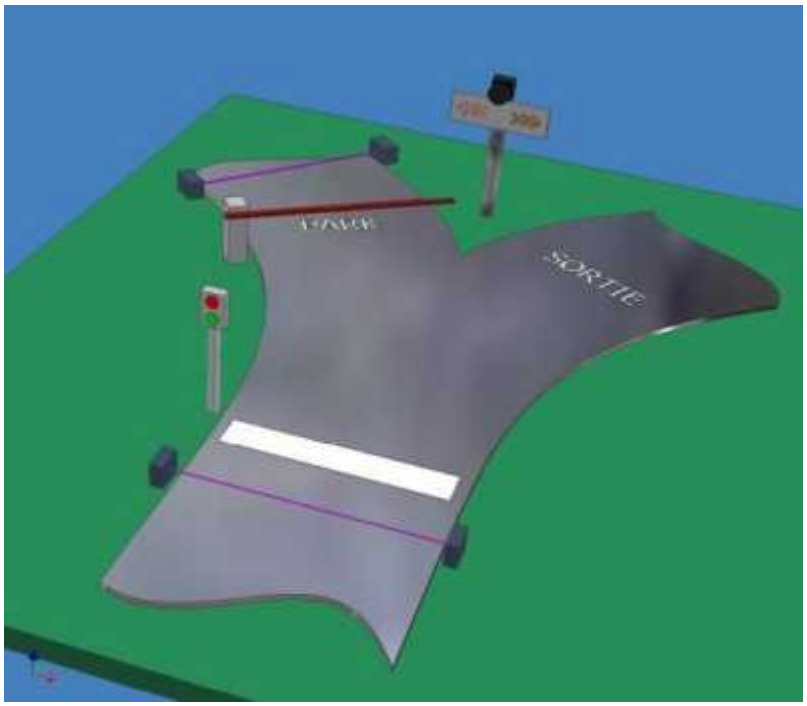


Figure 1.1 : Schéma d'un système d'accès automatique.

Figure 1.2 : Automatisation d'accès

1.2.3 Définition d’une plaque d’immatriculation (plaque minéralogique) :

Une plaque d'immatriculation est une plaque portant une combinaison unique de chiffres ou de lettres (pour une zone géographique donnée), destinée à identifier facilement un véhicule.

1.2.3.1 Système d’immatriculation algérien :

Le numéro d'immatriculation est composé de 3 groupes de chiffres séparés par un espace (figure 1.1) :

- **1er groupe :** 5 chiffres, 6 chiffres au maximum, qui correspond au numéro de dossier du véhicule.
- **2eme groupe :** composé de 3 chiffres : le premier chiffre permet d'identifier le type de véhicule (1 véhicule classique, 2 pour un poids lourd, 3 voiture commerciale,4 bus, 5 tracteurs de semi-remorque ,6 tracteur, 7 engins, 8 remorques, 9 motos), les deux suivants renvoient à l'année de mise en circulation du véhicule.
- **3eme groupe :** 2 chiffres qui identifient la wilaya d'immatriculation (De 01 à 58 actuellement).

Figure 1.3 : exemple de plaque d’immatriculation algérienne

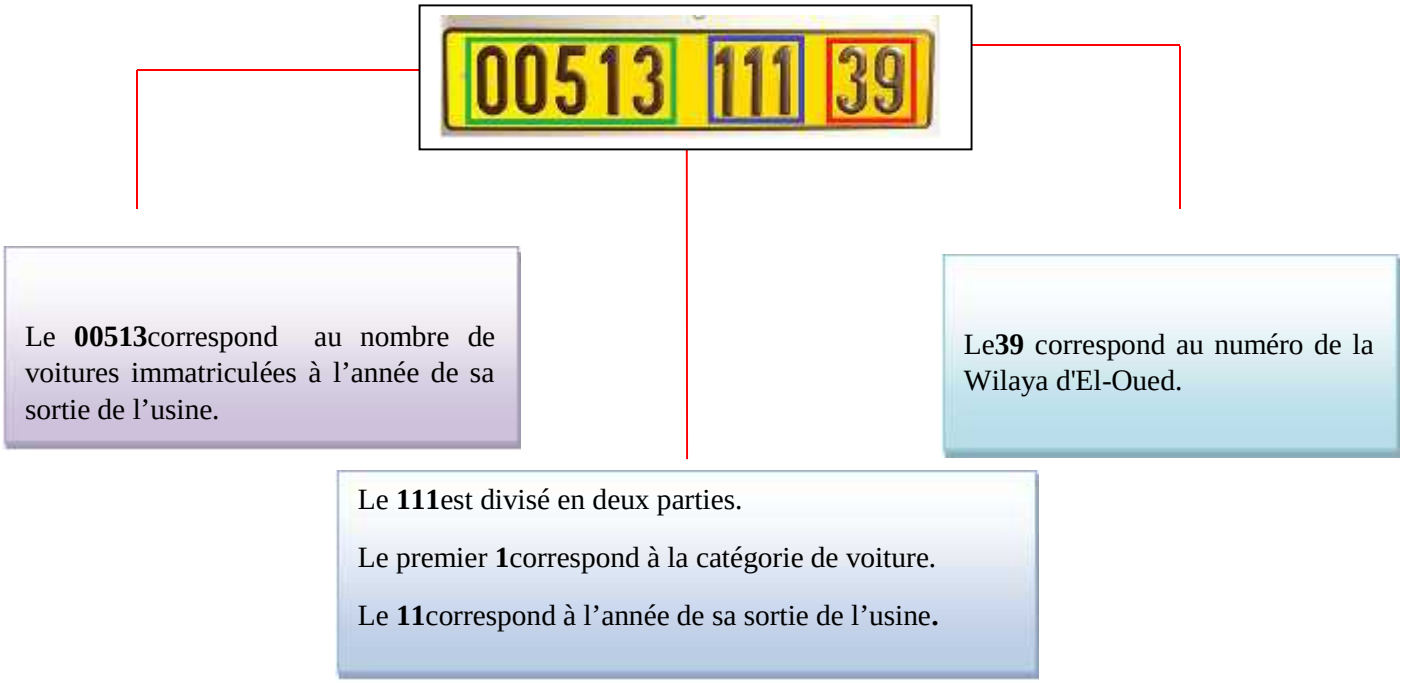


Figure 1.4 : Présentation d'une plaque d'immatriculation.

1.3 Image et traitement d'image :

Au cœur de la l'ANPR se trouve son entrée, qui est appelée une image. Les sources d'images comprennent les caméras, les enregistreurs vidéo, les scanners et images microscopiques, entre autres. Dans les prochaines rubriques nous allons parler de ce que sont les images, comment les acquérir, comment sont-ils stockés et représentés, Quelles sont les types d'image et quelles sont ces caractéristiques etc…

1.3.1 Définition d'image :

La définition du terme « image » lui-même, telle qu'elle est donnée par exemple par le Petit Robert, englobe une multitude de significations distinctes. Cela va de la «

reproduction exacte ou représentation analogique d'un être, d'une chose », à la « représentation mentale d'origine sensible » ou à des concepts plus physiques comme un « ensemble des points » où convergent des rayons lumineux (cas des images optiques).[7]

1.3.2 Acquisition d'image

L'acquisition permettant la conversion du document papier sous la forme d'une image numérique (bitmap). Cette étape est importante car elle se préoccupe de la préparation des documents à saisir, du choix et du paramétrage du matériel de saisie (scanner), ainsi que du format de stockage des images.

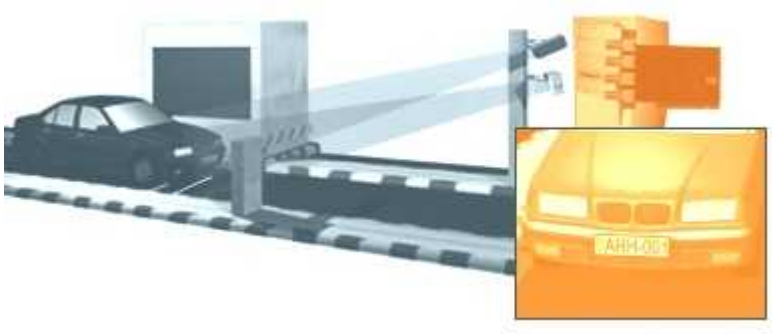


Figure 1.5 : Schéma de dispositif d'acquisition

1.3.3 Image numérique

Dans son sens le plus général, toute image qui a été acquise, traitée et sauvegardée sous une forme codée représentable par des nombres (valeurs numériques).

La numérisation est le processus qui permet de passer de l'état d'image physique (image optique par exemple) qui est caractérisée par l'aspect continu du signal qu'elle représente (une infinité de valeur dans l'intensité lumineuse par exemple), à l'état d'image numérique qui est caractérisée par l'aspect discret (l'intensité lumineuse ne peut prendre que des valeurs quantifiées en un nombre fini de points distincts).[7]

C’est cette forme numérique qui permet une exploitation ultérieure par des outils logiciels sur ordinateur.

1.3.4 Types d’images

On distingue deux types d’images numériques :

1.3.4.1 Images matricielles :

Elle est composée d’une matrice (tableau) de points à plusieurs dimensions, chaque dimension représentant une dimension spatiale (hauteur, largeur, profondeur) ou autre (par exemple, un niveau de résolution). Dans le cas d’images 2D les points sont appelés pixels qui sont représentés sous forme de minuscules carrés, et derrière chaque pixel se trouve un échantillon de couleur. [8]

1.3.4.2 Image vectorielle :

Le principe est de représenter les données de l'image par des formules géométriques qui vont pouvoir être décrites d'un point de vue mathématique. Cela signifie qu'au lieu de mémoriser une mosaïque de points élémentaires, on stocke la succession d'opérations conduisant au tracé. Grâce à la vectorisation, chaque élément a une place bien définie ce qui empêche la déformation et la perte de qualité de l’image.[8]

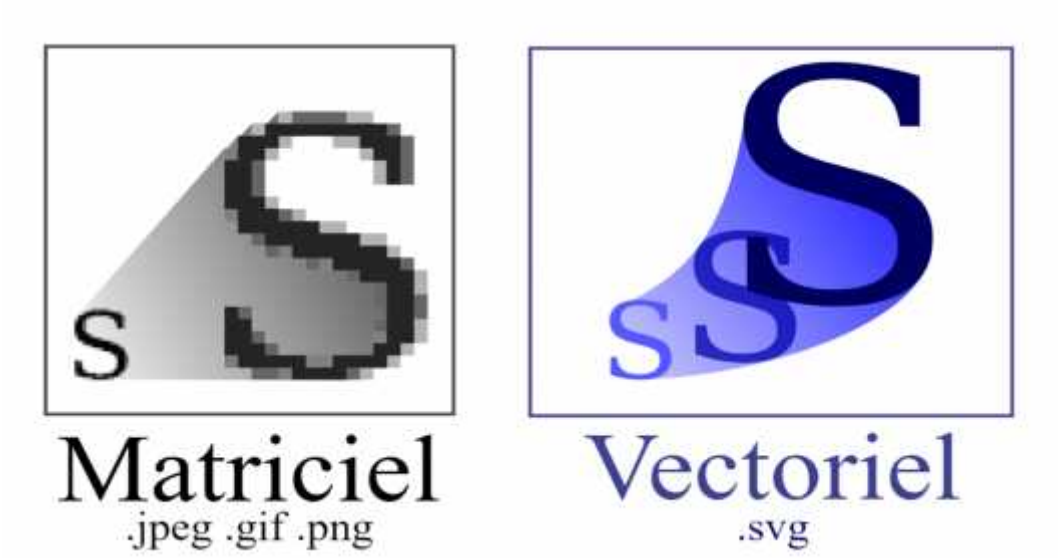


Figure 1.6: Image Matriciel vs Image Vectoriel

1.3.5 Caractéristiques d'une image numérique

1.3.5.1 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, le mot luminance est substitué au mot brillance, qui correspond à l'éclat d'un objet.[9] 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.

1.3.6 Codage des couleurs de l'image numérique

Le pixel est l'unité de base permettant de mesurer la définition d'une image numérique matricielle, une image dépend du codage des couleurs du pixel elle peut donc être :

a. Image binaire (noir et blanc):

Dans cette représentation, un pixel de cette image n'a la possibilité d'avoir que deux valeurs distinctes 0 pour le noir et 1 pour le blanc.

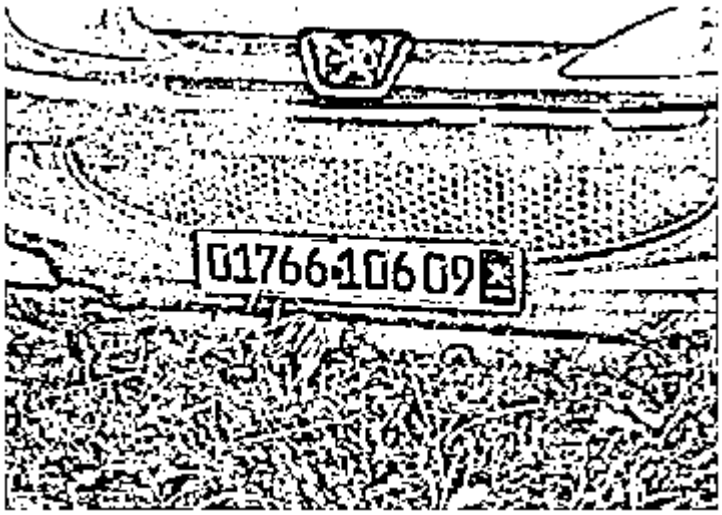


Figure 1.7 : Exemple image binaire (0 et 1)

b. Image en niveau de gris :

Dans une image numérique, le niveau de gris représente la luminosité d'un pixel. Un pixel est codé généralement sur un octet et a donc 256 valeurs possibles ; il permet d'obtenir plus de nuances que le simple noir et blanc.

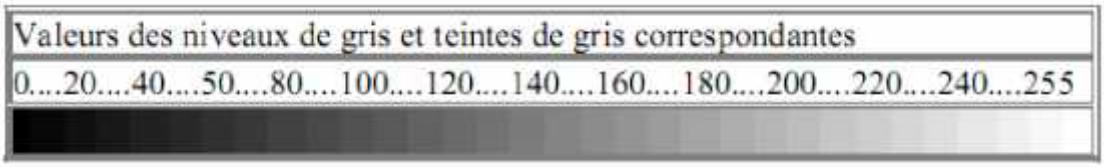


Figure 1.8 : Différents niveaux de gris.

Figure 1.9: Exemple image en niveau de gris

c. Image en couleurs :

Il existe plusieurs modes de codages informatiques des couleurs, le plus utilisé pour le remaniement des images est l'espace colorimétrique rouge, vert, bleu (RVB ou en anglais RGB Red, Green, Blue). Cet espace est basé sur la synthèse additive des couleurs, les couleurs sont obtenues en mélangeant les trois couleurs fondamentales, 18 à chaque couleur est associé un octet et chaque pixel « couleur » est codé sur 3 octets pour former 16 millions de couleurs différentes.

Figure 1.10: Image en couleur

1.3.7 Traitement d'images numérique :

Le traitement d'images est l'ensemble des opérations effectuées sur l'image, afin d'en améliorer la lisibilité et d'en faciliter l'interprétation. C'est, par exemple, le cas des opérations de rehaussement de contraste, élimination du bruit et correction d'un flou. C'est aussi l'ensemble d'opérations effectuées pour extraire des "informations" de l'image comme la segmentation et l'extraction de contours. Avant le traitement d'images, on peut aussi effectuer des opérations de prétraitement qui sont toutes les techniques visant à améliorer la qualité d'une image. De ce fait, la donnée de départ est l'image initiale et le résultat est également une image,

1.3.7.1 Filtrage d'images :

Le principe du filtrage est de modifier la valeur des pixels d'une image, généralement dans le but d'améliorer son aspect. En pratique, il s'agit de créer une nouvelle

image en se servant des valeurs des pixels de l'image d'origine. Deux types de filtrages peuvent être appliqués au traitement d'images : filtrage linéaire et filtrage non linéaire.

a. Le filtrage linéaire :

Consiste simplement à remplacer chaque niveau de gris par une combinaison linéaire des niveaux de gris des pixels voisins. Les filtres linéaires sont caractérisés par leur réponse pulsionnelle (figure. 1.15).

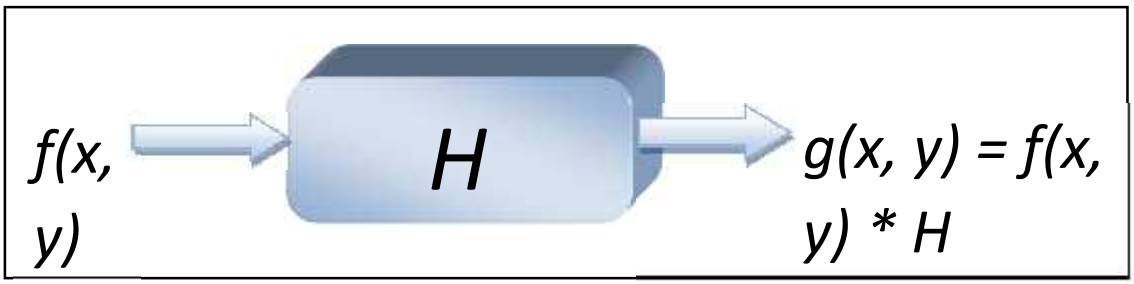


Figure 1.11 : représentation d'un système linéaire.

f: représente l'image d'entrée.

g: représente l'image de sortie

H : représente le masque.

Remarque Les images d'entrée et de sortie sont de même taille N*N, et le masque à une taille (2*n+1). (2*n+1). Parmi les filtres linéaires on a : le filtre moyen et le filtre gaussien.

Filtrage gaussien :

Parmi les plus courants filtres linéaires, le filtre gaussien représente une bonne qualité de résultats ainsi qu'une facilité de mise en œuvre. Ce filtre permet d'éliminer les bruits dit gaussien. Ce type de filtre utilise la moyenne pondérée des voisins (Figure 1.16).

Figure 1.16.a : image d'origine

Figure 1.16.b : filtrage Gaussien avec un masque (5×5).

Figure 1.12: Le filtre gaussien.

b. FILTRAGE NON LINEAIRE

Le filtre non linéaire a été développé pour pallier aux insuffisances des filtres linéaires : principalement la mauvaise conservation des contours. Dans ce filtre non linéaire les pixels voisins interviennent suivant une loi non linéaire (Exemple : filtrage médian).

Filtrage médian

Le filtre médian consiste à remplacer un pixel par la valeur médiane de la fenêtre d'analyse centrée sur le pixel. La taille du masque dépend de la variance du bruit et de la taille des détails significatifs de l'image traitée. Ce filtre est très utilisé pour éliminer les bruits impulsifs sur une image qui peuvent être de différentes origines (poussières, petits nuages etc..) et qui se traduisent par des taches de faible dimension dont la distribution sur l'image est aléatoire. L'avantage de ce filtre réside dans le fait qu'il garde la netteté de l'image (Figure 1.17).

Figure 1.13.a : image d'origine

Figure 1.13.b : filtrage médian avec un masque (5×5).

Figure 1.13 : exemple de filtre médian sur une image.

1.3.7.2 Binarisation (seuillage) :

La binarisation permet de passer d'une image de niveaux de gris à une image binaire composée de deux niveaux de gris 0 et 255 qui correspond respectivement au noir et blanc, plus simple à traiter. En générale, on utilise un seuil de binarisation approprié qui traduit la limite des contrastes fort et faible dans l'image. Alors la binarisation permet de distinguer un objet en contraste sur un fond. Les niveaux de gris sont partitionnés en deux classes (noire et blanche), ou le noir représente la forme, et le blanc représente le fond. L'opération de seuillage (binarisation) consiste à comparer l'intensité de tous les pixels avec une valeur de référence « seuil ». Il existe deux types de binarisation : binarisation globale et binarisation locale.

a. Binarisation globale (moyenne) :

Le seuillage global consiste à prendre un seuil, ajustable, mais identique pour toute l'image. Pour une échelle à 256 niveaux de gris, cette valeur est comprise entre 0 et 255, généralement dans la plage médiane. Chaque pixel est comparé à ce seuil : en polarité directe, ceux de niveaux de gris inférieurs sont mis à « blanc » (0), ceux supérieurs mis à « noir ».

Image d'entrée **image binaire**

Figure 1.14 : Binarisation globale.

b. Binarisation locale :

Pour la binarisation locale, l'affectation d'un pixel dépend non seulement du pixel lui-même, mais aussi de ses pixels voisins (locaux). Bien que cette méthode soit robuste, elle a l'inconvénient d'être lente, car la binarisation de chaque pixel nécessite l'analyse de son voisinage.

Soit f (i, j) représente le pixel central pour un voisinage de (2n+1) (2n+1).

$$\mu = \frac{1}{(2n+1)^2} * \sum_{k=-n}^{n} \sum_{l=-n}^{n} f(i+k, j+l)$$

$$\sigma = \sqrt{\frac{1}{(2n+1)^2} * \sum_{k=-n}^{n} \sum_{l=-n}^{n} |f(i+k, j+l) - \mu|^2}$$

Pour chaque pixel de l'image est comparé en polarité directe au seuil suivant :

$$S=\mu-k*\sigma.$$

Avec :

μ : la moyenne du voisinage (2n+1) (2n+1) du pixel traité.

σ : l'écart type du voisinage (2n+1) (2n+1) du pixel traité.

k : coefficient de binarisation ($0 \leq k \leq 1$).

Figure 1.15 : Binarisation locale.

Figure 1.16 : La binarisation locale de la figure 1.17.a.

1.3.7.3 Normalisation de caractères :

La normalisation est un processus d'expansion ou de contraction de tous les points d'image, c'est une opération qui consiste à rendre l'image en une taille standard ou fixe. Mais si la taille fixée est très petite, on peut perdre l'information, si elle est très grande, l'étape de reconnaissance va opérer lentement [12], donc une bonne normalisation va permettre d'augmenter la rapidité de traitement du système et réduire considérablement le temps de reconnaissance sans perdre les informations contenues dans l'image. La normalisation dans notre cas consiste à prendre la même taille (40*40) pour toutes les images qui avaient probablement des tailles différentes dues au découpage.

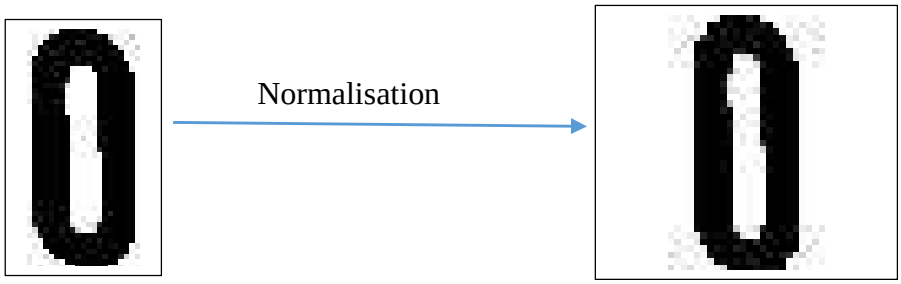


Figure 1.17 : normalisation de caractère 0 de taille 21x45p à 40x40p

1.4 Reconnaissance des formes :

1.4.1 Introduction

La reconnaissance des formes (RDF) est un domaine d'application de l'intelligence artificiel qui s'intéresse au traitement des différentes formes qui existent dans notre univers autrement dit ; la RDF est un domaine qui cherche des outils logiciels qui sont capables d'interpréter des signaux captés dans le monde physique donc, elle rentre dans le cadre de la communication homme machine.

Cette tâche est souvent difficile pour plusieurs raisons ; les formes appartiennent à un monde physique dont la transcription numérique est très complexe à cause de l'absence de capteurs adaptés à toutes les situations. Ensuite, la nature des formes et leur

apparence varient d'un échantillon à l'autre (même au sein d'une même famille), ce qui multiplie les dimensions de l'espace de représentation et rallonge le temps de décision.

1.4.2 Buts de la RDF :

La reconnaissance des formes consiste en une automatisation de tâches de perception artificielle réalisées usuellement par le cerveau et le système sensoriel humain, par exemple ; reconnaitre un caractère manuscrit, un son, un signal et un objet dans une image numérique. Une forme est une représentation simplifiée du monde extérieur définie sous une forme acceptable par l'ordinateur (par exemple un vecteur de réels, ou bien un mot d'un langage donné, …).

1.4.3 Domaine d'application

1) Reconnaissance des formes sur signaux temporels

❖ Signal de parole :

- Reconnaissance de la parole (qu'à t on dit ?).
- Reconnaissance du locuteur (qui a parlé ?).

❖ Signaux biomédicaux :

- Exemple : électrocardiogramme...
- Surveillance d'instruments, diagnostics de panne.

2) Reconnaissance des formes dans les images numériques

❖ Lecture automatique de caractères.

❖ Reconnaissance d'empreintes digitales.

- Radiographies.

- Analyse de scènes, interprétation d'images.

Dans ces dernières approches on voit rapidement apparaître la nécessité de coupler les techniques « pures et dures » de la reconnaissance des formes classiques avec les techniques de l'intelligence artificielle...

1.4.4 Les méthodes de la reconnaissance des formes

1.4.4.1 Méthodes statistiques

On considère ici la situation où il existe un ensemble de réalisations étiquetées permettant un apprentissage supervisé (par opposition aux méthodes qui partent d'une connaissance nulle sur les étiquettes des réalisations et cherchent à les grouper sur des critères de ressemblance). Les exemplaires des classes correspondent aux observations d'une variable aléatoire X. Chaque réalisation x est représentée par un vecteur de Rn. Chaque composante du vecteur correspond à un descripteur.

L'objectif est toujours d'assigner une réalisation inconnue à sa classe d'appartenance en minimisant l'erreur de décision. Ce problème peut être résolu de nombreuses manières, le choix de la méthode dépend en partie des connaissances a priori que l'on a sur les distributions de probabilités des exemplaires des classes.

1.4.4.2 Méthodes structurelles (ou syntaxiques)

Les méthodes structurelles ou syntaxiques reposent sur l'utilisation de structures de données symboliques (chaînes, arbres, graphes) pour la représentation de formes, permettant ainsi une description explicite de la façon dont la forme est construite à partir de nombreux composants inters reliés.

Exemple : reconnaissance de chromosomes par analyse de contour (partie convexe de forte courbure, partie concave de forte courbure, segment de droite).

1.4.4.3 Les méthodes connexionnistes (les réseaux de neurones)

Les méthodes connexionnistes constituent un domaine de recherche à part entière mais elles se sont imposées en reconnaissance de formes. Elles y trouvent matière à développer et à valider de multiples variantes de réseaux. Elles permettent d'étendre les techniques de séparation linéaire (le cas du perceptron) à des séparations non-linéaires.

L'idée des réseaux connexionnistes a été inspirée par la structure neurophysiologique du cerveau. Un neurone formel est une unité d'un système organisé en réseau par des connexions entre unités.

Plusieurs études considèrent que les réseaux de neurones font partie des approches statistiques. L'approche connexionniste est une implémentation dérivée de l'approche statique.

1.4.4.4 Erreurs de reconnaissance

Les défauts sur les caractères engendrent pour les logiciels de la ROC des erreurs de reconnaissance. L'évaluation de la qualité de ces logiciels nécessite de caractériser puis d'évaluer quantitativement les erreurs.

1.5 Reconnaissance optique des caractères (OCR) :

1.5.1 Historique :

En 1950, Frank Rowlett, qui avait cassé le code diplomatique japonais PURPLE, demanda à David Shepard, un cryptanalyste de l'AFSA (prédécesseur de la NSA américaine), de travailler avec Louis Tordella pour faire à l'agence des propositions de procédures d'automatisation des données. La question incluait le problème de la

conversion de messages imprimés en langage machine pour le traitement informatique. Shepard décida qu'il devait être possible de construire une machine pour le faire, et, avec l'aide de Harvey Cook, un ami, construisit « Gismo » dans son grenier pendant ses soirées et ses week-ends. Le fait fut rapporté dans le Washington Daily News du 27 avril 1951 et dans le New York Times du 26 décembre 1953 après le dépôt du brevet numéro 2 663 758. Shepard fonda alors Intelligent Machines Research Corporation (IMR), qui livra les premiers systèmes de ROC au monde exploité par des sociétés privées. Depuis 1965, la Poste des États-Unis utilise pour trier le courrier des machines ROC dont le principe de fonctionnement a été imaginé par Jacob Rabinow, un inventeur prolifique. La Poste canadienne utilise des systèmes ROC depuis 1971. Les systèmes ROC lisent le nom et l'adresse du destinataire au premier centre de tri automatisé, et impriment sur l'enveloppe un code-barres fondé sur le code postal. Les lettres n'ont plus qu'à être triées dans les centres suivants par des trieuses moins coûteuses qui n'ont qu'à lire le code-barres. Pour éviter toute interférence avec l'adresse lisible qui peut se trouver n'importe où sur la lettre, une encre spéciale est utilisée, qui est clairement visible sous une lumière UV. Cette encre semble orange dans des conditions d'éclairage normales.[10]

1.5.2 Introduction :

La reconnaissance optique de caractères (ROC), en anglais Optical Character Recognition (OCR),Le principe de cette technologie est en effet, de lire un document (une image) et détecter les formes présentes dans ce dernier pour les comparer ensuite à une bibliothèque intégrée contenant des milliers d’échantillons possibles pour chaque forme, cette comparaison vise à faire correspondre chaque forme détectée à une forme de la bibliothèque pour en fournir le caractère alphanumérique correspondant.

Dorénavant les nouveaux systèmes d’OCR avancées sont devenus plus intelligent et peuvent même reconnaitre la plupart des polices avec un haut niveau de précision sans avoir besoin d’un apprentissage au préalable.

1.5.3 Définition :

La reconnaissance optique de caractères est la reconnaissance de caractères spécifiques à une langue par un ordinateur en analysant une image, qui est déjà lisible par ordinateur. Cela se fait souvent en prenant d'abord une image du document en le scannant ou en prenant une photo numérique. Cela crée une image matricielle composée de données que l'ordinateur comprend, et grâce à des algorithmes spécifiquement programmés, dont la plupart sont utilisés dans le domaine de l'intelligence artificielle, l'ordinateur reconnaît les motifs(pattern) dans l'image, et dans ce cas, les motifs sont des caractères. Le programme crée ou sort ensuite des codes de caractères, généralement ASCII, qui sont équivalents aux caractères reconnus de l'image d'entrée. La plupart des programmes d'OCR doivent être formés afin de mieux reconnaître les caractères.[11]

1.5.4 Domaines d’application OCR :

Les outils d’OCR ont été développés en une gamme d’applications spécifiques au domaine, notamment la reconnaissance de reçu, de facture, de chèques, de documents légaux, etc. parmi les domaines les plus utilisés :

1.5.4.1 Domaine bancaire et assurances :

Ces 3 secteurs, de par leur nature, sont tous de grands consommateurs de l’OCR. L’utilisation la plus courante de l’OCR est la saine gestion des chèques :

Ses détails sont transformés en texte numérique ;

La signature est validée ;

Le chèque est approuvé en temps réel.

Le tout sans implication humaine.

1.5.4.2 Monde légal :

Peu d'industries génèrent autant de paperasse que l'industrie juridique, donc il est simple de comprendre les avantages de l'OCR ici.

La numérisation, le stockage, la conservation en base de données accessible à la recherche sont désormais possibles pour tous les documents imprimés : affidavits, jugements, déclarations, avis, testaments, etc.

L'OCR est également disponible pour des documents en chinois, en arabe et en orthographes pour les langues ayant une autre écriture que celles de type « romaine ».

1.5.4.3 Santé :

Une autre industrie qui se prête bien à l'OCR est la santé. Il est possible de numériser tout l'historique médical d'un patient : rapports de santé, radiographies, historique de maladies, suivi des traitements, diagnostics, dossiers hospitaliers, couverture d'assurance, paiements. Après numérisation, toutes ces informations sont disponibles et consultables en un seul endroit.

Le fait que l'ensemble du dossier patient soit stocké numériquement représente un avantage majeur pour l'épidémiologie et pour la logistique (maintien des niveaux de médicaments en pharmacies, équipements et autres produits de santé, etc.)

Une fois numérisés, tous les dossiers forment une énorme base de données qui peut être utile d'étudier dans son ensemble pour fournir des insights aux législateurs et aux réseaux de santé partout dans le monde.

1.5.4.4 Sécurité routière et surveillance d'accès :

Avec l'augmentation du trafic routier l'OCR devient de plus en plus un besoin qu'un choix volitif désormais l'ANPR qui est une technologie basée sur l'OCR est une solution idéale et peut être implémenté dans plusieurs applications comme :

Reconnaît automatiquement les excès de vitesse et les franchissements de feu rouge.

Vérifie la carte grise et l'immatriculation du véhicule. Fournit une alerte automatique lors du passage de véhicules inscrits sur liste noire.

Contrôle des entrées et des sorties. Lève les barrières pour les véhicules autorisés.

1.5.5 Fonctionnement d'un OCR :

Le processus d'OCR est une activité composite qui comprend différentes phases. (Figure 1.14) Ces phases sont les suivantes :

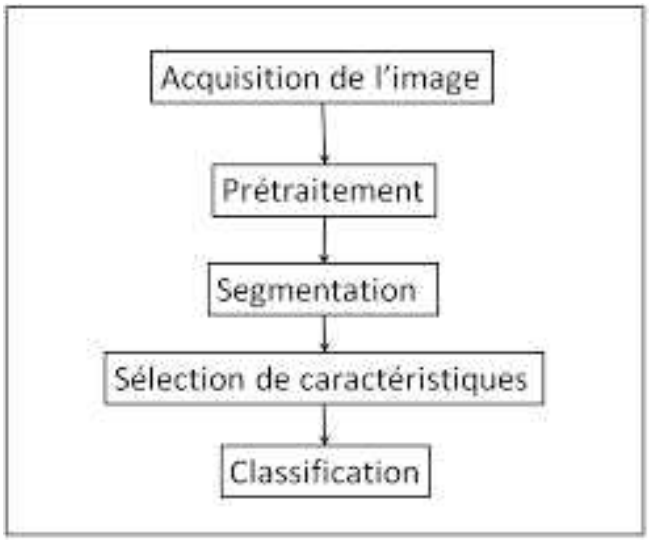


Figure 1.18 : processus de fonctionnement de l'OCR

1.5.5.1 Acquisition d'images

L'acquisition d'images est la première étape de l'OCR qui comprend l'obtention d'une image numérique et sa conversion en forme appropriée qui peut être facilement traitée par ordinateur. Cela peut impliquer la quantification ainsi que la compression d'image. Un cas particulier de quantification est la binarisation qui n'implique que deux

niveaux d'image. Dans la plupart des cas, l'image binaire suffit à caractériser l'image. La compression elle-même peut être avec ou sans perte

1.5.5.2 Prétraitement :

Comme déjà décrit dans la partie précédente s'intitulant "prétraitement", après l'acquisition d'images, il y a un prétraitement qui vise à améliorer la qualité de l'image. Et ce en utilisant différentes méthodes de segmentation(local/globale), filtrage, ou bien même de modification morphologique comme l'érosion et la dilation

1.5.5.3 Segmentation des caractères

Dans cette étape, l'image est segmentée en caractères avant de passer en phase de classification. La segmentation peut être effectuée explicitement ou implicitement en tant que sous-produit de la phase de classification. De plus, d'autres phases de l'OCR peuvent aider à fournir des informations contextuelles utiles pour la segmentation de l'image.

1.5.5.4 Extraction de caractéristiques :

Dans cette étape, diverses caractéristiques des caractères sont extraites. Ces caractéristiques identifient de manière unique les caractères. La sélection des bonnes fonctionnalités et le nombre total de fonctionnalités à utiliser est une question de recherche importante. Différents types d'éléments tels que l'ïmage elle-même, des éléments géométriques (boucles, traits) et la caractéristique statistique (moments) peuvent être utilisés. Enfin, diverses techniques telles que l'analyse des composants principaux peuvent être utilisées pour réduire la dimensionnalité de l'image.

1.5.5.5 Classification :

Il est défini comme le processus de classification d'un caractère en sa catégorie appropriée. L'approche structurelle de la classification est basée sur les relations présentes dans les Composants de l'image. Les approches statistiques reposent sur l'utilisation d'une fonction de discrimination pour classer l'image. Certaines approches de classification statistique sont le classificateur bayésien, classificateur d'arbre de décision, classificateur de réseau neuronal, le plus proche classificateur de voisinage, etc. Enfin, il y a des classificateurs basés sur une approche syntaxique qui suppose une approche grammaticale pour composer une image à partir de ses sous-constituants.

1.6 Conclusion :

Ce chapitre, nous l'avons voulu à ce qu'il soit une brève introduction sur ce que c'est que l'ANPR, tout en couvrant des généralités sur l'image et les différents traitements qu'elle peut subir, la reconnaissance des formes et la reconnaissance optique de caractères et leur impact dans l'implémentation de notre travail qui consiste en la détection de plaque d'immatriculation algérienne dans des images et la reconnaissance des caractères la composant (les chiffres).

2 Chapitre 2 : Prétraitement et réseaux de neurones

2.1 Introduction

Ce chapitre est divisé en trois parties, pour commencer nous allons présenter les différentes techniques de prétraitement qu'on va utiliser, on décrira par la suite les différentes méthodes appliquées pour l'extraction des caractéristiques, pour finir avec la reconnaissance et classification par réseau de neurone.

2.2 Prétraitement

En raison de la multitude de sources de bruits, et surtout de la multitude d'effets de ces bruits sur une image, il n'existe pas de technique de restauration générale, adaptée à toutes les situations. Il existe de nombreuses recherches permettant d'approcher ce traitement automatisé du bruit mais, en général, ces techniques cherchent à adapter les paramètres de traitements spécialisés, en fonction d'estimations calculées à partir de l'image. Ainsi nous présenterons plusieurs techniques qu'on a utilisé nous permettant de réduire les dégradations.

2.2.1 Filtrage bilatéral :

Le filtrage bilatéral proposé par Tomasi et al, repris par Paris et al, est une technique de débruitage effective de l'image. Le filtrage bilatéral est une technique qui se base sur des gaussiennes spatiales et une intensité. Il permet de faire un lissage et d'éliminer des détails inutiles, avec l'avantage de préserver les contours entre les régions de l'image ; quand on fait le lissage on ne se déplace que dans les zones semblables. La clé de cette analyse consiste à exprimer le filtre dans un espace de dimension supérieure où l'intensité du signal est ajoutée à la dimension de domaine d'origine. Il convient, avant d'aborder le filtre bilatéral, de donner la notion de la convolution gaussienne. [2]

$$gC[I]_p = \sum_{q \in S} g_\sigma(\|p - q\|) I_p$$

Où

$\|p - q\|$: La distance Euclidienne entre les endroits de pixel p et q

I_p la valeur d'image à la taille de pixel de la position p.

$g_\sigma(x)$: Le gain gaussien 2D

- **Expression du filtre bilatéral :**

La formule mathématique de ce filtre est la suivante :

$$U(\hat{x}, \hat{y}) = \frac{\int_{-3\sigma}^{+3\sigma} W_r . W_s . U(\hat{x}+i, \hat{y}+j) . d(d_{ij})}{\int_{-3\sigma}^{+3\sigma} W_r W_s . d(d_{ij})}$$

Où

$U(\hat{x}, \hat{y})$: La valeur du pixel, d : la distance entre deux pixels

W_r : Fonction d'éloignement en intensité, W_s: Fonction d'éloignement spatial,

2.2.2 Binarisation locale :

Comme déjà décrit dans le premier chapitre la binarisation permet de passer d'une image de niveaux de gris à une image binaire, nous avons choisi dans notre étude d'utiliser une binarisation locale pour sa robustesse et son efficacité.

Dans les images avec une distribution de contraste uniforme de l'arrière-plan et du premier plan comme les images de document, le seuillage global est plus approprié. Dans d'autres images dégradées comme un véhicule, où il existe un bruit de fond considérable ou une variation de contraste et d'éclairage, il existe de nombreux pixels qui ne peuvent pas être facilement classés comme premier plan ou arrière-plan. Dans de tels cas, la

binarisation avec seuillage local est plus appropriée. Que nous ayons profondément décrit dans le premier chapitre.

2.3 Extraction des caractéristiques :

Cette étape est primordiale pour la reconnaissance, elle nous permet alors d'extraire les caractéristiques (dans notre cas les caractères) pour les passer ensuite à la phase de reconnaissance.

Pour ce faire trois étapes sont nécessaire la détection des contours de l'image, traçage de ligne droite passant par les caractères en appliquant la transformé de Hough, calcul de la distance entre chaque contour et la ligne pour extraire les caractères contenus dans la plaque.

2.3.1 Détection de contour :

La détection de contour a pour but de délimiter tous les objets de l'image en incluant les caractères de la plaque d'immatriculation. Cette procédure est expliquée en détails dans les paragraphes suivants

Cette méthode commence par un balayage de l'image binaire jusqu'à la rencontre d'un pixel noir (pixel objet de valeur 1) précédé par un pixel blanc (valeur 0), le pixel objet est considéré comme premier pixel du contour et le pixel blanc sera considéré comme premier pixel voisin de ce dernier (pixel 1) (Figure 2.1.1).

Ensuite balayer le voisinage du pixel objet commençant par le pixel 2 jusqu'au pixel 8 dont la numérotation se fait dans le sens horaire commençant par le pixel 1 considéré (Figure 2.1.2), si un pixel objet est rencontré sur ce voisinage, il est considéré comme second pixel du contour, donc il faut balayer son voisinage (du pixel 2 jusqu'au pixel 8) dans le sens horaire, le pixel 1 est le pixel qui précède le second pixel contour dans le balayage précédent. Le cycle se répète jusqu'à la rencontre du premier pixel contour (fin du contour).

Remarque :

Pour chaque pixel contour trouvé on affecte une valeur (un niveau de gris) différente de ‘’0’’ et de ’’1’’ (niveau 3 par exemple), pour éviter la détection d’un même contour plusieurs fois.

L’étape suivante est de chercher un autre contour, pour cela continuer le balayage de l’image jusqu’au dernier pixel. La figure 2.3 présente l’application du suivi de contour sur une image de véhicule.

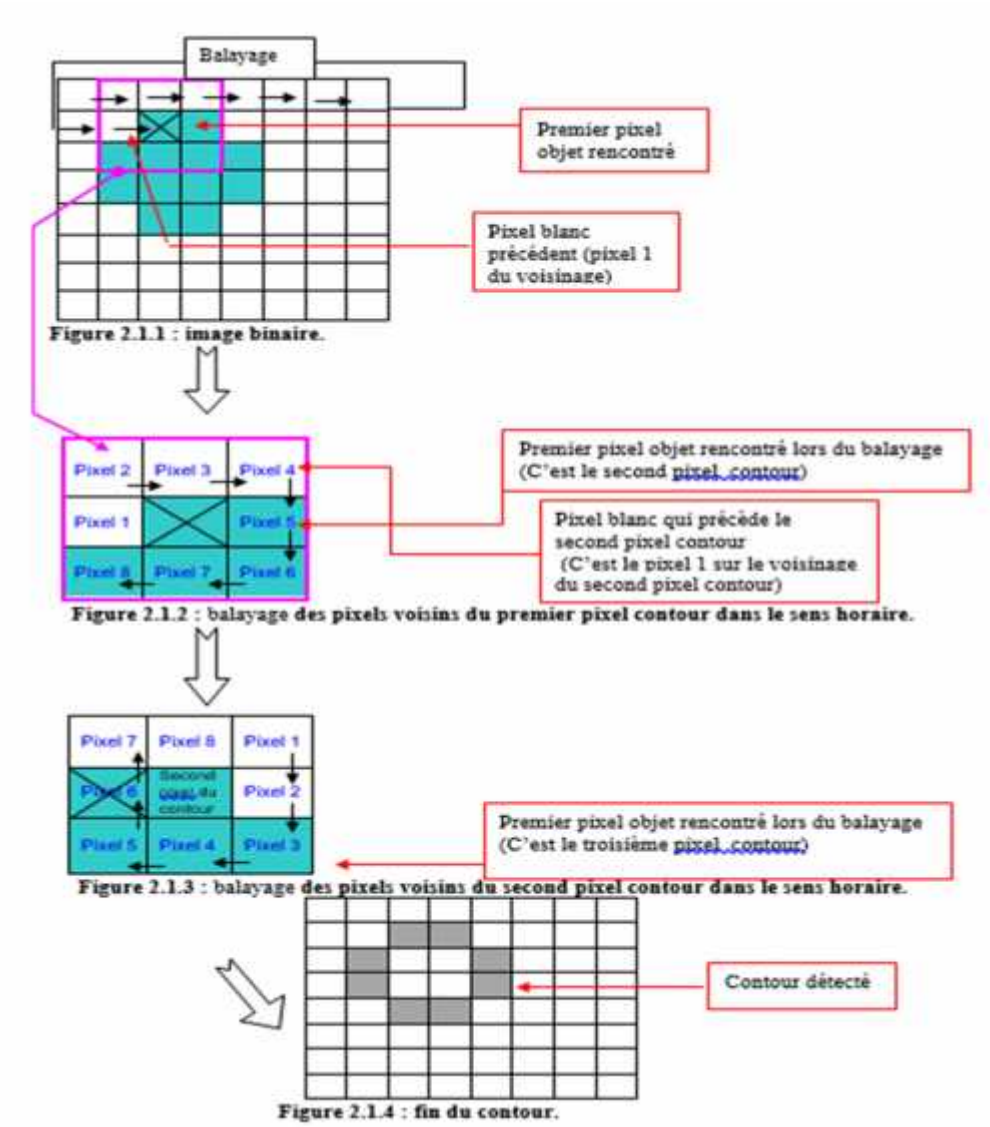


Figure 2.1: Détection de contour d’une image binaire

Figure 2.2: Image binaire.

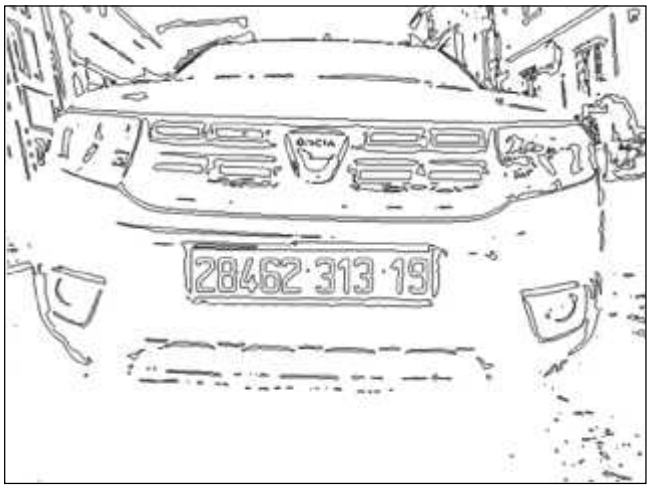


Figure 2.3: Image après détection de contour.

2.3.2 Hough Transform :

2.3.2.1 Historique :

La transformée de Hough est une technique de reconnaissance de formes inventée en 1959 par Paul Hough, faisant l'objet d'un brevet, et utilisée dans le traitement d'images numériques.

L'application la plus simple permet de détecter les lignes présentes dans une image, mais des modifications peuvent être apportées à cette technique pour détecter d'autres formes géométriques : c'est la transformée généralisée de Hough développée par Richard Duda et Peter Hart en 1972. [3]

2.3.2.2 Approche théorique :

Le problème posé est celui de la recherche et de la détection de lignes qui seraient éventuellement présentes dans une image analysée malgré les imperfections de l'image : points manquants (la ligne pouvant être partiellement masquée par un objet), bruit. La transformée de Hough consiste à représenter chaque point de contour détecté dans un espace de paramètres à deux dimensions :

- Une droite est caractérisée par deux paramètres, elle est donc représentée par un point dans cet espace de paramètres.
- Si l'on considère l'ensemble des droites passant par un point, l'image de cet ensemble est une courbe dans l'espace de paramètres.

La transformée de Hough d'un point de l'image analysée est la courbe de l'espace des paramètres correspondant à l'ensemble des droites passant par ce point.

Si des points sont colinéaires, alors toutes les courbes de l'espace de paramètres se coupent au point représentant la droite en question.

Du fait des imperfections de l'image, les points détectés ne sont pas parfaitement alignés et donc les courbes ne sont pas parfaitement concourantes. La méthode consiste donc à discrétiser l'espace de paramètres, à le découper en petits rectangles, et à

dénombrer pour chaque rectangle le nombre de courbe y passant. On construit ainsi une matrice dite d'accumulation, les maximas locaux de cette matrice correspondant à des droites probables. [3]

L'algorithme comporte donc trois étapes :

- Pour chaque point de contour détecté, détermination de la courbe correspondante dans l'espace des paramètres ;
- Construction de la matrice d'accumulation à partir de ces courbes ;
- Détection de pics dans la matrice d'accumulation.

2.3.2.3 Approche pratique :

L'espace de Hough est un plan 2D qui a un axe horizontal représentant la pente et l'axe vertical représentant l'intersection d'une ligne sur le contour de l'image. Une ligne sur une image de bord est représentée sous la forme $y = ax + b$ (Hough, 1962). Une ligne sur l'image de bord produit un point sur l'espace de Hough puisqu'une ligne est caractérisée par sa pente a et son point d'intersection b. D'autre part, un point de bord (x_i, y_i) sur l'image de bord peut être traversé par un nombre infini de lignes. Par conséquent, un point de bord produit une ligne dans l'espace de Hough sous la forme $b = ax_i + y_i$ (Leavers, 1992). Dans l'algorithme de transformation de Hough, l'espace de Hough est utilisé pour déterminer si une ligne existe dans l'image de contour.

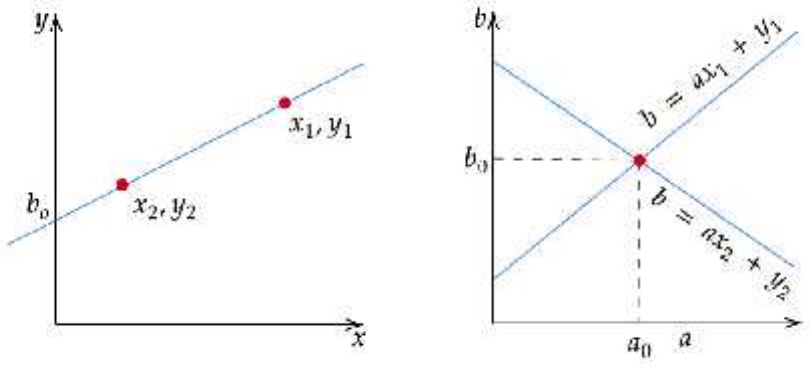


Figure 2.4 : représentation d'une ligne dans l'espace de Hough sous la forme (ax + b)

Il y a un défaut avec la représentation des lignes sous la forme de y = ax + b et l'espace de Hough avec la pente et l'interception. Sous cette forme, l'algorithme ne pourra pas détecter les lignes verticales car la pente 'a' est indéfinie/infinie pour les lignes verticales (Leavers, 1992). Par programmation, cela signifie qu'un ordinateur aurait besoin d'une quantité infinie de mémoire pour représenter toutes les valeurs possibles de 'a'. Pour éviter ce problème, une ligne droite est plutôt représentée par une ligne appelée ligne normale qui passe par l'origine et est perpendiculaire à cette ligne droite. La forme de la ligne normale est $\rho = x \cos(\theta) + y \sin(\theta)$ où est la longueur de la ligne normale et θ est l'angle entre la ligne normale et l'axe x.[4]

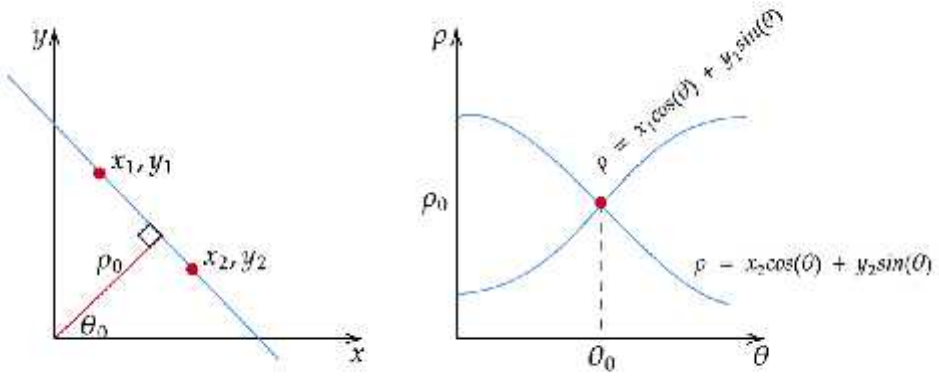


Figure 2.5 : représentation d'une ligne dans l'espace de Hough sous la forme ρ = x cos(θ) + y sin(θ)

En utilisant cela, au lieu de représenter l'espace de Hough avec la pente a et l'interception b, il est maintenant représenté avec ρ et θ où l'axe horizontal est pour les valeurs θ et l'axe vertical pour les valeurs ρ. Le mappage des points de bord sur l'espace de Hough fonctionne de manière similaire, sauf qu'un point de bord (x_i, y_i) génère désormais une courbe en cosinus dans l'espace de Hough au lieu d'une ligne droite (Leavers, 1992). Cette représentation normale d'une ligne élimine le problème de la valeur illimitée de 'a' qui se pose lorsqu'il s'agit de lignes verticales.

Comme mentionné, un point d'arête produit une courbe en cosinus dans l'espace de Hough. À partir de là, si nous devions mapper tous les points de bord d'une image de bord sur l'espace de Hough, cela générerait beaucoup de courbes en cosinus. Si deux points de bord se trouvent sur la même ligne, leurs courbes cosinus correspondantes se couperont sur une paire spécifique (ρ,θ). Ainsi, l'algorithme de transformation de Hough détecte les lignes en trouvant les paires (ρ, θ) qui ont un nombre d'intersections supérieur à un certain seuil. Il convient de noter que cette méthode de seuillage peut ne pas toujours donner le meilleur résultat sans effectuer un prétraitement comme la suppression de voisinage sur l'espace de Hough pour supprimer des lignes similaires dans l'image de bord.La figure 2.6 illustre le processus de détection des lignes dans une image oùles points jaunes dans

l'espace de Hough indiquent que des lignes existent et sont représentés par les paires ρ et θ.[4]

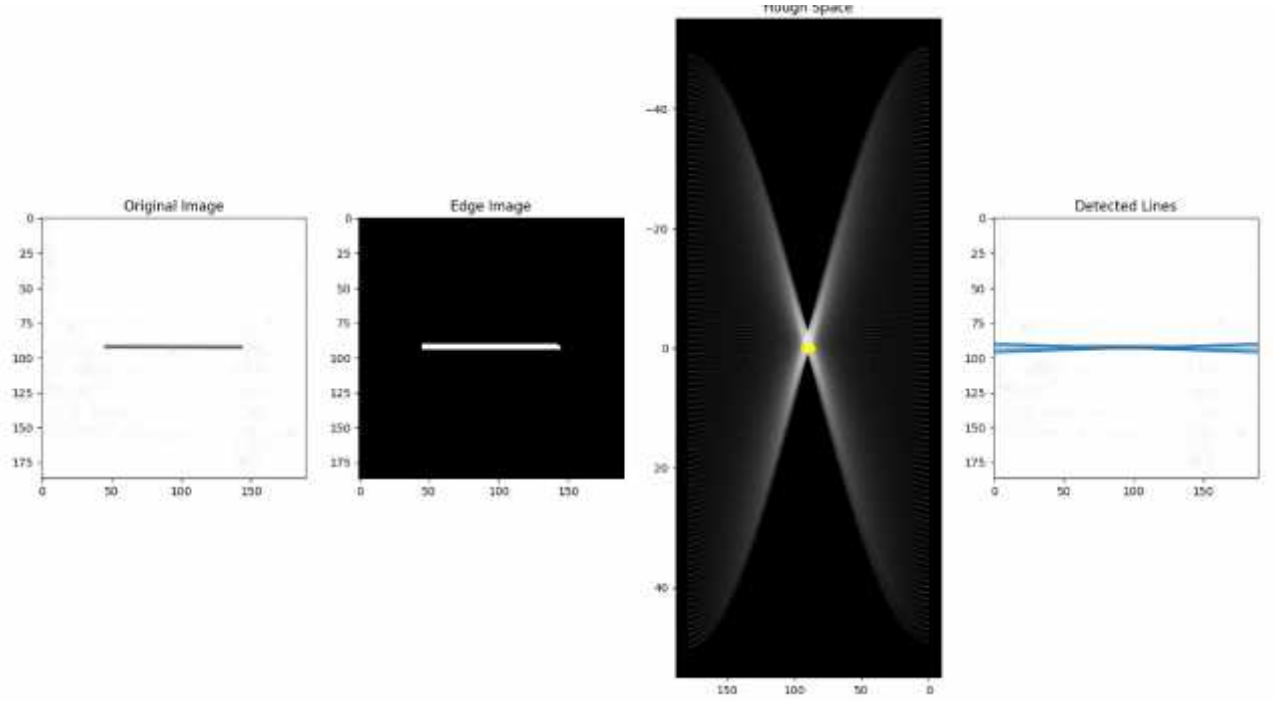


Figure 2.6 : Le processus de détection des lignes dans une image.

2.3.3 Distances :

En mathématiques, une distance est une application qui formalise l'idée intuitive de distance, c'est-à-dire la longueur qui sépare deux points. C'est par l'analyse des principales propriétés de la distance usuelle que Fréchet introduit la notion d'espace métrique, développée ensuite par Hausdorff. [Wiki]

2.3.3.1 Distance Point-Droite:

Pour déterminer la distance qui sépare un point d'une droite, il faut déterminer la longueur du segment qui joint perpendiculairement le point à la droite.

Équation:

Soit un point A (x_0, y_0) et une droite BC d'équation y = mx+b. La formule à utiliser est:

d (BC, A) = $|m*x_0 – y_0 +b| / \sqrt{(1 + m^2)}$

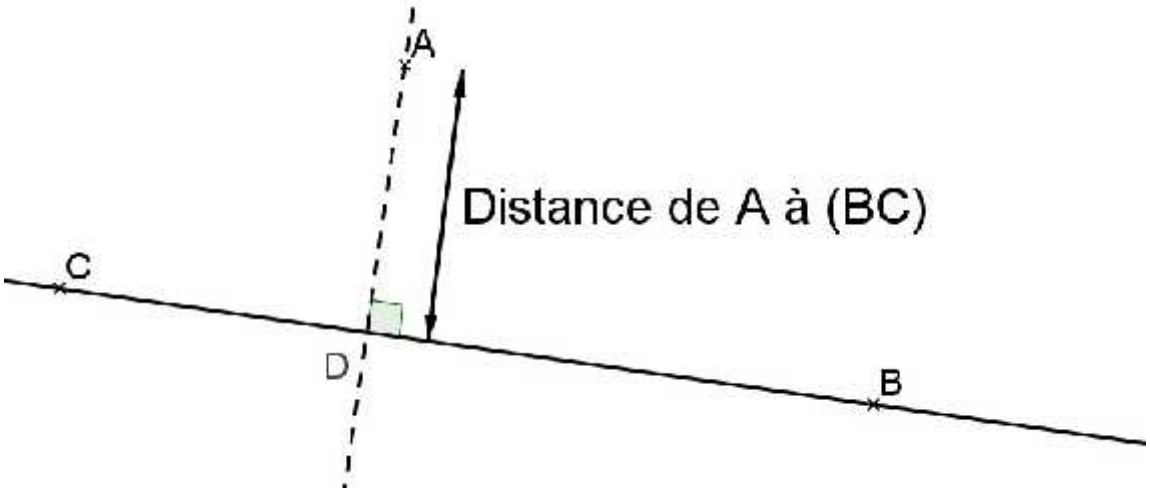


Figure 2.7 : distance point-droite

2.3.3.2 Distance Point-Point:

L'expression qui permet de calculer la distance entre A (x_0, y_0) et B (x_1, y_1) est :

d (A,B) = $\sqrt{(x_0-x_1)^2+(y_0-y_1)^2}$.

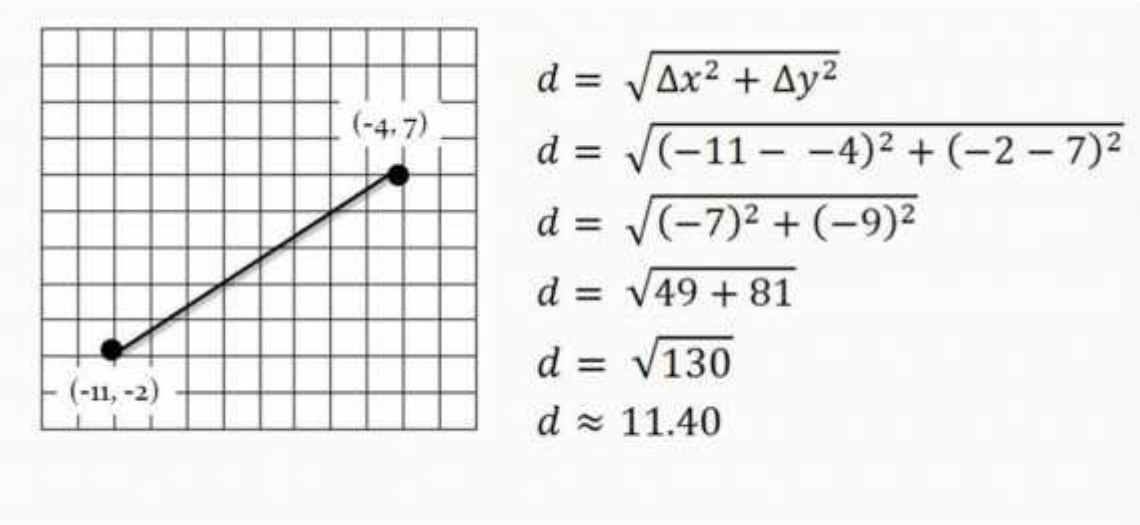


Figure 2.8 : Distance Point-Point

2.4 Réseaux de neurones

2.4.1 Reconnaissances à base de réseaux de neurones :

Le cerveau humain est capable de s'adapter, d'apprendre et de décider, et c'est sur ce fait que des chercheurs se sont intéressés à comprendre son principe de fonctionnement et de pouvoir l'appliquer au domaine de l'informatique. C'est ainsi que dans les années cinquante on formalisa le neurone en un modèle mathématique à partir du modèle biologique [14].

Dans ce chapitre on va s'intéresser à l'étude du réseau de neurone, le passage du neurone biologique au neurone mathématique (artificiel), et à la fin on verra le perceptron multicouches (PMC) qui est le modèle adéquat à notre travail.

2.4.2 Modelé biologique :

2.4.2.1 Définition du neurone biologique :

Le neurone biologique est une cellule vivante spécialisée dans le traitement des signaux électriques. Les neurones sont reliés entre eux par des liaisons appelées axones et dendrites. Ces axones vont eux-mêmes jouer un rôle important dans le comportement logique de l'ensemble. Ces axones conduisent les signaux électriques de la sortie d'un neurone vers l'entrée (synapse) d'un autre neurone. Les neurones font une sommation des signaux reçus en entrée et en fonction du résultat obtenu, vont fournir un courant en sortie [15].

2.4.2.2 Structure du neurone biologique :

Le système nerveux compte plus de 1000 milliards de neurones interconnectés. Bien que les neurones ne soient pas tous identiques, leur forme et certaines caractéristiques permettent de les repartir en quelque grande classes. En effet, il est aussi important de savoir, que les neurones n’ont pas tous un comportement similaire en fonction de leur affectation.

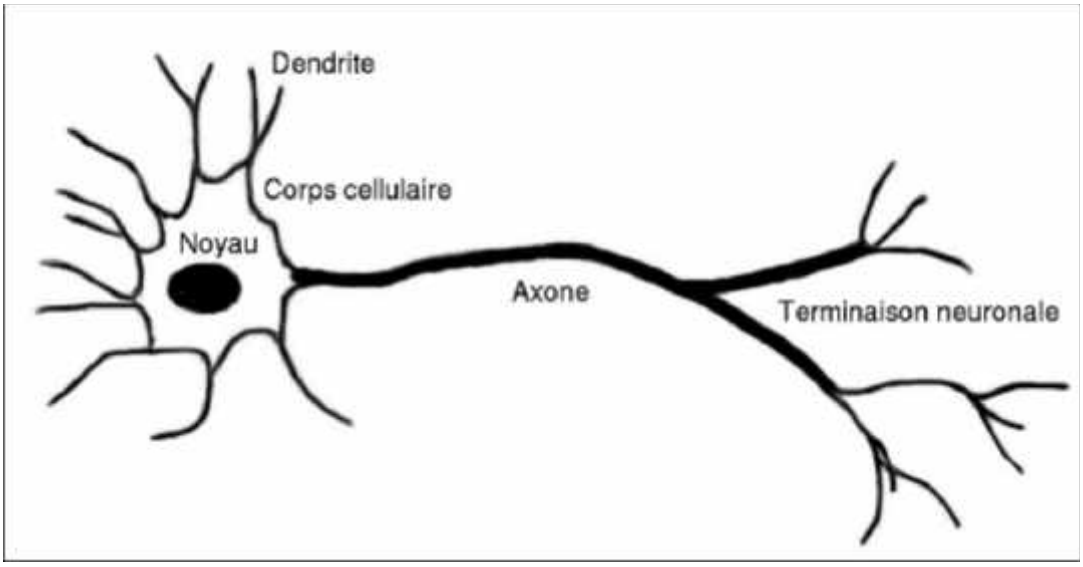


Figure 2.7: Morphologie du neurone biologique.

a. Corps cellulaire :

Il contient le noyau du neurone ainsi que la machine biochimique nécessaire à la synthèse des enzymes. Ce corps cellulaire de forme sphérique ou pyramidale contient aussi les autres molécules essentielles à la vie de la cellule. Sa taille est de quelques microns de diamètre.

b. Dendrites :

Ce sont des prolongements implantés sur le corps cellulaire qui se présente sous forme d'arborisations fines et courtes, disposées irrégulièrement et se terminant en de très nombreuses ramifications. Le nombre des dendrites varie selon chaque type de cellule, certaines possédant plusieurs dendrites, et d'autre n'en possède qu'un seuls les signaux envoyés au neurone sont captés par les dendrites [16].

c. Axones :

On lui donne le nom également de cylindraxe.

L’axone ou cylindraxe se présente sous forme d’une tige allongée, de surface lisse, de calibre invariant. Il n’existe qu’un seul axone par cellule nerveuse. L’axone est parfois très court, mais sa longueur est parfois considérable [17].

d. Synapse :

Une synapse est une jonction entre deux neurones, et généralement entre l'axone d'un neurone et une dendrite d'un autre neurone (mais il existe aussi des synapses axo-axonales par exemple) [18].

2.4.3 Fonctionnement d’un neurone :

A point de vue fonctionnel, il faut considérer le neurone comme une entité polarisée, c'est-à-dire que l’information ne se transmet que dans un seul sens des dendrites vers le corps, et du corps vers l’axone.

Le neurone va donc recevoir des informations, venant d’autres neurones, grâce à ces dendrites. Il va ensuite y avoir sommation, au niveau du corps cellulaire, de toutes ces informations et via un potentiel d’action (un signal électrique) le résultat de l’analyse va transiter le long de l’axone jusqu’aux terminaisons synaptiques.

A cet endroit, lors de l’arrivée du signal, des vésicules synaptiques vont finir avec la membrane cellulaire, ce qui va permettre la libération des neurotransmetteurs (médiateur chimique) dans la fonte synaptique. Le signal électrique ne pouvant pas passer la synapse (dans le cas d’une synapse chimique), les neurotransmetteurs permettent donc le passage des informations, d’un neurone à un autre.

Les neurotransmetteurs excitent (neurotransmetteurs excitateurs) ou inhibent (neurotransmetteurs inhibiteurs) le neurone suivant et peuvent ainsi générer ou interdire la propagation d’un nouvel influx nerveux.

Les synapses possèdent une sorte de « mémoire » qui leur permet d'ajuster leur fonctionnement. En fonction de leur « histoire », c'est-à-dire de leur activation répétée ou non entre deux neurones, les connexions synaptiques vont donc se modifier.

Ainsi la synapse va faciliter ou non le passage des influx nerveux. Cette plasticité est à l'origine des mécanismes d'apprentissage [28].

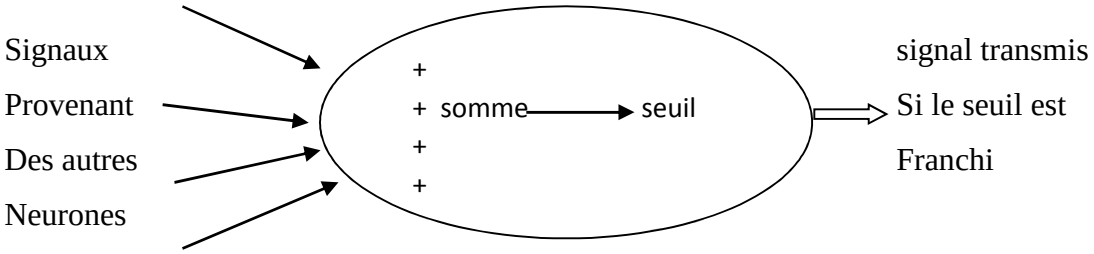


Figure 2.8: Schéma de principe de fonctionnement d'un neurone

2.4.3.1 Définition du réseau de neurones :

Ce que l'on désigne habituellement par "réseau de neurones" (neural network), ou réseau neuromimétique est un réseau de neurones artificiels basé sur un modèle simplifié de neurone. Ce modèle permet certaines fonctions du cerveau, comme la mémorisation associative, l'apprentissage par l'exemple, le travail en parallèle, etc. Cependant le neurone formel ne possède pas toutes les capacités des neurones biologiques, comme le partage de synapses, l'activation membranaire ou la structuration prénatale des neurones, par conséquent les réseaux de neurones actuels sont loin d'avoir les possibilités du cerveau.

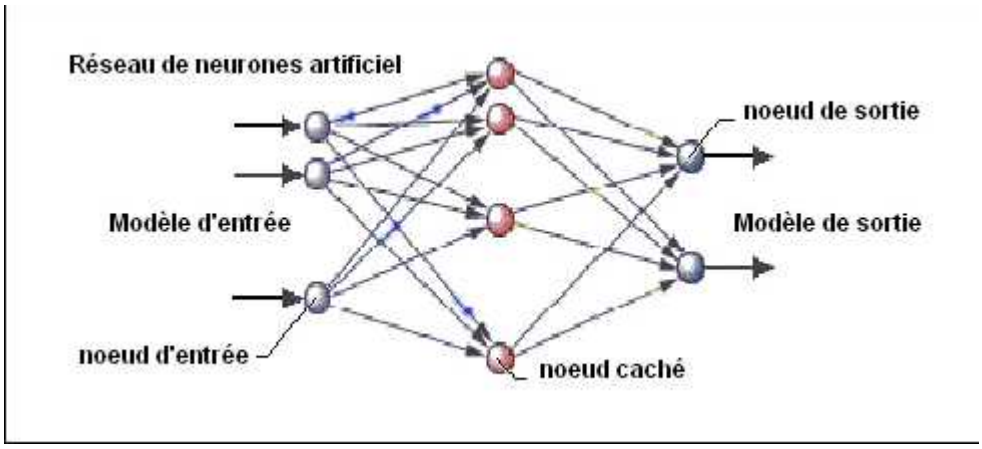


Figure 2.9: Réseau de neurones artificiel

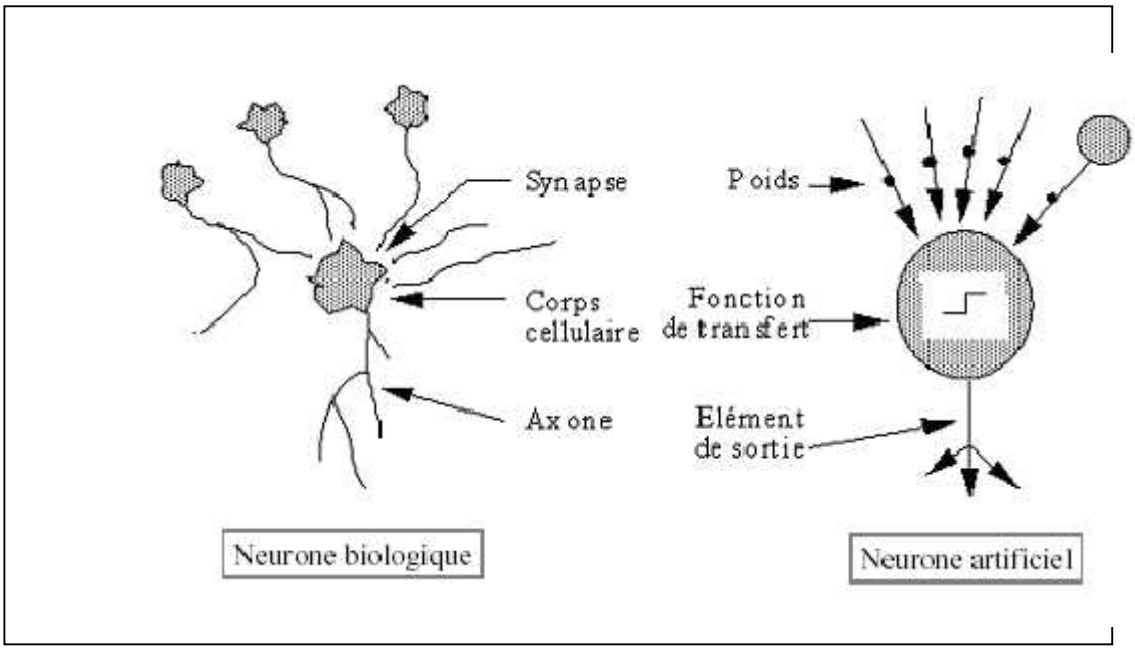


Figure 2.10 : Dualité entre un neurone biologique et un neurone artificiel

2.4.3.2 Différents types des Réseaux de Neurones Artificiels

a. Le perceptron monocouche :

C'est un des premiers réseaux de neurone, conçu en 1958. Il est linéaire et monocouche. Il est inspiré du système visuel. La première couche (couche d'entrée) représente la rétine. Les neurones de la couche suivante sont les cellules, et les neurones de la couche finale sont les cellules de la décision.

Les sorties des neurones ne peuvent prendre que des états (-1 et 1 ou 0 et -1).

Seuls les poids des liaisons entre la couche d'association et la couche finale peuvent être modifiés.

La règle de modification des poids utilisée est :

- Si la sortie (celle d'une cellule de décision donc) est égale a la sortie désirée, le poids de la connexion entre ce neurone et le neurone d'association qui lui est connecté n'est pas modifié.

- Dans le cas contraire le poids est modifié en fonction de l'entrée :

$W_i <= W_i+K.(d-s)$

Avec K : constante positive, s : sortie et d : sortie désirée.

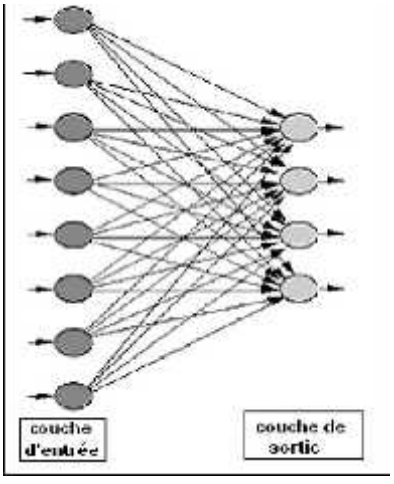


Figure 2.11 : Perceptron monocouche

b. Le perceptron multicouche (PMC) :

Le Perceptron Multicouche ou MLP (Multi Layer Perceptron) est une amélioration du perceptron comprenant une ou plusieurs couches intermédiaires dites couches cachées, pour modifier leurs poids, un algorithme de rétro propagation du gradient. Il s'agit toujours de minimiser l'erreur quadratique, ce qui est assez simple quand on utilise une fonction f dérivable (la sigmoïde par exemple). On propage la modification des poids de la couche de sortie jusqu'à la couche d'entrée [20].

Les PMC agissent comme un séparateur non linéaire et peuvent être utilisés pour la classification, le traitement de l'image ou l'aide de la décision

Couche cachée Couche de sortie

$$c_j = 1 \text{ si } \sum_k w_{jk} x_k > 0 \; 0 \quad o_i = 1 \text{ si } \sum_k w_{ik} c_k > 0$$

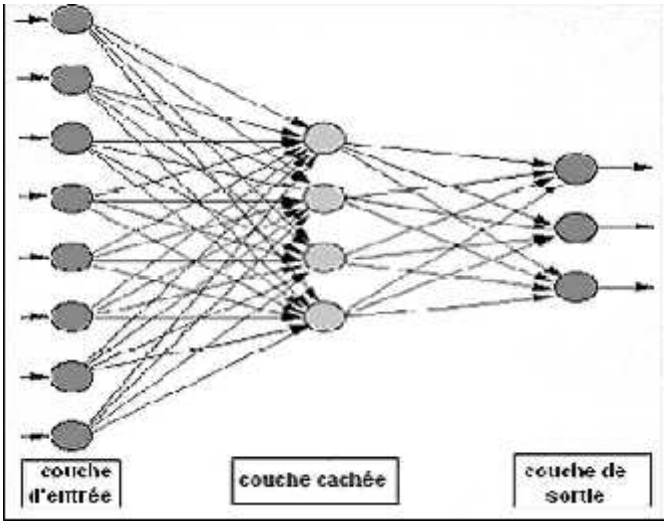


Figure 2.12 : Perceptron Multicouche (PMC)

- **Couche d’entrée :**

Le nombre de nœuds dans cette couche est noté N1, il correspond a la taille des vecteurs représentant les motifs d’apprentissage.

- **Couche de sortie :**

C’est la couche de décision du réseau. Généralement le nombre de nœuds dans cette couche est noté N2, il correspond au nombre de classes d’apprentissage des motifs que le réseau apprend à reconnaitre.

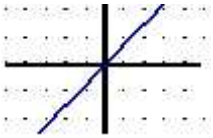
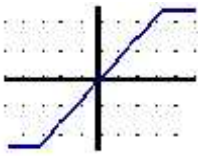
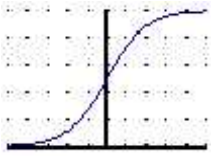
- **Couche cachée :**

Le choix du nombre de nœuds cachés nécessaire doit obéir à un compromis optimisant d’apprentissage [20].

2.4.4 Fonction d’activation du PMC :

Chaque neurone calcule sa valeur de sortie y à partir de la somme pondérée de ses entrées et de ses poids, il existe différentes fonctions d’activation permettant de calculer cette valeur. Dans sa première version, le neurone formel était donc implémenté avec une fonction à seuil, mais de nombreuses versions existent. Ainsi le neurone de McCulloch et Pitts a été généralisé de différentes manières, en choisissant d'autres fonctions d'activations, comme les fonctions linéaires ou les sigmoïdes par exemple.

Voici un tableau récapitulatif de différents types de fonctions d’activation les plus utilisées, avec leurs équations mathématiques et leurs dérivées [20].

CATEGORIE	*TYPE*	*EQUATION*	*ALLURE*	*DERIVEE*
SEUIL	*Binaire (fonction de Heaviside)*	*f(x)=0 si f(x)>0* *f(x)=1 si f(x)≤ 0*		-
	Signe	*f(x)=1 si f(x)>0* *f(x)=-1 si f(x)≤ 0*		-
LINEAIRE	*Identité*	*f(x)=x*		*f'(x)=1*
	Saturé *Positif*	*f(x,k)=0 si f(x)<0* *f(x,k)=1 si f(x)≥1/k* *f(x,k)=k.x sinon*		*f(x,k)=0 si f(x)<0* *f(x,k)=0 si f(x)≥1/k* *f(x,k)=k sinon*
	Saturé symétrique	*f(x,k)=-1 si f(x)<-1/k* *f(x,k)=1 si f(x)≥1/k* *f(x,k)=0 sinon*		*f(x,k)=0 si f(x)<-1/k* *f(x,k)=0 si f(x)≥1/k* *f(x,k)=k sinon*
SIGMOIDE	*Positive (type logistique)*	$f(x,k) = \frac{1}{1+e^{-k}}$		$f'(x,k) = \frac{k}{2+e^{-k}+e^{k}}$

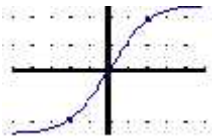
	Symétrique (type tanh)	$f(x,k) = \frac{1}{1+e^{-k}} - 1$		$f'(x,k) = \frac{2.k}{2+e^{-k}+e^{k}}$

Figure 2.13 : tableau récapitulatif des différentes fonctions d'activation

2.4.5 Types d'apprentissage :

Les procédures d'apprentissage peuvent être classées en trois catégories : non supervisé, supervisé et semi-supervisé.

2.4.5.1 Apprentissage non supervisé

L'apprentissage non supervisé, dans lequel on se contente de présenter des formes sans indication sur leur identité. Une règle d'apprentissage est la règle de Hebb, qui revient à augmenter le poids de la connexion entre deux cellules si celles-ci sont simultanément actives et à le diminuer dans le cas contraire. Les cartes de Kohonen, utilisent un apprentissage non supervisé compétitif, inspiré de la loi de Hebb.

2.4.5.2 Apprentissage supervisé

Dans l'apprentissage supervisé, on impose une réponse en sortie du réseau pour une forme d'entrée donnée (dispose d'un comportement de référence vers lequel il tente de faire converger le réseau). Dans ce cas, il s'agit de calculer l'erreur commise par le réseau comme une fonction de la "distance" entre la sortie désirée et la sortie obtenue. Il existe deux grands types d'erreurs utilisés en pratique, correspondant à deux critères d'optimisation :

> Le critère des moindres carrés fondé sur l'utilisation d'une distance euclidienne entre les formes,

> Le critère d'entropie croisée dans lequel on considère les sorties du réseau comme des distributions de probabilités sur les variables d'entrée. Il s'agit alors de maximiser l'information mutuelle entre les sorties réelles et désirées.

Dans le cas du perceptron multicouches, l'algorithme d'apprentissage, dit de rétro propagation du gradient d'erreur, est itératif. Le principe est d'adapter les différents poids des connexions du réseau à chaque présentation d'une forme en entrée, selon le gradient de l'erreur commise en sortie.

2.4.5.3 Apprentissage semi-supervisé

L'apprentissage semi-supervisé suppose qu'un comportement de référence précis n'est pas disponible, mais qu'en revanche il est possible d'obtenir des indications qualitatives (par exemple, correct/incorrect) ou lacunaires (par exemple, une indication tous les N patrons) sur les performances du réseau.

2.4.6 Calcul des poids synaptiques :

La retro propagation (back propagation) est une méthode de calcul des poids pour un réseau à apprentissage supervisé qui consiste à minimiser l'erreur quadratique de sortie (somme des carrés de l'erreur de chaque composante entre la sortie réelle et la sortie désirée).

D'autre méthode de modification des poids sont plus « locales », chaque neurone modifie ses poids en fonction de l'activation ou non des neurones proches. C'est le cas des réseaux suivant la règle de Hebb (Hebbian rule) ou les réseaux de Kohonen.

2.4.7 Application du PMC :

La liaison entre la couche d'entrée et la couche cachée est représentée par les poids synaptiques Wij, par contre celle entre la couche cachée et la couche de sortie est représentée par les poids synaptiques Vjk où :

- i=1,……,n. n étant le nombre de neurones utilisés dans la couche d'entrée.
- J=1 ,….m. m étant le nombre de neurones utilisés dans la couche cachée.
- K=1,…..,l. l étant le nombre de neurones utilisés dans la couche de sortie.

2.4.8 Règle d'apprentissage :

Fondamentalement, le processus de rétro propagation se divise en deux passages aux différentes couches du réseau ; le passage vers l'avant, et passage vers l'arrière. Dans le premier cas, le vecteur d'entrée est appliqué à la couche d'entrée ;

Son effet propage à travers le réseau couche par couche. Finalement un ensemble de sorties est produit comme étant des sorties actuelles du réseau. Pendant cette étape, les poids synaptiques du réseau sont tous fixés. Par contre, dans la deuxième étape, les poids synaptiques sont ajustés suivant la règle de correction de l'erreur.

Les réponses actuelles sont soustraites des sorties désirées pour produire l'erreur du signal. Les poids synaptiques sont ajustés afin d'approcher le plus que possible les réponses actuelles aux sortie désirées. Le réseau perceptron multicouche a trois caractéristiques essentielles :

Le modèle de chaque neurone du réseau inclut la non linéarité à la fin des sorties. Le point le plus important ici est que la non linéarité est différentielle à chaque nœud du réseau ; contrairement à ce qui a été utilisé dans le perceptron de Rossinante. Généralement la forme non linéaire satisfaisant cette exigence, est la fonction sigmoïde définie par :

$$Y_i = \frac{1}{1 + e^{-X_i}} \qquad \text{..... (2.1)}$$

Ou Xi représente le niveau d'activité interne propre au neurone i, et Yi est la sortie actuelle du même neurone.

La présence de non linéarité est très importante, car l'erreur du signal peut être fortement réduite entre l'entrée et la sortie. Le réseau contient un ou plusieurs neurones de la couche cachée. Ces neurones permettent au réseau de mieux apprendre les taches les plus complexes par l'extraction progressive des caractéristiques les plus significatives des motifs d'entrées (vecteurs).

Le réseau présente une très grande connectivité inter neurones déterminée par les synapses du réseau. Le changement de la connectivité recommande le changement de la population des connexions synaptiques ou leurs poids.

2.4.9 Algorithme de retro propagation d'erreur :

La retro propagation est une méthode de calcul des poids synaptiques, l'objectif de cet algorithme est de modifié les poids synaptiques, et minimiser l'erreur quadratique apparente mesurée sur l'ensemble d'apprentissage supervisé.

On utilise un réseau multicouche avec une couche de sortie ayant M neurones, chaque neurone représente une classe. Il faut que la sortie du neurone M soit égale à 1 et le reste égal à zéro.

L'algorithme de rétro propagation du gradient comprend les étapes suivantes :

1) Initialisation des poids à des valeurs aléatoires entre **[-0.5, 0.5].**

2) Sélection aléatoire d'un motif dans la base d'apprentissage, et attribution des sorties désirées suivant l'entrée pour chaque motif.
3) Appliquer l'entrée pour X sur le réseau et calculer les sorties actuelles de chaque couche.

$$Y_j = \frac{1}{1 + e^{\sum_{i=1}^{N} X_i . W_{ij}}} \qquad \text{..... (2.2)}$$

N : nombre de neurone de la couche d'entrée.

$$S_i = \frac{1}{1 + e^{\sum_{j=1}^{M} Y_i . v_{ij}}} \qquad \text{..... (2.3)}$$

M : nombre de neurone de la couche cachée.

4) Calculer la somme des erreurs quadratiques au niveau des neurones de la coche de sortie

$$E_i = \sum_{j=1}^{M} (D_j - S_j)^2 \qquad \text{..... (2.4)}$$

D_j : est sortie désirée de j neurone et Si réponse actuelle du j neurones.

Si la base d'exemple n'est pas épuisée, aller à l'étape 2, sinon aller à l'étape 5.

5) Calculer l'erreur totale de tous les exemplaires :

$$ET = \sum_{i=1}^{cap} E_i \quad \text{..... (2.5)}$$

Cap : est le nombre d'exemples traités.

Si ET est inférieure à une valeur seuil fixée au départ alors aller à la fin.

Sinon, aller à l'étape 6.

6) Pour chaque sortie du réseau calculer le terme d'erreur :

$$\delta_i^1 = (D_j - S_j).(1-S_j).S_j \text{ ..... (2.6)}$$

Pour le neurone j de la couche cachée, calculer l'erreur :

$$\delta_j^2 = (1 - Y_j).Y_j \sum \delta_j^1 . v_{ij} \text{ ..... (2.7)}$$

7) Mettre à jour chaque poids synaptique du réseau :

$$w_{ij}(k+1) = w_{ij}(k) + \alpha.\delta_j^2.X_i + \zeta.(w_{ij}(k) + w_{ij}(k-1)) \text{...(2.8)}$$

$$w_{ij}(k+1) = v_{ij}(k) + \alpha.\delta_j^1.X_i + \zeta.(v_{ij}(k) + v_{ij}(k-1)) \text{…(2.9)}$$

Avec :

α : Le coefficient d'apprentissage compris entre 0 et 1.

ζ : Coefficient de viscosité.

Wij : Matrice des poids synaptiques entre la couche d'entrée et la couche cachée.

Vij : Matrice des poids synaptiques entre la couche d'entrée et la couche de sortie.

2.4.10 Domaines d'application des réseaux de neurones :

Il existe beaucoup d'applications de reconnaissance des formes basées sur les réseaux de neurones on trouve en particulier :

- La reconnaissance de forme. Dans ce type de problème, les données d'entrée représentent l'information recueillie par un ou plusieurs capteurs (caméra, sonar, micro). Le but est de reconnaître le ou les objets perçus par le capteur.
- En reconnaissance de la parole : Les entrées peuvent être une segmentation de la fréquence d'un signal sonore, et les sorties identifient la valeur de tel ou tel phénomène associé à l'entrée. Ou bien on peut transcrire le langage parlé en texte ASCII.
- Reconnaissance de cibles : Applications militaires qui utilisent des données issues d'images vidéo ou infrarouge pour déterminer la présence d'une cible ennemie.
- Reconnaissance de Codes postaux : Cette application fût l'un des premiers problèmes de reconnaissance d'images assez complexe, en raison de la variabilité des styles d'écritures, mais en plus cette application est moins onéreuse que de rectifier une erreur de tri.
- Reconnaissance d'écriture : Qu'elle soit manuscrite ou imprimée, le réseau va être confronté à un problème de classification, par rapport à la base d'apprentissage qui constitue la base de données.

- La prédiction et le contrôle de processus. Le problème de la prédiction consiste à estimer l'état futur d'un processus à partir d'un historique de son

comportement et des variables environnementales attenantes. En contrôle, il s'agit non seulement d'estimer passivement l'état futur, mais aussi d'intervenir de façon à atteindre un but donné. Dans ce cadre, un domaine où les réseaux ont un succès incontestable est dans le domaine de la finance. Dans un contexte plus industriel, les applications sont diverses ; citons la prédiction de séries temporelles (par exemple, la consommation électrique ou le contrôle automatique de processus dynamiques.

- En médecine. Les entrées du réseau peuvent être un ensemble de valeurs résultant d'une liste de tests médicaux, les sorties étant les symptômes possibles de maladies infectieuses.

2.4.11 Propriétés et limites des réseaux de neurones :

a- Propriétés

L'intérêt porté aujourd'hui aux réseaux de neurones tient sa justification dans les quelques propriétés fascinantes qu'ils possèdent [21] :

- La capacité d'adaptation : Elle se manifeste par la capacité d'apprentissage qui permet de tenir compte de nouvelles contraintes ou de nouvelles données du monde extérieur.

- Le parallélisme : Les réseaux de neurones sont considérés comme un ensemble d'entités élémentaires qui travaillent simultanément. Le parallélisme permet une rapidité de calcul supérieur mais exige de poser différemment les problèmes à résoudre.

- La capacité de généralisation. La capacité de généralisation d'un

réseau de neurones est sa capacité à donner une réponse satisfaisante à une entrée qui ne fait partie de son ensemble d'apprentissage.

b- Limites

Les principales limites actuelles sont [21] :

- La plupart des réseaux de neurones sont simulés sur des machines séquentielles. Ce qui entraîne des temps de calculs importants dès que la taille du problème devient grande.
- Incapacité d'expliquer les résultats qu'ils fournissent. Les réseaux de neurones se comportent comme des boites noires dont les règles de fonctionnement sont inconnues.

2.5 Solution proposé : modèle d’apprentissage à l’aide des réseaux de neurones:

Les réseaux de neurones "artificiels" s'inspirent du cerveau organique, traduit en ordinateur. La comparaison n'est pas parfaite, mais on retrouve des neurones, des activations et beaucoup d'inter connectivité, même si les processus sous-jacents sont très différents.

2.5.1 Construction du réseau :

Pour créer notre réseau de neurones, nous avons suivi un certain nombre d'étapes afin de le rendre aussi efficace que possible.

Tout d'abord, nous commençons par créer un fichier nommé layer_model.py qui contiendra toutes les classes nécessaires à la création de notre modèle de réseau de neurones.

2.5.1.1 Couche dense (couche entièrement connectée) :

La première classe de notre fichier sera une classe nommée Layer Dense qui contiendra trois méthodes principales (__init__, forward, backward) qui seront décrites en détail juste après.

a. Méthode d'initialisation :

La méthode init est la méthode qui va initialiser les poids et les biais, elle prend un ensemble de poids et de biais comme paramètres.

```
def __init__(self, weights, biases):
    self.weights = weights
    self.biases = biases
```

Figure 2.14 : Fonction d'initialisation

b. Méthode de propagation en avant :

La méthode de propagation en avant est la fonction qui va prendre les entrées et les faire passer dans les neurones (passage avant).

La propagation en avant consiste principalement à effectuer un produit scalaire entre nos entrées et nos poids et à ajouter le biais à chaque fois.

```
def forward(self, inputs):
    self.inputs = inputs
    self.output = np.dot(inputs, self.weights)+self.biases
```

Figure 2.15: Fonction de propagation en avant

2.5.1.2 Fonctions d'activation :

Pour les fonctions d'activation, nous allons avoir deux fonctions différentes.

a. La fonction d'activation linéaire rectifiée RelU :

La fonction d'activation linéaire rectifiée est littéralement y=x, coupée à 0 du côté négatif. Si x est inférieur ou égal à 0, alors y est 0 sinon, y est égal à x.

Nous allons utiliser la RelU pour notre couche cachée, et la raison pour laquelle nous avons utilisé la fonction d'activation RelU est que, même si elle est assez simple,

c'est aussi une fonction d'activation puissante et c'est la fonction d'activation la plus utilisée pour diverses raisons - principalement la vitesse et l'efficacité.

La fonction d'activation ReLU est extrêmement proche d'être une fonction d'activation linéaire tout en restant non linéaire, en raison de cette courbure après 0. Cette simple propriété est cependant très efficace.

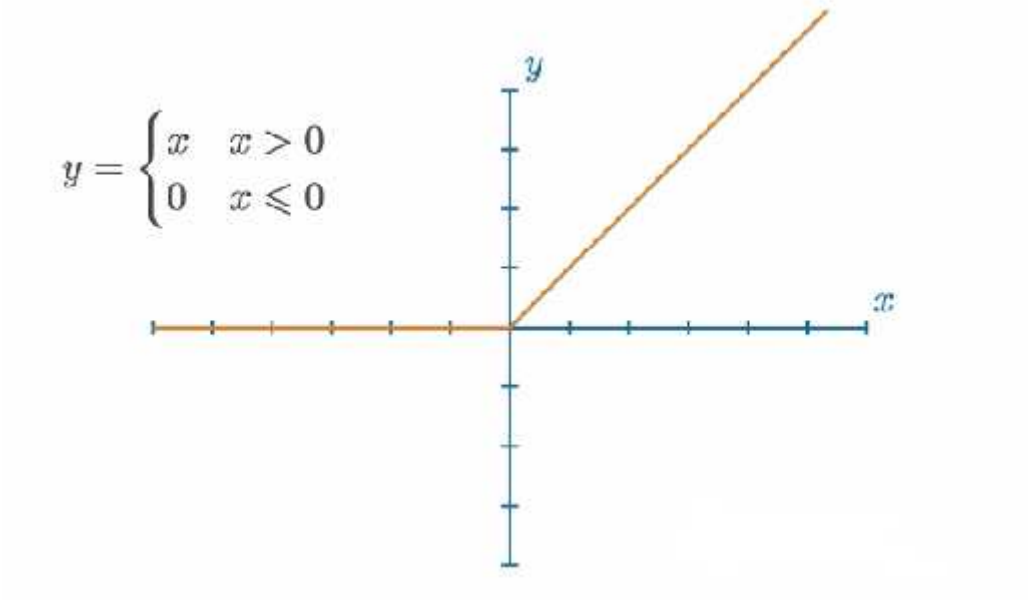


Figure 2.16: Fonction d'activation RelU graph

```
class Activation_ReLU:

    def forward(self, inputs):
        self.inputs = inputs
        self.output = np.maximum(0, inputs)
```

Figure 2.17: Fonction d'activation RelU

b. Fonction d'activation Softmax :

Pour la couche de sortie, nous avons utilisé la fonction d'activation Softmax et la raison en est la suivante :

Dans notre cas, nous cherchons à faire de ce modèle un classificateur, et nous voulons donc une fonction d'activation destinée à la classification. L'une de ces fonctions est la fonction d'activation Softmax. La raison en est que dans le cas de la classification, ce que nous voulons voir est une prédiction de la classe que le réseau "pense" que l'entrée représente. Cette distribution renvoyée par la fonction d'activation Softmax représente des scores de confiance pour chaque classe et sa somme sera égale à 1. La classe prédite est associée au neurone de sortie qui a renvoyé le score de confiance le plus élevé.

Softmax est représentée par L'équation :

$$S_{i,j} = \frac{e^{z_{i,j}}}{\sum_{l=1}^{L} e^{z_{i,l}}}$$

```
class Activation_Softmax:
    def forward(self, inputs):
        self.inputs = inputs
        exp_values = np.exp(inputs - np.max(inputs, axis=1, keepdims=True))
        probs = exp_values / np.sum(exp_values, axis=1, keepdims=True)
        self.output = probs
```

Figure 2.18: Fonction d'activation Softmax

2.5.2 Apprentissage :

Avant de commencer notre processus d'apprentissage, nous devons définir le nombre de couches dont nous avons besoin ainsi que le nombre de neurones de chaque couche. Dans notre cas, nous avons utilisé une couche d'entrée, une couche cachée et une couche de sortie.

Les poids de notre réseau seront initialisés de façon aléatoire, et les biais seront fixés à zéro pour commencer.

L'entrée sera soit des données de formation réelles, soit les sorties des neurones de la couche précédente du réseau neuronal.

2.5.2.1 Préparation des données :

Dans les réseaux neuronaux, nous appelons nos données des tenseurs, et un objet tenseur est simplement tout objet qui peut être représenté comme un tableau. Dans notre cas, nos tenseurs seront des images.

Pour préparer nos données, nous devons trier nos images et les organiser dans des dossiers nommés respectivement de 0 à 9. Ensuite, nous commençons par parcourir nos dossiers et copier les images dans un tableau à 2 dimensions. Les lignes contiendront les images et les colonnes contiendront la classe (étiquette) respective. Après avoir chargé notre ensemble de données, nous devons le mélanger pour que le réseau neuronal puisse généraliser ses prédictions.

2.5.2.2 Initialisation des poids et des biais :

Au cours de l'apprentissage, les paramètres de la fonction __init__ seront modifiés car nous n'avons pas d'ensemble de poids et de biais, nous en recherchons un.

Nous définissons les poids comme étant aléatoires et les biais comme étant nuls. Pour ce faire, nous devons apporter quelques modifications à la fonction init de Layer Dense.

Les paramètres seront le nombre d'entrées et les neurones qui seront utilisés dans notre couche dense.

Par conséquent, nous initialisons les poids et les biais en fonction de ces nombres. (Chaque neurone a un ensemble de poids basé sur le nombre d'entrées et un biais).

```
def __init__(self, n_inputs, n_neurons):
    self.weights = 0.10 * np.random.randn(n_inputs, n_neurons)
    self.biases = np.zeros((1, n_neurons))
```

Figure 2.19: Fonction d'initialisation aléatoire

2.5.2.3 Propagation en avant :

Nous effectuons un passage en avant pour la première couche (couche cachée), puis nous utilisons la fonction d'activation RelU sur la sortie du passage en avant.

Nous envoyons la sortie de la fonction d'activation RelU à la deuxième couche (couche de sortie), puis nous utilisons la fonction d'activation Softmax sur la sortie.

Avec un modèle initialisé de façon aléatoire, ou même un modèle initialisé avec des approches plus sophistiquées, notre objectif est de former, ou d'enseigner, un modèle au fil du temps. Pour entraîner un modèle, nous modifions les poids et les biais afin d'améliorer la précision et la confiance du modèle. Pour ce faire, nous calculons le niveau d'erreur du modèle.

a. La Perte :

La fonction de perte, également appelée fonction de coût, est l'algorithme qui quantifie l'erreur d'un modèle. La perte est la mesure de cette métrique. Puisque la perte représente l'erreur du modèle, nous souhaitons idéalement qu'elle soit égale à 0.

```
class Loss:
    def calculate(self, output, y):
        sample_losses = self.forward(output, y)
        data_loss = np.mean(sample_losses)
        return data_loss
```

Figure 2.20: Fonction de calcule de perte

Puisque nous avons utilisé la fonction d'activation softmax pour la couche de sortie, (ce qui signifie qu'elle sort une distribution de probabilité). L'entropie croisée catégorielle est explicitement utilisée pour comparer une probabilité "de base" (y ou "cibles") et une certaine distribution prédite (y-hat ou "prédictions"), il est donc logique d'utiliser l'entropie croisée ici. C'est également l'une des fonctions de perte les plus couramment utilisées avec une activation softmax sur la couche de sortie.

L'entropie croisée catégorielle est représentée par L'équation :

$$L_i = -\sum_j y_{i,j} log(\hat{y}_{i,j})$$

Où Li désigne la valeur de perte d'échantillon, i est le i-ème échantillon de l'ensemble, j est l'indice d'étiquette/de sortie, y représente les valeurs cibles, et y-hat représente les valeurs prédites.

```
class Loss_CategoricalCrossentropy(Loss):
    def forward(self, y_pred, y_true):
        samples = len(y_pred)
        y_pred_clipped = np.clip(y_pred, 1e-7, 1 - 1e-7)

        if len(y_true.shape) == 1:
            correct_confidences = y_pred_clipped[range(samples), y_true]
        elif len(y_true.shape) == 2:
            correct_confidences = np.sum(y_pred_clipped * y_true, axis=1)

        negative_log_likelihoods = -np.log(correct_confidences)
        return negative_log_likelihoods
```

Figure 2.21: Fonction de L'entropie croisée catégorielle

b. Précision :

Bien que la perte soit une métrique utile pour optimiser un modèle, la métrique couramment utilisée dans la pratique avec la perte est la précision, qui décrit la fréquence à laquelle la plus grande confiance est la classe correcte en termes de fraction. De manière pratique, nous pouvons réutiliser les définitions de variables existantes pour calculer la métrique de précision. Nous utiliserons les valeurs argmax des sorties de softmax et les comparerons ensuite aux cibles.

2.5.2.4 Rétro propagation :

Afin d'effectuer la rétro propagation, nous allons ajouter une fonction de rétro propagation à chaque classe que nous avons créée.

La rétro propagation est la tâche la plus importante de notre réseau neuronal, et ce, essentiellement parce que c'est la tâche qui va permettre à notre réseau de fonctionner correctement.

Pour cela, nous devons connaître l'impact de chaque poids et biais sur la fonction de perte (calculer combien chaque unique poids et biais change la valeur de perte) et donc le meilleur ensemble de poids et de biais. En outre, la modification et la recherche aléatoires des poids et des biais optimaux ne s'avèrent pas fructueuses. La raison en est que le nombre de combinaisons possibles de poids et de biais est infini et que nous avons besoin de quelque chose de plus intelligent que la chance pure pour obtenir un quelconque succès.

Chaque poids et biais peut également avoir différents degrés d'influence sur la perte, cette influence dépend des paramètres eux-mêmes ainsi que de l'échantillon actuel, qui est une entrée de la première couche. Ces valeurs d'entrée sont ensuite multipliées par les poids, de sorte que les données d'entrée affectent la sortie du neurone et influencent l'impact des poids sur la perte. Le même principe s'applique aux biais et aux paramètres des couches suivantes, qui prennent les sorties de la couche précédente comme entrées.

Cela signifie que l'impact sur les valeurs de sortie dépend des paramètres ainsi que des échantillons, c'est pourquoi nous calculons la valeur de la perte pour chaque échantillon séparément.

La fonction de perte ne contient cependant pas de poids ou de biais. L'entrée de cette fonction est la sortie du modèle, et les poids et biais des neurones influencent cette sortie. Ainsi, même si nous calculons la perte à partir de la sortie du modèle, et non des poids/biais, ces poids et biais ont un impact direct sur la perte.

Notre objectif ici est de diminuer la perte, et nous allons le faire en utilisant la descente de gradient. Le gradient, d'un autre côté, est le résultat du calcul des dérivées

partielles, et nous allons le rétro propager en utilisant la règle de la chaîne pour mettre à jour tous les poids et les biais.

a. Dérivées et dérivées partielles :

Chacune des entrées de la fonction a un certain impact sur la sortie de cette fonction, même si cet impact est nul. Nous devons connaître ces impacts, cela signifie que nous devons calculer la dérivée par rapport à chaque entrée séparément pour en savoir plus sur chacune d'entre elles. C'est pourquoi nous appelons cela des dérivées partielles par rapport à une entrée donnée - nous calculons une partie de la dérivée, liée à une entrée particulière.

La dérivée partielle mesure l'impact d'une seule entrée sur la sortie d'une fonction. La méthode de calcul d'une dérivée partielle est la même que pour les dérivées, nous devons simplement répéter ce processus pour chacune des entrées indépendantes.

La dérivée partielle d'une somme :

$$f(x,y) = x + y \quad \rightarrow \quad \frac{\partial}{\partial x} f(x,y) = \frac{\partial}{\partial x}[x+y] = \frac{\partial}{\partial x}x + \frac{\partial}{\partial x}y = 1 + 0 = 1$$
$$\frac{\partial}{\partial y} f(x,y) = \frac{\partial}{\partial y}[x+y] = \frac{\partial}{\partial y}x + \frac{\partial}{\partial y}y = 0 + 1 = 1$$

La dérivée partielle de la multiplication :

$$f(x,y) = x \cdot y \quad \rightarrow \quad \frac{\partial}{\partial x} f(x,y) = \frac{\partial}{\partial x}[x \cdot y] = y\frac{\partial}{\partial x}x = y \cdot 1 = y$$
$$\frac{\partial}{\partial y} f(x,y) = \frac{\partial}{\partial y}[x \cdot y] = x\frac{\partial}{\partial y}y = x \cdot 1 = x$$

La dérivée partielle de Max :

$$f(x,y) = max(x,y) \quad \rightarrow \quad \frac{\partial}{\partial x} f(x,y) = \frac{\partial}{\partial x} max(x,y) = 1(x > y)$$

```
def backward(self, dvalues):

    self.dweights = np.dot(self.inputs.T, dvalues)
    self.dbiases = np.sum(dvalues, axis=0, keepdims=True)
    self.dinputs = np.dot(dvalues, self.weights.T)
```

Figure 2.22: Function de retro propagation

b. Gradient :

La dérivée partielle est une équation unique, et la dérivée de la fonction multi variable complète qui consiste en un ensemble d'équations appelé le gradient. En d'autres termes, le gradient est un vecteur de la taille des entrées contenant les solutions de la dérivée partielle par rapport à chacune des entrées.

Le gradient est un vecteur composé de toutes les dérivées partielles d'une fonction, calculées par rapport à chaque variable d'entrée.

c. Descente de gradient et règle de la chaîne :

Nous utiliserons les dérivées des fonctions à un seul paramètre et les gradients des fonctions multivariées pour effectuer la descente de gradient à l'aide de la règle de la chaîne ou, en d'autres termes, pour effectuer la rétro propagation, qui fait partie de l'apprentissage du modèle.

La règle de la chaîne : Le passage à travers notre modèle est une chaîne de fonction. Nous transmettons des échantillons, les données traversent toutes les couches et les fonctions d'activation pour former une sortie.

```python
def backward(self, dvalues):

    self.dinputs = dvalues.copy()
    self.dinputs[self.inputs <= 0] = 0
```

Figure 2.23 : Dérivé de RelU

```python
def backward(self, dvalues):

    self.dinputs = np.empty_like(dvalues)

    for index, (single_output, single_dvalues) in enumerate(zip(self.output, dvalues)):

        single_output = single_output.reshape(-1, 1)

        jacobian_matrix = np.diagflat(single_output) - np.dot(single_output, single_output.T)

        self.dinputs[index] = np.dot(jacobian_matrix, single_dvalues)
```

Figure 2.24 : Dérive de Softmax

```python
def backward(self, dvalues, y_true):

    samples = len(dvalues)

    labels = len(dvalues[0])
    if len(y_true.shape) == 1:
        y_true = np.eye(labels)[y_true]

    self.dinputs = -y_true / dvalues

    self.dinputs = self.dinputs / samples
```

Figure 2.25 : Dérive de L'entropie croisée catégorielle

d. Optimiseur Stochastic Gradient Descent SGD:

L'optimiseur est la classe qui va modifier les valeurs de nos poids et biais en fonction du taux d'apprentissage que nous avons fixé et des résultats que nous avons obtenus lors de notre passage en arrière.

```
class Optimizer_SGD:

    def __init__(self, learning_rate=1.0):
        self.learning_rate = learning_rate

    def update_params(self, layer):
        layer.weights += - self.learning_rate * layer.dweights
        layer.biases += - self.learning_rate * layer.dbiases
```

Figure 2.26 : Optimiseur SGD

2.6 Conclusion

Dans ce chapitre on a présenté le prétraitement d'images, l'extraction des caractéristiques ainsi que les réseaux de neurones multicouches qui a été l'objectif de ce chapitre, où on a acquis des connaissances sur ce dernier, tout en faisant un passage sur les méthodes d'apprentissage, où l'apprentissage supervisé semble le plus adéquat comme classificateur pour la reconnaissance des caractères.

3 Chapitre 3 : Implémentation ET Résultats

3.1 Introduction

Afin de réaliser notre travail, nous avons eu recours à plusieurs outils de développement notamment la bibliothèque de vision par ordinateur Open CV 3.4, le langage de programmation python version 3.9 et le module PyQt5, dans ce présent chapitre nous allons présenter les différentes bibliothèques et modules auxquels nous avons fait appel afin de concevoir un système la lecture automatique de plaques d'immatriculation en temps réel et qui nous ont permis de gagner un temps considérable.

3.2 Modules et outils de développement

3.2.1 OpenCV (Open Source Computer Vision Library):

OpenCV (Open Source Computer Vision Library) a été initialement développée par Intel en 1999 par Gary Bradsky, et la première version est sortie en 2000. Vadim Pisarevsky a rejoint Gary Bradsky pour gérer l'équipe russe de logiciels OpenCV d'Intel. En 2005, OpenCV a été utilisé sur Stanley, le véhicule qui a remporté le DARPA Grand Challenge 2005. Plus tard, son développement actif s'est poursuivi avec le soutien de Willow Garage avec Gary Bradsky et Vadim Pisarevsky à la tête du projet. OpenCV prend désormais en charge une multitude d'algorithmes liés à la vision par ordinateur et à l'apprentissage automatique et se développe de jour en jour. [22]

OpenCV prend en charge une grande variété de langages de programmation tels que C++, Python, Java, etc., et est disponible sur différentes plates-formes, notamment Windows, Linux, OS X, Android et iOS. Des interfaces pour les opérations GPU à haute vitesse basées sur CUDA et OpenCL sont également en cours de développement. [22]

OpenCV est une bibliothèque open source qui comprend plusieurs centaines d'algorithmes de vision par ordinateur. Le document décrit la soi-disant API OpenCV 2.x, qui est essentiellement une API C++, par opposition à l'API OpenCV 1.x basée sur C (l'API C est obsolète et n'est pas testée avec le compilateur "C" depuis les versions OpenCV 2.4). [23]

OpenCV-Python est l'API Python pour OpenCV, combinant les meilleures qualités de l'API OpenCV C++ et du langage Python. [22]

OpenCV a une structure modulaire, ce qui signifie que le package comprend plusieurs bibliothèques partagées ou statiques. Les modules suivants sont disponibles :

- Fonctionnalité de base (core) - un module compact définissant les structures de données de base, y compris le tableau multidimensionnel dense Mat et les fonctions de base utilisées par tous les autres modules.

- Traitement d'image (imgproc) - un module de traitement d'image qui inclut un filtrage d'image linéaire et non linéaire, des transformations d'image géométriques (redimensionnement, déformation affine et perspective, remappage générique basé sur une table), conversion d'espace colorimétrique, histogrammes, etc.

- Analyse vidéo (vidéo) - un module d'analyse vidéo qui comprend des algorithmes d'estimation de mouvement, de soustraction d'arrière-plan et de suivi d'objet.

- ... ainsi que de nombreux autres modules que nous n'avons pas mentionnés. [23]

3.2.2 PyQt:

PyQt connecte le framework multiplateforme Qt C++ avec le langage Python, c'est un module GUI.

Qt est écrite en C++ et elle est, à la base, conçue pour être utilisée en C++. Toutefois, il est aujourd'hui possible de l'utiliser avec d'autres langages comme Java, Python, etc.

Qt est plus qu'une boîte à outils d'interface graphique, c'est pourquoi il propose des abstractions de sockets ou de threads réseau, ainsi que Unicode, SQL, des bases de données, SVG, OpenGL, XML, un navigateur Web opérationnel, un système de service et une vaste gamme de widgets GUI.

Le principe sur lequel fonctionne une classe Qt est lié à un mécanisme de slot chargé d'offrir une communication entre les éléments dans le but de concevoir facilement des composants logiciels réutilisables.

De plus, Qt est livré avec Qt Designer, un outil qui agit comme une interface utilisateur graphique. PyQt peut concevoir du code Python à partir de Qt Designer, tout en ajoutant de nouveaux contrôles GUI lorsque Qt Designer et le langage de programmation Python sont utilisés. [24]

3.2.3 Python:

Python a été créé au début des années 1990 par Guido van Rossum au Stichting Mathematisch Centrum (CWI, voir https://www.cwi.nl/) aux Pays-Bas en tant que successeur d'un langage appelé ABC. Guido reste l'auteur principal de Python, bien qu'il inclue de nombreuses contributions d'autres.

En 1995, Guido a poursuivi ses travaux sur Python à la Corporation for National Research Initiatives (CNRI, voir https://www.cnri.reston.va.us/) à Reston, en Virginie, où il a publié plusieurs versions du logiciel.

En mai 2000, Guido et l'équipe de développement de base Python ont déménagé sur BeOpen.com pour former l'équipe BeOpen PythonLabs. En octobre de la même année, l'équipe PythonLabs a déménagé à Digital Creations (maintenant Zope Corporation ; voir https://www.zope.org/). En 2001, la Python Software Foundation (PSF, voir https://www.python.org/psf/) a été formée, une organisation à but non lucratif créée spécifiquement pour posséder la propriété intellectuelle liée à Python. Zope Corporation est un membre parrain de la PSF. [25]

Python est un langage de programmation interprété, orienté objet, de haut niveau avec une sémantique dynamique. Ses structures de données intégrées de haut niveau, combinées à un typage dynamique et à une liaison dynamique, le rendent très attrayant pour le développement rapide d'applications, ainsi que pour une utilisation comme langage de script ou de collage pour connecter des composants existants entre eux. La syntaxe simple et facile à apprendre de Python met l'accent sur la lisibilité et réduit donc le coût de maintenance du programme. Python prend en charge les modules et les packages, ce qui encourage la modularité du programme et la réutilisation du code. L'interpréteur Python et la vaste bibliothèque standard sont disponibles sous forme source

ou binaire sans frais pour toutes les principales plates-formes et peuvent être distribués gratuitement. [26]

Dans notre travail, nous avons vu et utilisé plusieurs bibliothèques et modules python, nous allons les passer en revue dans ce qui suit.

3.2.3.1 Numpy:

NumPy est une extension du langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux.[27] Elle fournit de multiples fonctions permettant notamment de créer directement un tableau depuis un fichier ou au contraire de sauvegarder un tableau dans un fichier, et manipuler des vecteurs, matrices.

3.2.3.2 La bibliothèque standard Math:

Pour disposer des fonctions mathématiques usuelles, la librairie d'origine du python se nomme math. On peut alors d'importer juste les fonctions nécessaires par from math import cos, log ou toutes les fonctions mathématiques par from math import *. Dans le premier cas l'inconvénient est qu'il faut savoir à l'avance les fonctions utilisées par la suite, dans le deuxième cas on risque de surcharger inutilement la mémoire.

3.2.4 Outils:

3.2.4.1 Pycharm :

PyCharm est un environnement de développement intégré (abrégé EDI en français ou en anglais : IDE (Integrated Development Environment)) utilisé pour programmer en Python.

3.2.4.2 Qt Designer :

Qt Designer est l'outil Qt pour la conception et la construction d'interfaces utilisateur graphiques (GUI) avec Qt Widgets. Vous pouvez composer et personnaliser vos fenêtres ou boîtes de dialogue de manière que ce que vous voyez est ce que vous obtenez (what-you-see-is-what-you-get WYSIWYG) et les tester à l'aide de différents styles et résolutions.

Les widgets et les formulaires créés avec Qt Designer s'intègrent de manière transparente au code programmé, en utilisant le mécanisme de signaux et de slots de Qt, afin que vous puissiez facilement attribuer un comportement aux éléments graphiques. Toutes les propriétés définies dans Qt Designer peuvent être modifiées dynamiquement dans le code. De plus, des fonctionnalités telles que la promotion de widgets et les plugins personnalisés vous permettent d'utiliser vos propres composants avec Qt Designer. [28]

3.3 Présentation de l’approche suivie

Pour arriver à notre objectif qui est le développement d’un programme de reconnaissance de plaques d’immatriculation en temps réel nous avons travaillé sur des images, et comme une vidéo n’est qu’une succession d’images on a qu’à faire boucler notre programme sur chaque image de la vidéo. Dans ce qui suit on va décrire notre travail sur 3 étapes : Acquisition et prétraitement d’images, extraction des caractéristiques, reconnaissance des caractères tout en montrant les résultats de chaque étape.

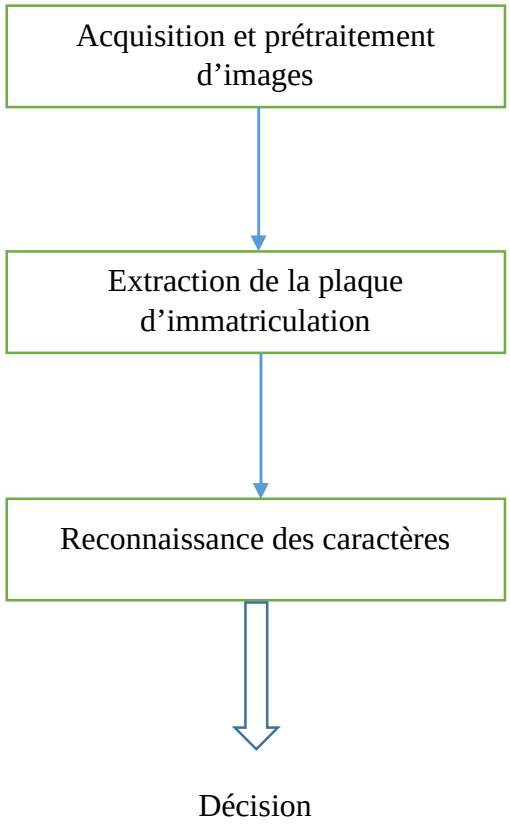


Figure 3.1 : Schéma de reconnaissance de plaque d’immatriculation de véhicule.

3.3.1 Acquisition et prétraitement d’images :

Avant de se lancer dans le développement du programme l’acquisition d’images que nous allons utiliser tout au long de notre parcours est nécessaire c’est pour ça que nous avons dû créer notre propre Dataset de plus de 500 images d’une résolution de 720x540 pixel pour apprentissage et les tests.la figure ci-dessous montre un exemple des images collecter

Après la création de notre Dataset nous passons chaque image au traitement en lui appliquant diffèrent modules (filtrage, segmentation, binarisation...), pour essayer de trouver la meilleure combinaison de modules et ainsi les optimiser dans le but de faciliter l’extraction et la reconnaissance des caractères dans ce qui suit nous présentons les différents modules qu’on a choisi.

3.3.1.1 Acquisition d’images :

Pour l’acquisition d’images on a simplement utilisé un appareil photo de 12 méga pixel d’une taille de 2880×2160p dont on a fait une réduction jusqu’à 720×540p de taille d’image :

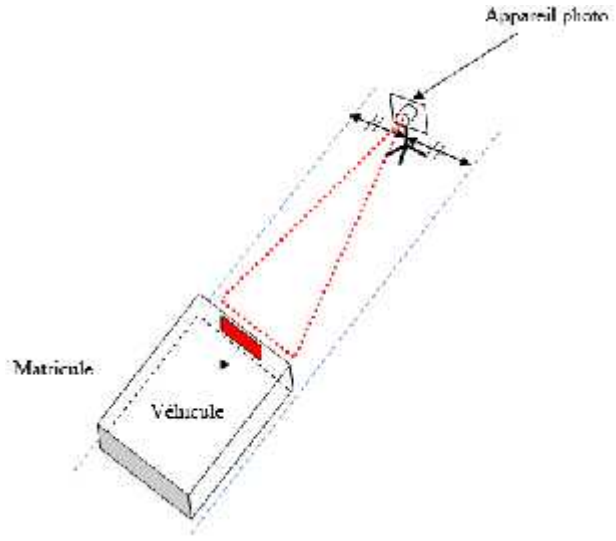


Figure 3.2 : Calibrage de l’appareil photo.

Figure 3.3 : Exemples d'acquisition d'images

3.3.1.2 Segmentation d'images :

Pour commencer on a segmenté l'image pour nous débarrasser des couleurs qui nous paraissent inutiles, vu que le matricule que nous voulons extraire n'a que 2 choix de couleurs (noir sur blanc / noir sur jaune) la figure 3.4 ci-dessous montre le résultat :

Figure 3.4 : Segmentation de l'image

3.3.1.3 Filtrage :

Chaque image a ses ensembles de bruits et le filtrage consiste à réduire voire supprimer ses bruits sans importance et faire apparaitre les caractéristiques significatives de l'image pour cela nous avons opter à utiliser un filtre bilatéral qui donne de très bons résultats.

Figure 3.5 : Application du filtre bilatéral

3.3.1.4 Conversion en gris :

Après la réduction du bruit, on convertit notre image en niveau de gris pour simplifier les algorithmes à utiliser et réduire le temps de computation le résultat est illustré dans la figure :

Figure 3.6 : Conversion en gris

3.3.1.5 Binarisation :

Pour finir notre phase de prétraitement d'images on binarize l'image où on partage les pixels de l'images en deux classes par rapport à un seuil, toujours pour réduire la quantité d'informations véhiculé par l'image et ne garder que les informations pertinentes la figure ci-dessous montre le résultat obtenu :

Figure 3.7 : image binaire

3.3.2 Extraction des caractéristiques :

Dans cette deuxième partie nous allons appliquer les algorithmes de détection de contour et lignes pour extraire les caractères de la plaque d'immatriculation ainsi que ses caractères en se basant sur les caractéristiques d'une plaque d'immatriculation algérienne :

3.3.2.1 Détection des contours :

Dans cette étape on identifie tous les contours présents dans notre image, pour ce faire on utilise la méthode cv2.findContours() d'OpenCV qui prend en paramètre une image binaire inversé présenté dans la figure à gauche, et nous retourne la figure à droite :

Figure 3.8 : detection des contours

On repère ensuite les contours en fonction de la hauteur qui est entre 20px jusqu'à 60px et de la largeur qui est entre 4px jusqu'à 30px ce qui nous permet d'éliminer les contours qui n'ont pas les caractéristiques d'un caractère puis on utilise la fonction cv2.boundingRect() d'OpenCV pour entourer chaque contour par un rectangle et on fixe un point au centre le résultat obtenu est présenté dans la figure 3.9 :

Figure 3.9 : Contours obtenus

3.3.2.2 Détection des lignes par la transformé de Hough :

Cette étape consiste à trouver une ligne qui passe par les caractères de notre plaque d'immatriculation, on se sert alors des centres des rectangle qu'on a dessiné précédemment (représenté par des points blanc sur la figure 3.9) pour optimiser nos résultats on implémente alors la transformé de Hough on lui introduisant différentes conditions, la ligne doit passer par 8 points au minimum ainsi qu'un écart de 25pixels au maximum entre chaque point et une longueur maximale de la ligne qui ne doit pas dépasser 200pixels le résultat est présenté dans la figure 3.10 :

Figure 3.10 : Détection de la ligne

3.3.2.3 Calcul de la distance :

Après avoir tracé notre ligne on calcule la distance entre chaque centre de contour et la ligne de la plaque d'immatriculation et on élimine tous contour qui n'est pas contenue dans la plaque d'immatriculation, la figure 3.11 montre le résultat obtenu (les contours qui ont été acceptés en vert et ceux qui ont été éliminé en rouge) :

Figure 3.11 : exemple 1 Resultat des contour acceptés

Figure 3.12 : la plaque détectée

Figure 3.13 : exemple 2 Resultat des contour acceptés

Figure 3.14 : la plaque détectée

Comme l'affiche la figure 3.13 et la figure 3.14 on a un contour qui a était acceptés mais qui ne représente pas un caractère pour remédier à ce problème, On sait que notre plaque est composée de trois groupe de numéro le 1er se compose de 5 à 6 caractères, le 2eme de trois, et le dernier de deux, on part de ce principe et on prend les trois derniers contours (« 1 » « 6 » « x ») de la plaque à droite, la distance entre le 1er contour et le 2eme doit être supérieure à celle entre le 2eme et 3 troisième contour, si elles sont approximativement identiques alors on a un intrus car le troisième groupe ne contient que 2 caractères on supprime donc le dernier caractère à droite.

Figure 3.15 : résultat après les calculs de distance point-point

3.3.3 Reconnaissance des caractères :

Dans cette dernière partie qui est la reconnaissance de caractères on se sert de notre modèle de perceptron multi couche pour classifier les caractères extrait, mais d'abord on doit normaliser ses derniers pour que notre MLP puisse interpréter l'image injectée, dans ce qui suit on expliquera ce procédé

3.3.3.1 Découpage :

Cette étape est très basique, après avoir identifié les caractères de la plaque dans l'étape précédente on prend leurs coordonnés sur le plan de l'image on les découpe pour la prochaine étape de normalisation ce qui nous donne en résultat la figure :

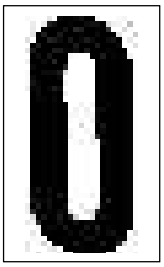


Figure 3.16 : Caractère avant normalisation (21x45p)

3.3.3.2 Normalisation :

Pour que notre perceptron puisse interpréter l'image et classifier les caractères, chaque image doit être normalisé, nous avons choisi comme standard une taille de 40×40 pixels, pour ce faire tout en respectant l'aspect ratio et en évitant de déformer le caractère, on prend le max(hauteur, largeur) de l'image, on le divise par notre taille désirée et on multiplie le résultat par la hauteur et la largeur de

l’image puis on remplit les bords vide restant par des pixels blanc, ce processus est expliqué dans la figure :

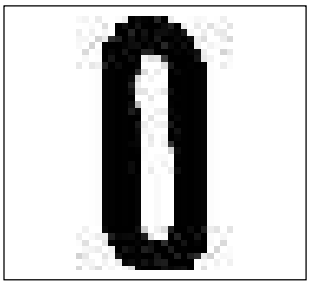


Figure 3.17 : le caractère après normalisation (40x40p)

3.3.3.3 Classification :

a. Réseau de neurones :

Comme c’est indiqué dans le chapitre II, le réseau de neurones est composé de trois couches :

- La couche d’entrée.
- La couche cachée.
- La couche de sortie.

- **Couche d’entrée :**

L’image normalisé est de taille 40×40p ce qui nous donne un totale de 1600 pixels ce qui implique 1600 neurones d’entrée.

La valeur de chaque neurone est égale à la valeur de chaque pixel correspondant 0 ou 1 (0 pixel noir, 1 pixel blanc)

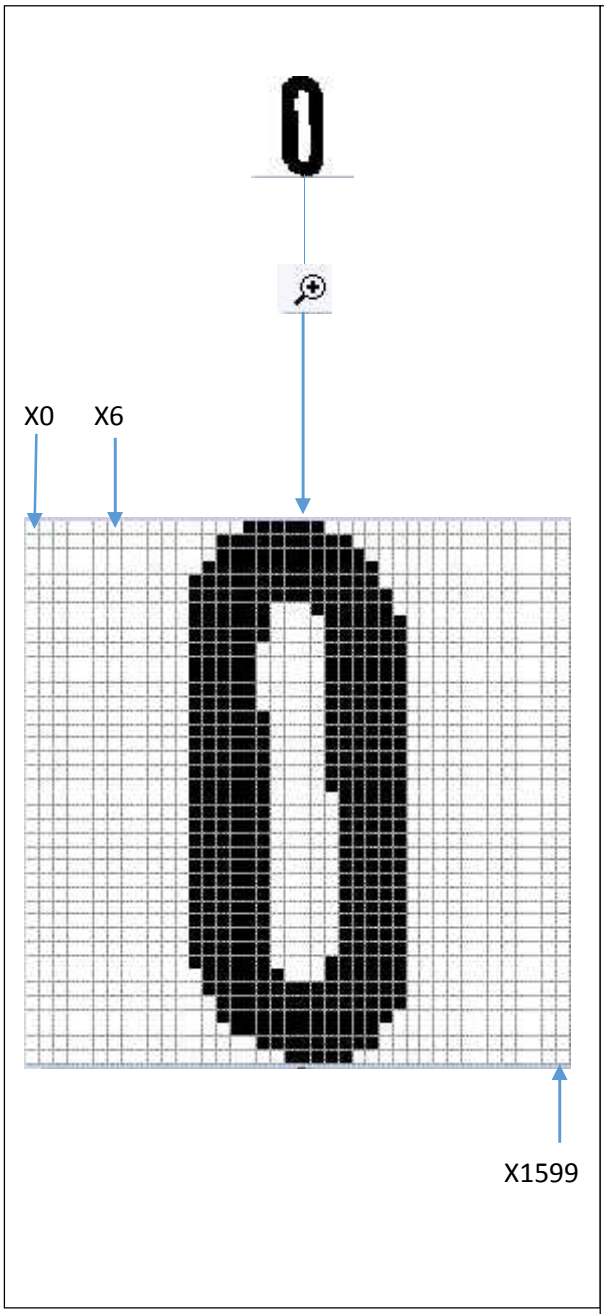


Figure 3.18: Représentation d'un caractère sous forme de pixels.

Xi	Valeur		Xi	Valeur
0	1		41	1
1	1		42	1
2	1		43	1
3	1		44	1
4	1		45	1
5	1		46	1
6	1		47	1
7	1		48	1
8	1		49	1
9	1		50	1
10	1		.	.
11	1		.	.
12	1		.	.
13	1		.	.
14	1		.	.
15	1		1573	0
16	0		1574	1
17	0		1575	0
18	0		1576	1
19	0		1577	1
20	0		1578	1
21	0		1579	0
22	0		1580	0
23	1		1581	0
24	1		1582	0
25	0		1583	0
26	0		1584	1
27	1		1585	1
28	0		1586	0
29	1		1587	1
30	1		1588	1
31	1		1589	1
32	1		1590	1
33	1		1591	1
34	1		1592	1
35	1		1593	1
36	1		1594	1
37	1		1595	1
38	1		1596	1
39	1		1597	1
40	1		1598	1
			1599	1

Figure 3.19 : Valeurs des neurones d'entrée pour le caractère représenté par la Figure 3.18

- **Couche cachée :**

Un neurone de cette couche (il existe 100 neurones) correspond à 1600 coefficients différents, à chaque neurone d'entrée est attribué un coefficient. On calcul la somme pondérée à l'entrée de chaque neurone caché ainsi que la valeur de sa sortie.

- **Couche de sortie :**

Le nombre de neurones de cette couche est égale au nombre de classes choisies (dix classes différentes), les valeurs de sortie sont comprises entre 0 et 1 et la sortie désignée par le système correspond à la classe du neurone qui donne la valeur maximale.

Exemple :

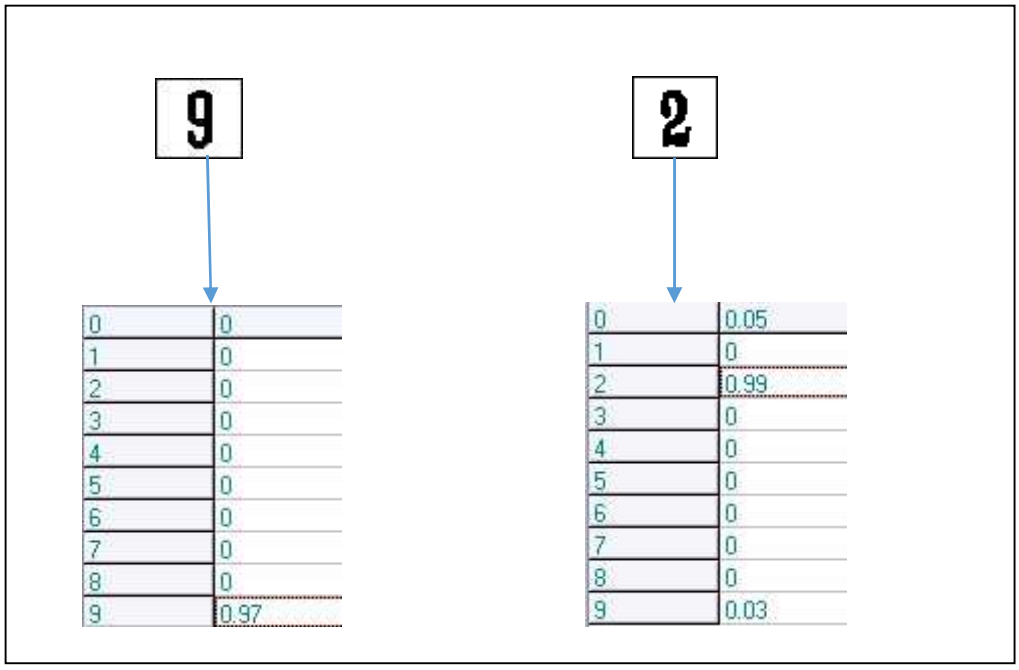


Figure 3.20 : Reconnaissance de caractère par PMC.

3.4 Résultats obtenus :

Pour pouvoir effectuer l'apprentissage et le test de notre PMC nous avons dû créer une dataset de plus de 7300 images de caractères (la figure 3.24 illustre des exemples de chaque classe) qu'on a ensuite divisées en deux parties,

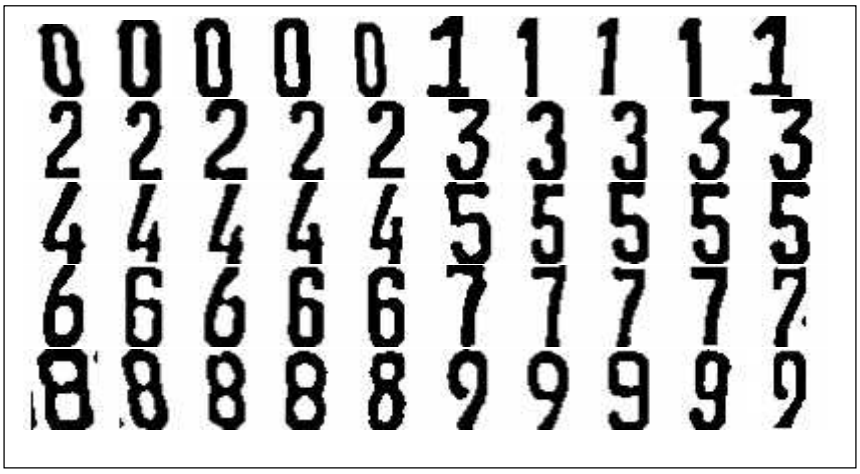


Figure 3.21 : exemples de caractères.

La première dédiée à l'apprentissage « train » de 500 images par classe pour un totale de 5000 images, la deuxième dédie au « test » avec plus de 2300 images.

- **Train :** la précision de reconnaissance des caractères est de 100%.
- **Test :** la précision de reconnaissance des caractères est de 99.16%.

En ce qui concerne la détection des plaques d'immatriculation, on a créé une dataset de test de plus de 130 images de véhicules (la figure 3.25 illustre des exemples de notre dataset) sur qui on est arrivés à un taux de reconnaissance de 93%.

Figure 3.22 : exemples des images de la dataset.

Le système réalisé dans ce PFE a montré de bonnes performances. Cependant il y a quelques points qui nécessitent des améliorations que nous allons présenter dans les perspectives.

Perspectives :

Dans le but d'améliorer notre système ANPR et le rendre plus fiable, nous pensons à :

- L’utilisation des techniques de prétraitements plus robustes pour minimiser le temps d’exécution du programme et avoir des résultats de traitement plus propres.
- L’adaptation du système pour une utilisation durant la nuit (éclairage automatique et ajustable).

3.5 Présentation de l’interface logiciel :

Figure 3.23 : Login

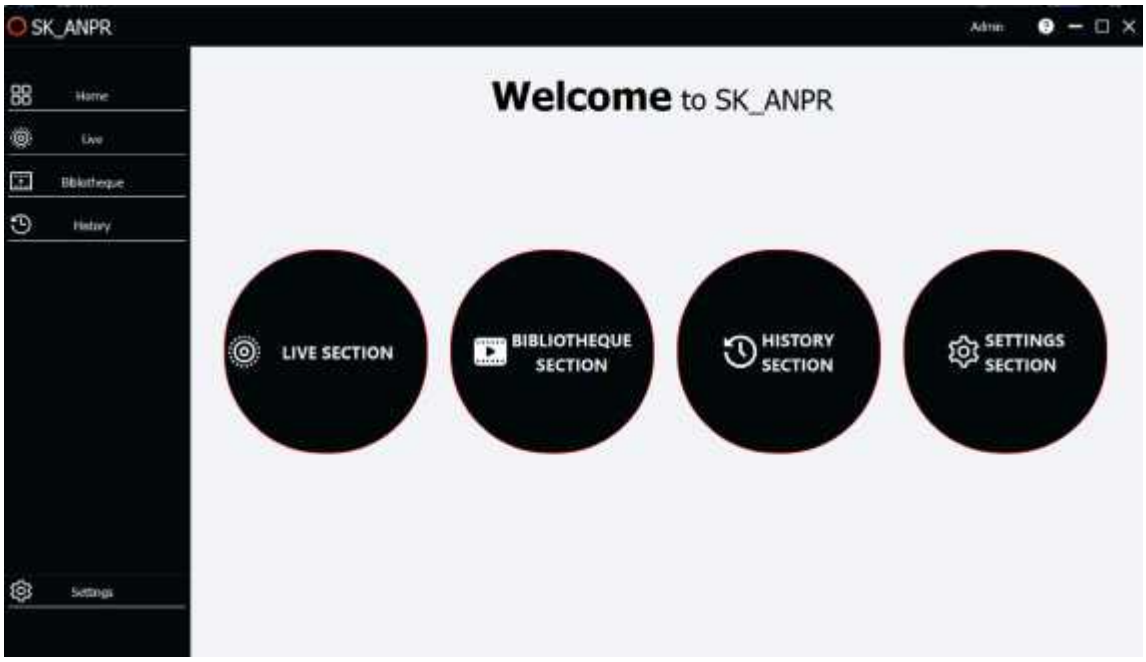


Figure 3.24 : Home page

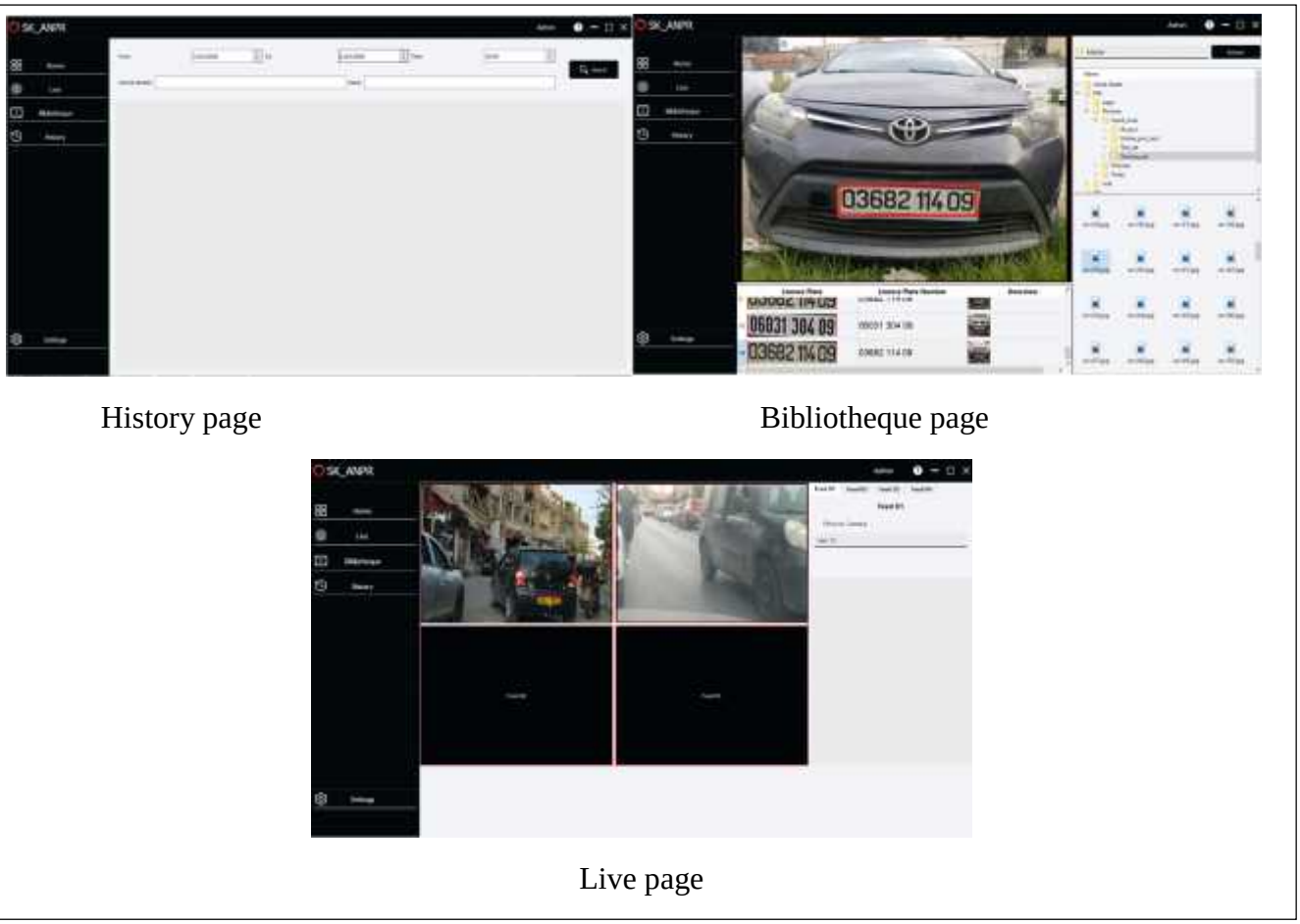


Figure 3.25 : les différentes sections du programme.

Lors de la création de l'interface graphique de notre logiciel notre objectif était de créer une interface « user-friendly » tout en offrant aussi une multitude de fonctionnalité.

La base de données permet au programme de sauvegarder toute reconnaissance effectuée, ainsi que les différents types utilisateurs connectés (administrateur ou simple utilisateur).

Après l'authentification de l'utilisateur, une page d'accueil (figure 3.20) comprenant les 4 différentes sections s'affiche :

Bibliothèque : ou il sera capable de parcourir les fichiers et sélectionner des images ou vidéos pour effectuer une reconnaissance.

Live : ici l'utilisateur pourra choisir le flux vidéo en direct pour la reconnaissance.

History : cette section permet de consulter l'historique des matricules déjà reconnus.

Settings : dans cette section contient les différents paramètres du programme.

3.6 Conclusion

Dans ce chapitre nous avons présente le résultat de tout notre travail ainsi que les méthodes que nous avons choisies et que ce soit dans le prétraitement avec le filtre bilatéral ou avec la binarisation locale, ainsi que les méthodes employées pour l'extraction des caractéristiques et en fin les résultats de la reconnaissance et la classification

Conclusion générale :

Le travail que nous avons effectués dans ce mémoire ou nous avons développé un logiciel qui permet la reconnaissance automatique de plaques d'immatriculation en temps réel, où on ses basé sur les étapes de prétraitements et d'extraction des caractéristiques afin d'améliorer et réduire les bruits et aussi le poids de l'image, afin de minimiser le temps d'exécution, pour une bonne reconnaissance où, on est arrivé à un taux de reconnaissance de plus de 90% avec un temps d'exécution de moins d'une demi second.

Le système conçu est d'une grande importance pour la gestion automatique des parkings et les entrées des établissements, où on gagne un temps considérable pour la fluidité des entrées et sortie.

Nous espérons que ce modeste travail peut servir de base de départ à d'autres projets éventuels dans le cadre de développement et l'amélioration de ce projet.

Bibliographies

[1] "CCTV network tracks 'getaway' car". BBC News. 21 November 2005. Retrieved 12 August 2013.]

[2] Guemmini Mourad, Filtrage Bilatérale Appliqué Sur Des Images Bruitées 2013

[3] Transformée de Hough. (2020, avril 20). Wikipédia, l'encyclopédie libre. http://fr.wikipedia.org/w/index.php?title=Transform%C3%A9e_de_Hough&oldid=169812 95.

[4] Socret Lee, Lines Detection with Hough Transform (May 2, 2020) https://towardsdatascience.com/lines-detection-with-hough-transform-84020b3b1549

[5] “History of ANPR". ANPR International. Retrieved 18 December 2019.

[6] “History of ANPR". ANPR International. Retrieved 26 October 2020

[7] http://www.map.toulouse.archi.fr/works/panoformation/imagenum/imagenum.htm

[8] Image numérique. (2021, septembre 20). Wikipédia, l'encyclopédie libre. http://fr.wikipedia.org/w/index.php?title=Image_num%C3%A9rique&oldid=186473566 .

[9] Jean-marie, La liaison automatique des plusieurs images perçues sur un scanner 2008

[10] Reconnaissance optique de caractères. (2021, mars 6). Wikipédia, l'encyclopédie libre. http://fr.wikipedia.org/w/index.php?title=Reconnaissance_optique_de_caract%C3%A8r es&oldid=180608413.

[11] Ros Definir.tech, https://definir-tech.com/reconnaissance-optique-de-caracteres-ocr/

[12] Pr. B. ALAIN, Anh. Tuan. NGHIEM : "Reconnaissance d'écriture manuscrite", 15 juillet

2005.

[13] Distance (mathématiques). (2021, mai 16). Wikipédia, l'encyclopédie libre. http://fr.wikipedia.org/w/index.php?title=Distance_(math%C3%A9matiques)&oldid=182955475.

[14] Ould Mohamed et Mohamed Mostapha, Methodes de regression pour la surveillance des eaux propres 2016

[15] Cours Rafic Younes 2005. « Chapitre 3 RN. Pdf ». http://www.ryounes.net/cours/

[16] Hatti, Mustapha & Mustapha, Tioursi & Zakia, Benrabia. (2012). Les Réseaux de Neurones dans les Systèmes à Pile à Combustible.

[17] Système nerveux. (2021, juillet 5). Wikipédia, l'encyclopédie libre. http://fr.wikipedia.org/w/index.php?title=Syst%C3%A8me_nerveux&oldid=184398248.

[18] Neurosciences - A la découverte du cerveau. Mark-F Bear, Barry W. Connors, Michael A. Paradiso. PRADEL (EDITIONS)

[19] Mina Kalakh, modélisation avec les réseaux de neurones d'un canal uvb dans un environnement minier souterrain 2013

[20] Mémoire de PFE ingénieur, réalisé par lourci med amine et bencherchali med, "reconnaissance de plaques d'immatriculation", année 2007/2008.

[21] M. Schyns Les réseaux de neurones : principes et application à la d´détection financière des faillites 1997

[22] Introduction to OpenCV-Python Tutorials https://docs.opencv.org/3.4/d0/de3/tutorial_py_intro.html

[23] introduction to OpenCV https://docs.opencv.org/4.5.0/d1/dfb/intro.html

[24] What is PyQt? https://pythonpyqt.com/what-is-pyqt/ 2020

[25] History and License, https://docs.python.org/3/license.html

[26] What is Python? Executive Summary, https://www.python.org/doc/essays/blurb/

[27] NumPy. (2021, août 24). Wikipédia, l'encyclopédie libre http://fr.wikipedia.org/w/index.php?title=NumPy&oldid=185764276.

[28] Qt Designer Manual, https://doc.qt.io/qt-5/qtdesigner-manual.html

Résumé

Le but de ce mémoire est de concevoir et réaliser un logiciel ANPR hors ligne, capable de localiser et de lire la plaque d'immatriculation en temps réel et pour étendre la capacité des logiciels ANPR. Le développement du logiciel ANPR comprend plusieurs étapes de traitement: acquisition d'images, prétraitement d'images, extraction de plaques d'immatriculation, segmentation de caractères et reconnaissance de caractères. Plusieurs tests ont été effectués afin de mesurer les performances et les capacités du logiciel ANPR développé. Sur la base de nos résultats, le système est fiable et robuste avec un taux moyen de plus de 80% et moins d'une demi-seconde de temps de réponse

Mots clés : Reconnaissance Optique de Caractères (ROC), Reconnaissance Automatique de Plaques d'immatriculation (RAPI), Intelligence Artificielle (IA), Réseaux de Neurones (RN), Transformée de Hough

Abstract

The aim of this research is to conceive and realize an offline ANPR software, which can locate and read the number plate in real-time, and to extend the capability of the developed ANPR software. The development of the ANPR software consists of several processing steps: image acquisition, image pre-processing, number plate extraction, character segmentation, and character recognition. Several tests were done in order to measure the performance and capability of the developed ANPR software. Based on the results, the system is reliable and robust with an average rate of more than 80% and less than half a second of response time.

Key words: Optical Character Recognition (OCR), Automatique Number Plate Recognition (ANPR), Artificial Intelligence (AI), Neural Networks (NN), Hough Transform

ملخص

هناك العديد من التطبيقات المفيدة لأنظمة التعرف التلقائي على لوحة الأرقام (ANPR) مثل تطبيق قانون المرور ، وتحصيل رسوم المرور ، وقياس وقت الرحلة ، وإدارة نظام وقوف السيارات. من بين العديد من التقنيات المتقدمة الأخرى لإدارة نظام وقوف السيارات ، اكتسب نظام ANPR الكثير من النوايا لأنه نهج غير تدخلي ولا يتطلب هذا النظام أي تعريف إضافي للمركبة ليتم تثبيته في السيارة. الهدف من هذا البحث هو تطوير برنامج ANPR غير المتصل بالإنترنت ، والذي يمكنه تحديد موقع لوحة الأرقام وقراءتها في الوقت الفعلي، ولتوسيع قدرة المطور برنامج ANPR. لقد اخترنا إنشاء شبكتنا العصبية الخاصة من الصفر ، ولهذا السبب ، كان لابد من إنشاء مجموعة بيانات ضخمة من صور السيارات التي تناسب بيئة المرور في الجزائر من الصفر. يتكون تطوير برنامج ANPR من عدة خطوات معالجة: الحصول على الصور و المعالجة المسبقة للصور واستخراج لوحة الأرقام وتجزئة الأحرف والتعرف على الحروف. تم إجراء العديد من الاختبارات من أجل قياس أداء وقدرة برنامج ANPR المطور. بناءً على النتائج ، وجدنا النظام موثوقًا وقويًا بمعدل متوسط يزيد عن 80٪ وأقل من نصف ثانية من وقت الاستجابة.

الكلمات المفاتيح: التعرف البصري على الأحرف، التعرف التلقائي على لوحة الأرقام، الذكاء الاصطناعي، الشبكات العصبية