REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie
Département d'informatique

N° d'ordre : SIOD /M2/2020

Mémoire

Présenté pour obtenir le diplôme de master académique en

Informatique

Parcours : **SIOD**

Reconnaissant automatique des fruits à partir d'images naturelles en utilisant l'apprentissage automatique

---

Par :
Abdelatif Chamekh

Soutenu le ../09/2020, devant le jury composé de :

Nom et prénom	Grade	Président
Ben Seghier Nadia	MCA	Rapporteur
Nom et prénom	Grade	Examinateur

Remerciements

Tout d'abord, Louons "ALLAH" qui nous a doté de la merveilleuse faculté de raisonnement. Je tiens à exprimer mes remerciements à mon superviseur Mme Ben Seghier Nadia De m'avoir soutenu et fait confiance pendant mon projet avec une grande patience. Mes remerciements et ma plus profonde gratitude s'adressent à mon parents qui n'ont fait preuve que de soutien et d'amour, et aussi à tous les membres de ma famille. En fin de compte, nous remercier toutes les personnes qui ont participé de près ou de loin à la la réalisation de ce travail.

Dédicace

Je dédie ce modeste travail

A mes très chers parents pour leur soutient et encouragement
durant toutes mes années d'études et sans lesquels je n'aurais jamais réussi, A mes frères et mes soeurs et à toute ma famille. A tous mes amis surto mon ami et frère Chihab Eddine Mechkouri, qui m'ont aidées, soutenues et encouragées. A tous mes enseignants durant mes années d'études avec lesquels j'ai beaucoup appris.

résumé

Identifier les formes quelles que soient leur type (visages, voitures, fruits, bâtiments) est une tâche facile pour les humains.Est-ce Pareil pour une machine ?Cela nie le problème de la reconnaissance automatique des formes, qui a donné lieu à de nombreuses recherches ces dernières années. Plusieurs techniques ont été développées pour la reconnaissance des fomes,Le but de ce mémoire est de mettre en place une application de reconnaissance des formes(un type des formes spécifique qui est le fruit de datte)capable de reconnaître ce fruit et classifie chacun a traver leur classe,en utilisant la technique des reseaux de neurone convolutionnel(CNN). Les résultats que nous avons obtenus sont très encourageantes.

Mots clés : reconnaissance des formes,Machine learning, reseaux de neurone convolutionnel.

Table des matières

Table des figures

Introduction

La reconnaissance des formes a pour objet de simuler les activités de perception sensorielle du cerveau. En premier lieu, la perception visuelle et auditive, y compris dans des bandes spectrales non perçues par l'homme (infra rouge, radar, sonar, ...). La reconnaissance a besoin d'un modèle de l'objet. Pour l'être humain, ce modèle correspond à une représentation mentale de l'objet qui peut être apprise en retenant les caractéristiques les plus discriminantes de l'objet. Les caractéristiques peuvent être toutes sortes d'attributs de l'objet : forme, couleur, texture, taille, volume, etc [1].

Objet de recherches depuis l'apparition de l'informatique, cette discipline prend désormais un essor considérable dans l'industrie, car les progrès de l'électronique permettent désormais de doter les systèmes opérationnels de fortes puissances de calcul à des prix raisonnables. Elle regroupe un certain nombre de techniques allant de la simple identification d'objets isolés à l'analyse de scènes complexes ou la compréhension de la parole. Dans ces derniers cas, il faut tout à la fois segmenter en objets, identifier ces objets et établir un réseau de relations entre objets. La notion d'apprentissage est au coeur de la plupart des techniques développées. Ces techniques peuvent être utilisées soit seules dans le cas de systèmes dont la mission principale est d'identifier des objets, soit en relation étroite avec des techniques d'intelligence artificielle dans le cas de systèmes devant à la fois percevoir et raisonner sur les choses perçues.

En d'autres termes, le problème que cherche à résoudre la reconnaissance des formes est d'associer une étiquette à une donnée qui peut se présenter sous la forme d'une image brute ou d'un signal. Des données différentes peuvent recevoir la même étiquette ; ces données sont les réalisations ou les exemplaires de la classe identifiée par l'étiquette. Par exemple, l'écriture manuscrite du caractère A varie d'un scripteur à l'autre mais le lecteur identifiera le caractère A pour chacune de ces réalisations[1].

Des méthodes générales ont été développées en reconnaissance des formes pour extraire automatiquement des informations des données sensibles afin de caractériser les classes de formes

(apprentissage) et d'assigner automatiquement des données à ces classes (reconnaissance).
une des methodes les plus populaires dans ce domaine est reseaux de neurones artificiels qui a prouvé sa capacité de reconnaissance des formes ,malgré ca RNA classique ait mal à développer la précision de la reconnaissance grace à cela ,les chercheures ont développé ce system pour obtenir meilleur resultat ;et à travers cela nous dirigeons vers Les réseaux de neurones convolutionnels(CNN) .
Dans ce travaill nous sommes posés comme objectif de modéliser et concevoir une application de reconaissance de forme qu'on utilisera pour détecter specialement de fruit,on a travaillé avec (fruit dates) pour en savoir plus sur ces fruits et leurs variétes dans le classification et reconnaissance,notre travail basé sure la methode de l'apprentissage (CNN) .
notre manuscrit est partagé en 4 chapitres :

— le premier chapitre traite les notions de la recenaissance des formes

— le deuxi‘eme chapitre présente les notions de machine learning

— le troisiéme chapitre détail le principe de RN(et RNC)

— le dernier chapitre présente les étapes de modélisation , conception et l'analyse des résultas

Chapitre I

Reconnaissance des formes

I.1 Introduction

L'un des principaux objectifs de l'ordinateur est d'automatiser les tâches habituellement effectuées par l'homme. La plupart des activités humaines reposent sur une faculté importante de notre cerveau : pour voir distinguer entre les formes. Par exemple, reconnaître une pomme d'une orange ; un bon diagnostic d'un mauvais, ou encore un sous-marin d'une baleine. Pour pouvoir automatiser ce genre de tâches, l'ordinateur doit obligatoirement acquérir la faculté de reconnaître les formes.
Dans ce chapitre, nous allons passer en revue quelques techniques les plus connues permettant la reconaissance des formes.

I.2 La reconnaissance des formes

I.2.1 Définition

On désigne par reconnaissance de formes (ou parfois reconnaissance de motifs) un ensemble de techniques et méthodes visant à identifier des motifs à partir de données brutes afin de prendre une décision dépendant de la catégorie attribuée à ce motif. On considère que c'est une branche de l'intelligence artificielle qui fait largement appel aux techniques d'apprentissage automatique et aux statistiques .
Les formes ou motifs à reconnaître peuvent être de nature très variée. Il peut s'agir de contenu visuel (code barre, visage, empreinte digitale. . .) ou sonore (reconnaissance de parole), d'images médicales (rayon X, EEG, IRM. . .) ou multi spectrales (images satellitaires) et bien d'autres . Watanabe[2] a défini une forme comme : (l'opposé du chaos ; c'est une entité vaguement définie, à laquelle ont peut associer un nom). En des termes informatiques, une forme est un ensemble de valeurs, appelés attributs, auxquels est associé un nom (ou étiquette), qui est leur classe. Plusieurs formes peuvent avoir la même classe, on dit alors que ce sont les exemples ou réalisations de la classe [1].
Le problème que cherche à résoudre la reconnaissance des formes est d'associer une classe à une forme inconnue (qui n'a pas encore de classe associée). On considère souvent la Reconnaissance des formes comme un problème de classification : trouver la fonction qui affecte à toute forme inconnue sa classe la plus pertinente. Elle est partie intégrante de tout système intelligent

destine à la prise de decision [1][3].

I.2.2 Historique

Or que ce soit pour déchiffrer un texte dactylographié ou manuscrit, pour compter des chromosomes, reconnaître une tumeur, un char ou un avion de guerre, la compréhension de l'image, sa classification passe toujours par la reconnaissance d'une forme.(Plusieurs approches théoriques ont été développées), explique *Olivier Faugeras* [2].
Les premières consistaient à faire des calculs à partir de l'image et construire des représentations symboliques de plus en plus complexes, d'abord en deux dimensions tel que sur l'image, puis tridimensionnelles, pour tenter de restituer une description proche de notre propre vision. Un peu partout dans le monde, les chercheurs ont mis au point des méthodes mathématiques permettant de détecter les contours des objets à partir des changements rapides de contraste dans l'image, des ombres et des lumières, des régions homogènes en couleur, en intensité, en texture.

Dès 1964, des chercheurs français, Georges Matheron (1930-2000) et Jean Serra, ont développé une autre approche théorique (baptisée morphologie mathématique) et un outil spécifique (l'analyseur de texture breveté en 1965, ndlr) d'abord pour analyser des microphotographies de terrain et évaluer des teneurs en minerai, puis pour d'autres applications comme la cytologie (caractérisation et comptage de cellules) rappelle Olivier Faugeras. En 1968, ils créent le Centre de morphologie mathématique de l'Ecole des Mines de Fontainebleau. Leurs outils d'analyse et d'interprétation d'images sont longtemps restés franco-français, jusqu'à ce qu'un américain, Robert Haralick (Université du Kansas à cette époque, de Seattle actuellement), en fasse une large publicité dans les années 1980, en les adaptant à de nombreuses applications : industrielles comme l'inspection radiographique des ailes d'avions de Boeing, aériennes ou médicales [6][2].

D'autres chercheurs, comme les américains Marvin Minsky et Seymour Papert du MIT (Massachussets Institute of Technology) ont considéré le problème dans l'autre sens, en cherchant à formaliser et à faire reproduire par l'ordinateur notre propre processus de reconnaissance d'images, donc notre propre vision. Cette démarche était dans l'air du temps, au coeur des promesses de ' l'intelligence artificielle' qui devait permettre de mettre l'intelligence en équations et doter les ordinateurs de toutes les capacités humaines de raisonnement, mémoire, perception. Or la vision s'est révélée être un domaine particulièrement complexe à modéliser tant elle est

basée sur une quantité phénoménale de connaissances à priori fondée sur notre intelligence et notre expérience [1][4].

I.3 Méthodes

La reconnaissance de motifs peut être effectuée au moyen de divers algorithmes d'apprentissage automatique tels :

— un réseau de neurones

— une analyse statistique

— l'utilisation de modèles de Markov cachés

— une recherche d'isomorphisme de graphes ou sous-graphes

Les formes recherchées peuvent être des formes géométriques, descriptibles par une formule mathématique, telles que :

— cercle ou ellipse

— courbes de Bézier, splines

— droite

Elles peuvent aussi être de nature plus complexe :

— lettre

— chiffre

— empreinte digitale

Les algorithmes de reconnaissance peuvent travailler sur des images en noir et blanc, avec en blanc les contours des objets se trouvant dans l'image. Ces images sont le fruit d'algorithmes de détection de contours. Ils peuvent aussi travailler sur des zones de l'image prédéfinies issues de la segmentation de l'image[1].

I.3.1 Méthode de reconnaissance des formes

— Mise en correspondance de graphes

— Méthode Bayesienne

— Estimation Paramétrique

— Classifieur linéaire

— Réseau de neurones

— ocal feature focus

— SVM : Support Vector Machine

— Polytôpes de contrainte

— Méthode des hypercubes

Un algorithme bien connu pour la détection de formes, la transformée de Hough, est une méthode d'estimation paramétrique[1][2].

I.4 La reconnaissance de plusieurs objects dans une image

Une seule image peut être constituée d'un ou plusieurs objets [7]. Dans le cas où on désire détecter plusieurs objets dans une même image, on peut utiliser le procédé de reconnaissance multi objets représenté sur I.1 ci- dessous.

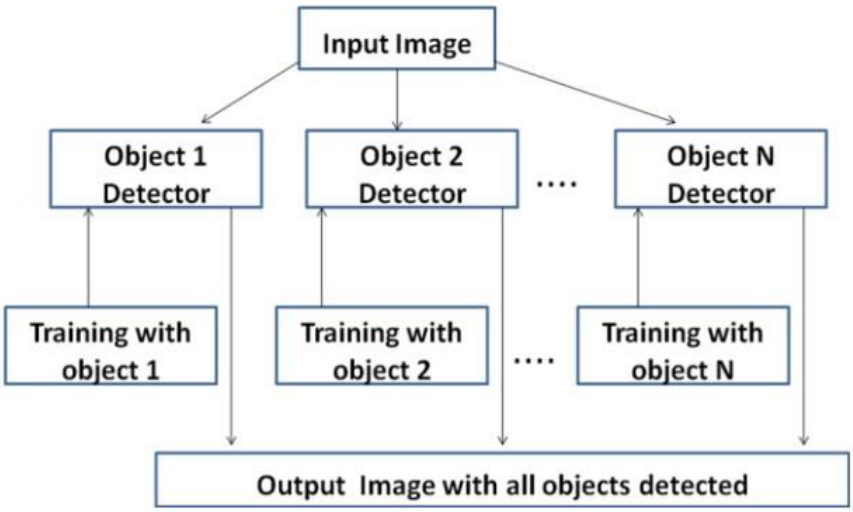

Figure I.1 – La méthode de reconnaissance de multi objets dans une image

I.5 Applications typiques de la reconnaissance des formes

— **Marketing :** La reconnaissance des formes est souvent utilisée pour classer les consommateurs selon les produits qu'ils sont susceptible d'acheter. Elle est aussi utilisée par les

sociétés de vente pour classer les clients selon qu'ils soient de bons ou mauvais payeurs, ou encore selon qu'ils vont oui ou non passer à la concurrence [8].

— **Finances :** les systèmes de reconnaissance des formes sont utilisés pour la détection de transactions bancaires frauduleuses ainsi que la prédiction des banqueroutes [9].

— **Usinage :** la qualité des produits dépend souvent de paramètrisation correcte, et les relations exactes entre la qualité et les valeurs des paramètres n'est pas claire. Les systèmes de reconnaissance des formes sont utilisés pour classer les paramètres selon la qualité des produits qu'ils sont susceptibles de générer. Ils permettent ainsi de réduire le nombre d'essais ce qui fait gagner du temps et de l'argent [10].

— **Energie :** les systèmes de reconnaissance des formes sont utilisés pour prévoir la consommation électrique (réduite, normale, élevée), permettant ainsi aux clients de réduire si nécessaire leur consommation, et aux producteurs de mieux gérer leurs unités de production [11].

— **Lecture automatisée :** les systèmes de reconnaissance des formes permettent de numériser les anciens documents ainsi que les archives, non pas sous la forms d'images, mais plutôt sous une forme textuelle [12].

— **SECURITE :** la reconnaissance vocale et rétinienne sont un exemple d'applications typiques de la reconnaissance des forms pour l'authentification. La vérification des signatures est aussi très populaire [13].

I.6 Schéma général d'un systéme de reconnaissance des formes

La majorité des systèmes de Reconnaissance des formes ont le schéma de fonctionnement suivant [14][1] :

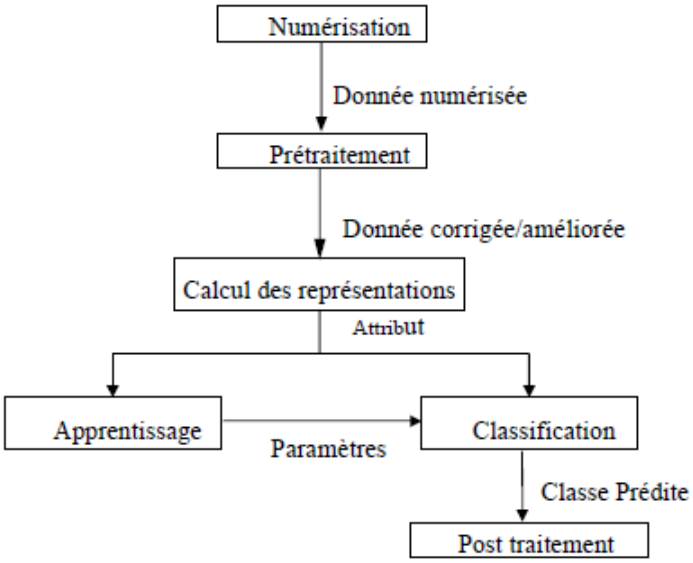

Figure I.2 – Schéma général d'un système de reconnaissance des formes

I.6.1 Préparation des données

— **Numérisation :** À partir des informations du monde physique, construire une représentation des données directement manipulable par la machine. Des capteurs (microphone, caméra, instruments de mesure) convertissent les signaux reçus du monde reel en une representation numérique discrète. L'espace résultant, appelé *espace de représentation* a une dimension **r** très grande lui permettant de disposer du maximum d'informations sur les forms numérisées.

— **Prétraitement :** Consiste à sélectionner dans l'espace de représentation l'information nécessaire au domaine d'application. Cette sélection passe souvent par l'élimination du bruit, la normalisation des données, ainsi que par la suppression de la redondance. Le nouvel espace de representation a une dimension **r'** très inférieure à **r** mais demeure un espace de grande dimension et contient des informations encore assez primitives [1].

— **Calcul des représentations :** Il s'agit de la phase finale de la préparation des données. Elle fournit un certain nombre de caractéristiques ou paramètres (les fameux attributs) en utilisant des algorithmes de sélection et/ou d'extraction d'attributs. Les attributs étant limités en nombre, *l'espace des paramètres* ainsi obtenu est de dimension p très petite par rapport à **r'** [1].

I.6.2 Apprentissage

L'apprentissage ou entraînement, est une partie importante du système de reconnaissance. Le classificateur étant généralement une fonction paramétrique, l'apprentissage va permettre d'optimiser les paramètres du classificateur pour le problème à résoudre, en utilisant des données d'entraînement. Lorsque les données d'entraînement sont préalablement classées, l'apprentissage est dit supervisé, sinon il est non supervisé [15].

I.6.3 Classification

cette phase est le noyau de la Reconnaissance des formes. En utilisant les modèles (paramètres) obtenus lors de l'apprentissage, le classificateur assigne à chaque forme inconnue sa ou ses formes les plus probables [1].

I.6.4 Post traitement

cette phase a pour but de corriger les résultats de la classification en utilisant des outils spécifiques au domaine d'application. Par exemple pour un système de reconnaissance de textes manuscrits, le classificateur se charge de classer chaque caractère séparément, alors que le post traitement appliqué un correcteur orthographique sur tout le texte pour valider et éventuellement corriger le résultat de la classification. Bien que facultative, cette phase permet d'améliorer considérablement la qualité de la reconnaissance [1].

I.7 Traitement d'image

L'image fournie par le capteur est transformée en un signal électrique. De ce signal il faut extraire les informations recherchées sur le contenu de la scène dont l'image a été captée. Les premières bases du traitement d'images sont directement issues du traitement du signal, phénomène normal puisque toute image, qu'elle soit continue ou numérique, peut être considérée comme un signal à 2 dimensions [1].
La vision n'est pas un domaine facile, car repérer un objet simple dans une image demande beaucoup d'opérations. L'objectif de ce travail est de montrer comment il est possible de développer

des algorithmes de traitement d'images pour réaliser par exemple une segmentation et détection d'objets dans une image. Le traitement, souvent appelé prétraitement, regroupe 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.

La représentation la plus élémentaire correspond à l'image binaire pour laquelle chaque pixel ne peut prendre qu'une valeur parmi deux autres. Pour les images monochromes, chaque pixel peut prendre une valeur parmi N. N correspond généralement à une puissance de 2, ce qui facilite la représentation de l'image en machine. Par exemple, pour une image en niveau de gris chacun des pixels peut prendre une valeur parmi 256. Sa valeur est alors codée par un octet de donnée. Une image est constituée par une matrice X lignes et Y colonnes de pixels chacun codé par x bits [1]. Les données quantitatives liées à la représentation des images sont représentées dans le tableau suivant I.3 :

1 bit	2 couleurs (noir & blanc)
4 bits	16 couleurs
8 bits	256 couleurs ou niveaux de gris
16 bits	65536 couleurs
24 bits	16777216 couleurs (vraies couleurs)
32 bits	4 294 967 296 couleurs

Figure I.3 – Données quantitatives des images

I.7.1 Préparation ou Traitement bas niveau

Avant de présenter les différentes étapes de prétraitement, il est nécessaire de définir ce qu'est une image en l'occurrence une image bruitée et le type d'image que nous allons travailler.

I.7.1.1 Définition d'image

Une image est constituée d'un ensemble de points pixels (Picture Element). Il représente le plus petit élément constituant d'une image numérique (on parle d'image numérique lorsque les quantités physiques qui caractérisent l'image sont converties par des valeurs numériques). L'ensemble des ces pixels est contenu dans un tableau à deux dimensions constituant l'image.

Les axes de l'image sont orientés de la façon suivante [1] :

• L'axe X est orienté à droite (largeur)

•L'axe Y est orienté de haut en bas (hauteur), contrairement aux notations conventionnelles en mathématiques, ou l'axe Y est orienté vers le haut.

Pour représenter informatiquement une image, il suffit donc de créer un tableau de pixels dont chaque case est codée par un certain nombre de débits déterminant la couleur ou l'intensité du pixel, la I.4 présente un réseau de pixels (L'élément grisé encadré par la couleur rouge correspond aux coordonnées i, j) [1][3].

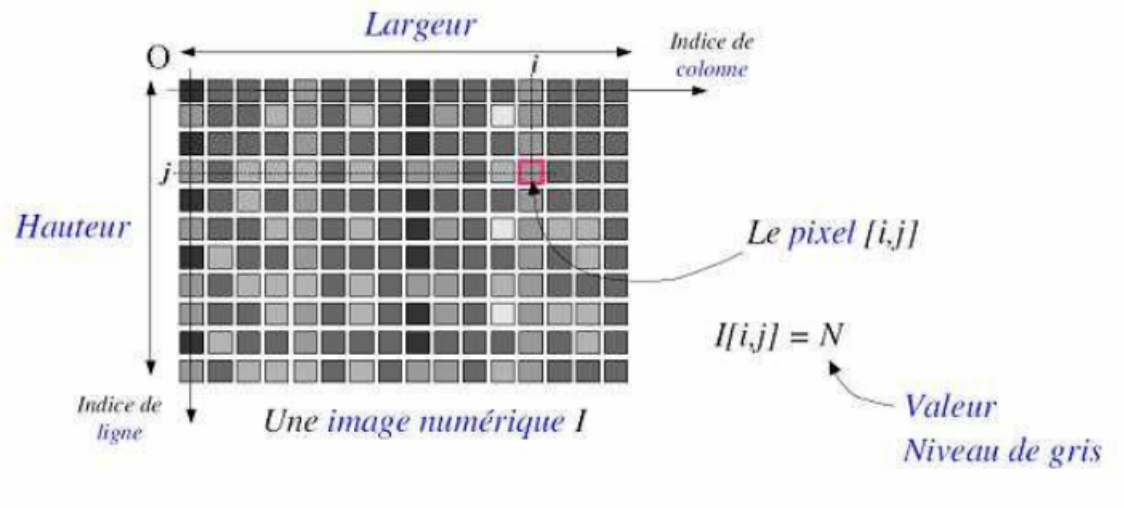

Figure I.4 – Exemple de réseau de pixels

I.7.1.2 Définition pixel

Le terme pixel est la contraction de l'anglais " Picture element ". C'est le plus petit élément de l'image. Sortie de l'appareil photo une image est composée d'un nombre x de pixels. Un pixel a une couleur exprimée (codée) en langage binaire mathématique (uniquement des 0 et des 1). On comprend donc tout de suite qu'il faut cette quantité minimum de nuances pour avoir une qualité correcte, appelée parfois qualité image. On dit que l'image est codée en 8 bits par couche ($28 = 256$ niveaux) soit 8 pour le R, 8 pour le V, 8 pour le B au total 24 bits en RVB [16][1]

I.7.1.3 Image en niveu de gris

On passe en niveaux de gris en considérant comme seule composante la luminosité. Il existe plusieurs manières de convertir une image RGB en niveaux de gris. La plus simple est de faire :

$$gris = \frac{rouge + vert + bleu}{3}$$

Cela équivaut aussi à affecter la couleur en niveau de gris à chacune des trois composantes RGB. L'idéal est de faire ressortir la luminosité d'un pixel. Celle-ci vient principalement de la présence de la couleur verte. On emploie généralement les coefficients suivants :

Gris= 0, 299 · rouge+ 0, 587 · vert + 0, 114 · bleu

Dans ce cas on dispose d'une échelle de teintes de gris, et la plupart du temps on dispose de 256 niveaux de gris avec [17] :

0 — noir, ...127 — gris moyen, ..., 255 — blanc

Certaines images peuvent être codées sur deux octets ou plus (certaines images médicales, des images astronomiques,...) ce qui peut poser des problèmes dans la mesure où les systèmes de traitement d'images courants supposent l'utilisation des pixels d'un octet. Notons au passage que l'humain standard ne reconnaît au plus que 64 niveaux de gris

I.8 Segmentation des images

La segmentation est une étape essentielle en traitement d'images et reste un problème complexe. La segmentation est un processus de la vision par ordinateur, généralement c'est la première étape de l'analyse d'image qui vient après le prétraitement. La segmentation est l'extraction de caractéristiques de l'objet, ce qui permet une distinction entre l'objet et le fond. Elle aide à localiser et à délimiter les entités présentes dans l'image. Il existe une multitude de méthodes de segmentation dont l'efficacité reste difficile à évaluer.
La segmentation des images est le procédé qui conduit à un découpage de l'image en un nombre fini de régions (ou segments) bien définies qui correspondent à des objets, des parties d'objets

ou des groupes d'objets qui apparaissent dans une image. C'est une transformation très utile en vision artificielle.
Une erreur dans la segmentation de la forme à reconnaître augmente forcément le risque d'une mauvaise reconnaissance. Essentiellement, l'analyse de l'image fait appel à la segmentation où l'on va tenter d'associer à chaque pixel de l'image un label en s'appuyant sur l'information portée (niveaux de gris ou couleur), sa distribution spatiale sur le support image, des modèles simples (le plus souvent des modèles géométriques) [18][1].

I.8.1 Les principes de la segmentation

De nombreux travaux ont été réalisés sur ce sujet, dans des domaines aussi variés que le domaine médical, militaire, industriel, géophysique, etc... C'est toujours un sujet d'actualité et un problème qui reste ouvert où l'on retrouve de très nombreuses approches :

•La segmentation par régions
•La segmentation par seuillage
•La segmentation par contours
•La Transformée de Hough
•La segmentation par étiquetage en composantes connexes
•La segmentation par LPE

Toutes ces approches visent à l'extraction des caractéristiques. Après de nombreuses années passées à rechercher la méthode optimale, les chercheurs ont compris que la segmentation idéale n'existait pas. Il n'existe pas d'algorithme universel de segmentation à chaque type d'images correspond une approche spécifique. Une bonne méthode de segmentation sera donc celle qui permettra d'arriver à une bonne interprétation [18].

I.9 Conclusion

On peut considérer les systèmes de reconnaissance des formes comme des systèmes à l'affectation d'une étiquette à une valeur d'entrée donnée, les méthodes de reconnaissance de formes visent généralement à fournir une réponse raisonnable pour toutes les entrées possibles et à effectuer une correspondance «la plus probable » des entrées, en tenant compte de leur variation statistique.

La Reconnaissance des Formes met en œuvre plusieurs étapes : segmentation des objets (analyse d'images), extraction de caractéristiques (géométrie, invariants, ...), classification (supervisée ou non, méthodes probabilistes, statistiques, ...). Les applications sont très variées et nécessitent des connaissances expertes du domaine d'application. Les méthodes sont nombreuses mais les principes de base sont assez stables.

Dans le prochain chapitre, nous allons décrire les differents types d'apprentissage.

Chapitre II

Machine learning

II.1 Introduction

Les inventeurs rêvent depuis longtemps de créer des machines réfléchissantes, et ce désir remonte au moins à l'époque des Grecs. Lorsque les ordinateurs programmables ont été conçus, les utilisateurs se sont demandé si ces appareils pourraient devenir intelligents. L'intelligence artificielle est aujourd'hui un domaine florissant qui compte de nombreuses applications et sujets de recherche actifs. Attendez-vous à des programmes intelligents pour automatiser le travail de routine, comprendre la parole ou l'image et établir des diagnostics Médecine et soutien à la recherche scientifique fondamentale. Le véritable défi de l'intelligence émotionnelle réside dans le fait qu'elle résout des tâches facilement réalisables, mais il est difficile de décrire des problèmes formels que nous résolvons de manière intuitive, tels que la reconnaissance de mots prononcés ou de des formes specials dans des images commes les fruits ou les vehicule. Le terme "apprentissage automatique" fait référence à la détection automatique de modèles de signaux non significatifs dans des données. Au cours des deux dernières décennies, il est devenu un outil commun dans presque toutes les tâches nécessitant l'extraction d'informations à partir de grands ensembles de données . L'apprentissage en profondeur est un sous-ensemble d'apprentissage automatisé qui est loin d'extraire des modèles de données utiles de manière automatisée en améliorant le réseau artificiel.

II.2 Apprentissage machine

L'apprentissage machine est une sous-discipline de l'intelligence artificiel dont l'objectif ultime est de reproduire chez l'ordinateur les capacités cognitives de l'être humain, par le biais de l'apprentissage. Ici, le terme apprentissage est à mettre en opposition avec des techniques d'intelligence artificielle basées sur des comportements intelligents "pré-codés", comme dans le fameux programme ELIZA [19], où l'ordinateur donnait l'illusion de pouvoir poursuivre une conversation intelligente avec un humain à partir d'un système en fait très basique de détection de mots-clés et de réponses prédéfinie. L'approche "apprentissage machine" consiste plutôt à programmer des mécanismes qui permettent de développer les connaissances c'est-à-dire d'apprendre à partir d'observations (les exemples d'apprentissage) de manière automatique. Les êtres humains faisant preuve de capacités d'apprentissage impressionnantes, il est naturel que l'apprentissage machine s'inspire d'eux pour tenter de reproduire leurs comportements. En particulier, les travaux en neurosciences visant à comprendre les mécanismes d'apprentissage dans

le cerveau. Une classe d'algorithmes d'apprentissage très populaire, les réseaux de neurones, a ainsi été inspirée de telles observations biologiques. Mais les détails du fonctionnement du cerveau restant pour l'instant en grande partie un mystère. Les algorithmes développés en apprentissage machine ont des objectifs moins ambitieux. Ils ont chacun leurs propres forces et faiblesses – souvent très différentes de celles des humains – et sont généralement prévus pour résoudre certains types de problèmes spécifiques [20].

II.3 Différents types d'apprentissage

II.3.1 Apprentissage supervisé

La forme d'apprentissage qui est considérée comme la plus intuitive est celle dite d'apprentissage supervisé. Cela signifie que la réponse attendue est observée dans les données collectées. Formellement, si l'on note z un exemple d'apprentissage, alors on peut écrire z=(x,y) avec x la partie "entrée", c'est-à-dire les données que l'algorithme est autorisé à utiliser pour faire une prédiction, et y la partie "étiquette", c'est-à-dire la valeur correcte pour la prédiction. Par exemple, si la tâche consiste à déterminer le sexe d'une personne à partir d'une photo d'identité, x serait la photo et y soit "homme", soit "femme" (en supposant qu'il s'agisse des deux seules possibilités). Dans un tel cas où les valeurs possibles de l'étiquette y ne sont pas interprétables comme des nombres, on parle de tâche de classification. Si par contre la tâche était de prédire l'âge de la personne plutôt que son sexe, y serait un nombre et on parlerait d'une tâche de régression. La III.1 illustre ces deux situations.

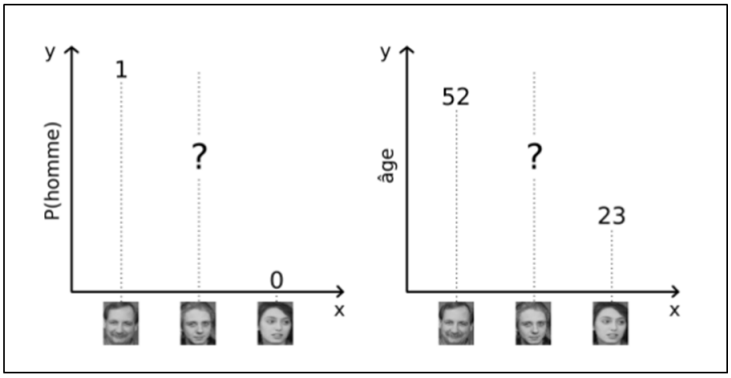

Figure II.1 – Exemples d'apprentissage supervisé : classification (à gauche) pour prédirele sexe, et régression (à droite) pour prédire l'âge, Les photos aux deux extrémités de l'axe des x sont des exemples d'entraînement pour lesquels l'étiquette est connue, tandis que la photo du milieu est celle pour laquelle on veut obtenir une prédiction

L'algorithme est entraîné sur un ensemble de données pré-collectées (l'ensemble d'entrainement), de la forme $D = z_1, \ldots, z_n$, avec $z_i = (x_i, y_i)$. De nombreux algorithmes supposent que ces exemples sont tirés de manière indépendante et identiquement distribuée (i,i,d) d'une distribution P, c'est-à-dire. $z_i ... P(X, Y)$ (où la forme exacte de P n'est pas connue d'avance). Selon la tâche à résoudre, la prédiction d'un algorithme d'apprentissage supervisé va généralement tenter d'approximer l'une des trois quantités suivantes :

— $P(x|y)$: dans notre exemple de classification (trouver le sexe), il faudrait prédire la probabilité qu'une photo soit une photo d'un homme par un nombre p (la probabilité que ce soit la photo d'une femme étant alors 1 p). Dans notre exemple de régression (trouver l'âge), la prédiction serait une densité de probabilité sur l'intervalle]0,+[.

— $argmax_y P(x|y)$: dans notre exemple de classification, il s'agirait d'une prédiction binaire du sexe le plus probable ("homme" ou "femme"). Dans notre exemple de régression, il s'agirait de l'âge le plus probable.

— $E_Y = [Y|x] = \int_y yP(y|x)y$: cette quantité n'a de sens que pour la régression, et il s'agirait dans notre exemple de prédire l'âge moyen de la personne étant donnée sa photo.
Il faut noter que la prédiction de $P(x|y)$ est la tâche la plus générale (dans la mesure où la résoudre permet également d'accomplir les deux autres), mais tous les algorithmes d'apprentissage ne sont pas capables d'estimer une distribution de probabilité [20].

II.3.2 Apprentissage non supervisé

Dans l'apprentissage non supervisé, un exemple $z_i = x_i \; .. P(x)$ ne contient pas d'étiquette explicite. Il existe plusieurs types de tâches d'apprentissage non supervisé, parmi les quelles les plus souvent rencontrées sont :

— L'estimation de densité : estimer $P(x)$, comme illustré en II.2.

— La génération de données : tirer de nouveaux exemples d'une distribution la plus proche possible de $P(X)$.

— L'extraction de caractéristiques : trouver une fonction f telle que $f(x)$ soit "intéressant", par exemple pour :
1. compresser x, c'est-à-dire. Que $f(x)$ de vrai être de dimension plus petite que x tout en permettant de reconstruire x par une fonction g telle que $x = g(f(x))$.
2. simplifier l'apprentissage à partir de x, c'est-à-dire. Que $f(x)$ de vrai être tel qu'un

algorithme d'apprentissage (possiblement supervisé) utilisant $f(x)$ comme entrée au lieu de x donne de meilleurs résultats

— Le regroupement ("clustering" en anglais) : partitionner les exemples en groupes $G_1, \ldots, G_k$ tels que tous les exemples dans un même groupe soient similaires (où plusieurs notions de similarité peuvent être utilisées, menant à des résultats potentiellement très déférents) [20].

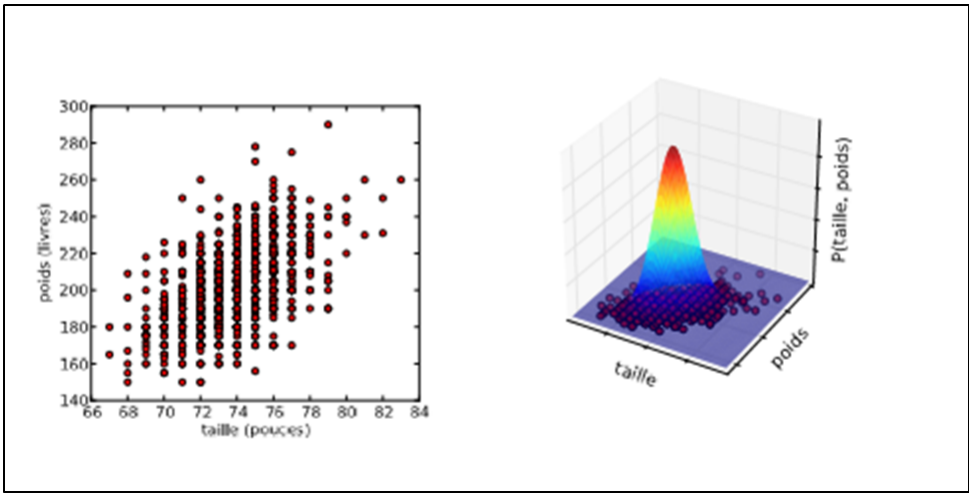

Figure II.2 – Exemple d'apprentissage non supervisé : l'estimation de densité. Àgauche, les données d'origine (taille et poids de 1033 joueurs de baseball aux Etats-Unis). À droite, l'estimation de la densité par une distribution Gaussienne.

II.3.3 Apprentissage semi-supervisé

Comme le nom l'indique, l'apprentissage semi-supervisé est à mi-chemin entre le supervisé et le non supervisé : certains exemples $z_i = (x_i, y_i))$ $(1 < \mathrm{i} < l)$ ont une étiquette, tandis que d'autres exemples $z_i = x_i$ $(l{+}1 < \mathrm{i} < n)$ n'en ont pas (on dit qu'ils ne sont pas "étiquetés"). Les algorithmes d'apprentissage semi-supervisé tentent généralement de résoudre des problèmes d'apprentissage supervisé, mais en utilisant les exemples non étiquetés pour améliorer leur prédiction. La distribution des entrées P(X) peut nous donner de l'information sur $P(x|y)$ même en l'absence d'étiquettes. Un exemple typique où c'est le cas, pour une tâche de classification, est lorsque les exemples de chaque classe forment des groupes distincts dans l'espace des entrées, séparés par des zones de faible densité P(x). La II.3 montre ainsi deux classes dont la forme en croissant est révélée par les exemples non étiquetés. Un algorithme purement supervisé donc ignorant ces exemples ne pourrait pas identifier correctement ces deux classes.

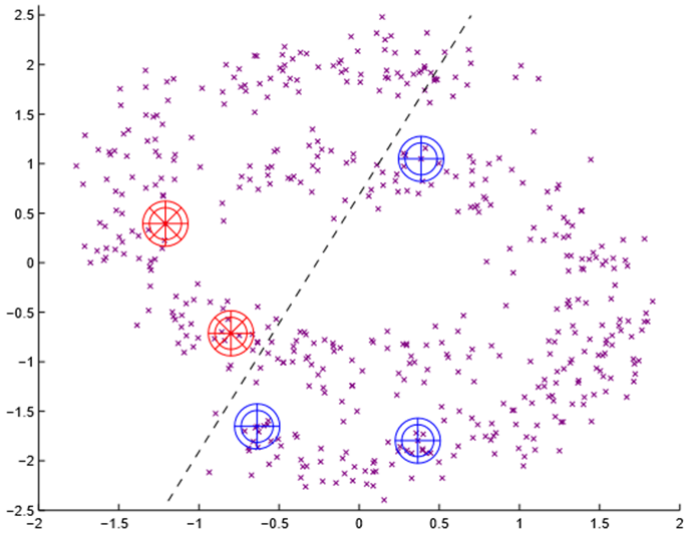

Figure II.3 – Apprentissage semi-supervisé : ici seuls 5 exemples sont étiquetés (deux de la classe * en rouge, et trois de la classe + en bleu). Un algorithme n'utilisant pas les exemples non étiquetés (petits x mauves) séparerait les deux classes par exemple selon la ligne en pointillés, alors qu'un algorithme semi-supervisé (ou un humain) pourrait séparer les deux "croissants de lune" correspondant à chaque classe.

Dans le cadre de l'apprentissage supervisé décrit précédemment, l'application d'un algorithme se fait typiquement en deux étapes :

— L'algorithme va d'abord apprendre une tâche (phase d'entraînement) sur un ensemble $D = (x_1, y_1), , (x_n, y_n)$.

— L'algorithme doit ensuite faire ses prédictions (phase de test) sur un ensemble T dans on ne fournit pas les étiquettes.

Un algorithme semi-supervisé peut également suivre ces deux étapes (en n'oubliant pas que D peut contenir également des exemples non étiquetés). On dit alors que la phase de prédiction sur l'ensemble de test se fait par induction. Mais puisque l'algorithme peut utiliser des exemples non étiquetés au cours de l'apprentissage, on peut également inclure T dans D : on parle alors de transduction, et en général les performances seront meilleures qu'en induction puisque plus d'exemples sont disponibles pour l'apprentissage. Il existe malgré tous des applications où il n'est pas possible d'inclure T dans l'ensemble d'entraînement, par exemple lorsque les éléments de T sont générés en temps réel et que l'on a besoin de prédictions immédiates : si ré-entraîner l'algorithme avant chaque nouvelle prédiction s'avère trop lent, l'induction est alors la seule option possible [20].

II.3.4 Apprentissage par renforcement

L'apprentissage par renforcement est une forme d'apprentissage supervisé où l'algorithme n'observe pas une étiquette pour chacune de ses prédictions, mais plutôt une mesure de la qualité de ses prédictions (possiblement prenant en compte toute une séquence de prédictions)[20].

II.4 Généralisation

II.4.1 Sur-apprentissage

L'entraînement d'un algorithme d'apprentissage consiste à extraire, de manière explicite ou implicite, des caractéristiques de la distribution de probabilité P qui génère les données. Mais P étant inconnue, l'algorithme se base à la place sur un nombre fini d'exemples d'entraînement, c'est-à-dire. Sur la distribution discrète P' des exemples disponibles dans D (appelée la distribution empirique). Lorsque certaines caractéristiques apprises sur P ne s'appliquent pas à P, on parle de sur-apprentissage, et on risque une mauvaise généralisation, c'est-à-dire. Que l'algorithme neva pas obtenir une bonne performance sur de nouveaux exemples tirés de P. Prenons l'exemple de la classification, lorsqu'un modèle estime $p(y|x)$ par une fonction $q_x(y)$. Afin de mesurer la similarité entre q_x et $p(.|x)$, on peut par exemple utiliser la divergence de Kullback-Leibler D_KL [20], définie par :

$$D_{KL}(P(\cdot|x)\|q_x) = \sum_y P(y|x) \ln \frac{P(y|x)}{q_x(y)}.$$

Cette quantité est toujours supérieure ou égale à zéro, et est égale à zéro si et seulement si q_x est égale à $p(.|x)$. La minimisation de la divergence de Kullback-Leibler est donc un critère raisonnable pour obtenir une fonction q_x qui approxime $p(.|x)$. Vu que le but est d'obtenir une bonne approximation pour toutes les valeurs de x qui pourraient être générées par P, il est logique de considérer la minimisation du critère. (éq 1.1)

$$\begin{aligned} C(q) &= E_X[D_{KL}(P(\cdot|X)\|q_X] \\ &= \int_x \sum_y P(x)P(y|x) \ln \frac{P(y|x)}{q_x(y)} dx. \end{aligned}$$

$C(q)$ Est ici ce que l'on appelle l'erreur de généralisation, c'est-à-dire. L'erreur moyenne sur des exemples tirés de P. Puisque P est inconnue, on minimise en pratique un critère C défini de la même manière en remplaçant P par P . C'est le principe de minimisation du risque empirique [20], et C s'écrit ici :(éq 1.2)

$$\hat{C}(q) = \sum_{i=1}^{n} \frac{1}{n} \ln \frac{1}{q_{x_i}(y_i)} = -\frac{1}{n} \sum_{i=1}^{n} \ln q_{x_i}(y_i)$$

qui est appelée la log-vraisemblance négative (en anglais NLL, pour "Negative Log-Likelihood"). Ce critère est minimisé dès que $q_(x_i)(y_i) = 1$, pour tous les exemples d'entraînement $(x_i, y_i)D$ et ce quelle que soit la valeur de $q_x(y)$ pour des valeurs de x non observées dans D.Une fonction q peut donc minimiser $C(q)$ sans nécessairement minimiser C(q) Si c'est le cas, on est en situation de sur-apprentissage, illustrée en II.4.

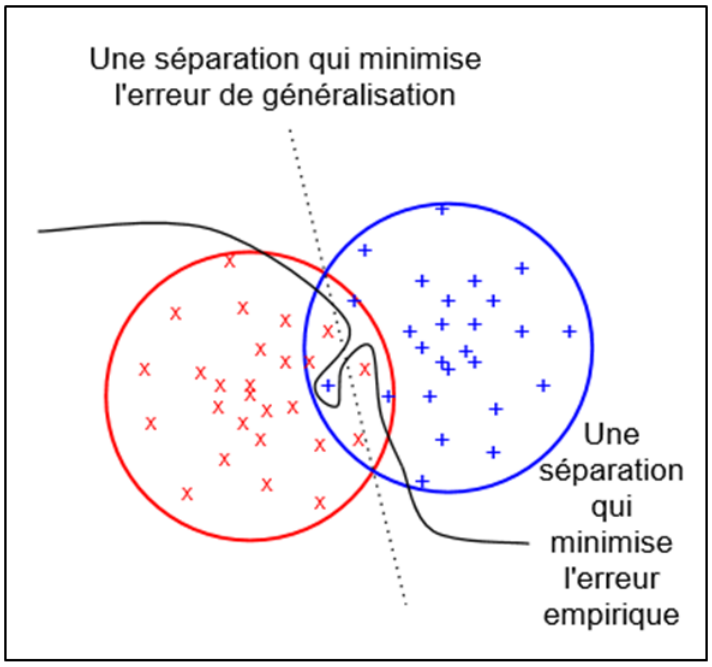

Figure II.4 – Situation de sur-apprentissage : classification binaire (les classes sont les cercles rouge et bleu, avec respectivement les et les + comme exemples d'entraînement). Une séparation idéale en termes d'erreur de généralisation serait la ligne en pointillés, mais un algorithme dont le but est uniquement de minimiser l'erreur empirique pourrait par exemple séparer les exemples selon la ligne pleine : la classification des exemples d'entraînement serait parfaite, mais l'erreur de généralisation serait plus élevée que celle de la ligne en pointillés.

Pour une distribution fixe des données, deux facteurs principaux augmentent le risque de sur-apprentissage :

– Le manque d'exemples d'entraînement : moins il y a d'exemples, plus il existe de fonctions minimisant le critère $C'(q)$(éq 1.2), parmi lesquelles seulement un petit nombre seront vraiment

proches de la “vraie” solution au problème.
– Pas assez de contraintes dans la forme de la fonction q : moins la classe de fonctions à laquelle q appartient est restreinte, plus l'algorithme d'apprentissage risque de tirer parti de la flexibilité de q pour apprendre des “détails” des exemples d'entraînement, qui ne se généralisent pas à la vraie distribution P. C'est le cas par exemple dans la II.4 où la séparation tarabiscotée des exemples par la ligne pleine permet de minimiser l'erreur empirique, mais va mener à une plus grande erreur de généralisation qu'une simple ligne droite [20].

II.4.2 Régularisation

Un moyen de lutter contre le sur-apprentissage est d'utiliser une technique dite de régularisation. Il existe plusieurs méthodes de régularisation, mais elles partagent le même principe : rajouter au processus d'apprentissage des contraintes qui, si elles sont appropriées, vont améliorer les capacités de généralisation de la solution obtenue.
Reprenons par exemple le cas de la classification, où l'on cherche à minimiser le critère $C(q)$(éq 1.1), que l'on approxime par $C'(q)$(éq 1.2)Comme nous venons de le voir, ce problème est mal défini car il existe une infinité de fonctions qui minimisent C', sans donner aucune garantie sur la valeur de C. Une première façon de régulariser le problème est donc de restreindre la forme de q : par exemple x dansR^d et y dans [0,1] on peut se limiter aux fonctions de la forme (éq 1.3) [20]

$$q_x(1) = 1/(1 + e^(- w^T x))$$

Où w dans R^d est le vecteur de paramètres du modèle.
Notons que si les x_i de l'ensemble d'entraînement sont linéairement indépendants, alors cette contrainte sur la forme de q n'est pas suffisante, puisqu'il est toujours possible que $C'(q)$ soit arbitrairement proche de zéro sans pour autant avoir de garantie sur la valeur de $C(q)$. Une technique classique de régularisation consiste alors à rajouter au critère C' une mesure qui pénalise la complexité de la solution, suivant le principe du rasoir d'Occam qui dit que les hypothèses les plus simples sont les plus vraisemblables voir [21]. Une possibilité est de minimiser(éq 1.4).

$$C''(q) = C'(q) + b||w||^2$$

Au lieu de C', pour q défini comme dans (éq 1.3), afin d'empêcher le vecteur w de contenir des valeurs arbitrairement grandes (en valeur absolue). Le paramètre b contrôle la force de cette contrainte (lorsque b –++ la seule solution possible est la fonction constante $q_x(1) = q_x(0) = 0,5$, qui est la plus simple qu'on puisse imaginer). Le critère empirique $C'(q^*)$pour la fonctionq^* qui minimise le critère régularisé C'' pourrait ne pas être proche de zéro, mais on peut souvent ainsi – pour certaines valeurs de b – obtenir des valeurs plus basses du critère de généralisation C (celui qui nous intéresse vraiment). C'est le principe de la minimisation du risque structurel [22][20].

Dans cet exemple, nous avons utilisé $||w||^2$ pour mesurer la complexité de la fonction q définie à partir de w par (éq 1.3). En général, il n'existe pas une seule mesure de complexité universelle qui soit appropriée pour tous les algorithmes d'apprentissage, et le choix de la mesure de complexité à pénaliser joue un rôle très important. La complexité de Kolmogorov [24][23], est une mesure de complexité très générique qui est intéressante en théorie, même si en pratique elle est souvent impossible à utiliser directement. Elle consiste à dire que la complexité d'une fonction est la taille du plus petit programme qui l'implémente. Un premier obstacle à l'utilisation de cette complexité est le fait qu'il faille choisir un langage de programmation approprié : par exemple si le langage choisi contient une fonction primitive qui calcule le produit scalaire, alors, dans notre exemple ci-dessus la plupart des fonctions q définies par (éq 1.3) ont la même complexité de Kolmogorov. Par contre, si le produit scalaire n'est pas une primitive du langage (et qu'il n'y a pas d'instruction de boucle), alors il faut l'écrire comme une somme de produits et q est d'autant plus complexe que w a d'éléments non nuls. Une autre difficulté est qu'il n'est en général pas possible d'optimiser la complexité de Kolmogorov de manière efficace, ce qui rend vaine son utilisation directe dans un processus d'optimisation. Elle a malgré tout de nombreuses applications, comme décrit dans le livre de Li et al [25][20].

II.4.3 Malédiction de la dimensionalité

On peut observer empiriquement – et dans certains cas justifier mathématiquement –que plus la dimension d de l'entrée x est élevée, plus les tâches d'apprentissage machine ont tendance à être difficiles à résoudre. C'est ce qu'on appelle la malédiction de la dimensionalité [26]. Il existe plusieurs manifestations de cette malédiction. La plus importante dans le contexte de cette thèse est le fait que le nombre de combinaisons possibles des entrées augmente exponentiellement avec la dimensionnalité d : en notant x_ij la valeur associée à la j-ème dimension de l'entrée x_i, si

l'on suppose que ces entrées ne peuvent prendre qu'un nombre fini k de valeurs, alors le nombre de combinaisons possibles est égal à k^d. Un algorithme qui apprend "bêtement" à associer une valeur à chaque combinaison sans partager d'information entre les différentes combinaisons n'a aucune chance de fonctionner en haute dimension, car il ne pourra pas généraliser aux multiples combinaisons qui n'ont pas été vues dans l'ensemble d'entraînement. Dans le cas où x_ij n'est pas contraint dans un ensemble fini de valeurs, l'intuition reste la même pour certains algorithmes qui consistent à "partitionner" R^d en régions indépendantes (possiblement de manière implicite) : si le nombre de ces régions augmente exponentiellement avec d, alors un tel algorithme aura de la difficulté à généraliser pour de grandes valeurs de d. La II.5 illustre ce phénomène en une et deux dimensions, et il faut garder à l'esprit que la situation peut s'avérer encore bien pire lorsque l'on manipule des entrées à plusieurs centaines de dimensions [20].

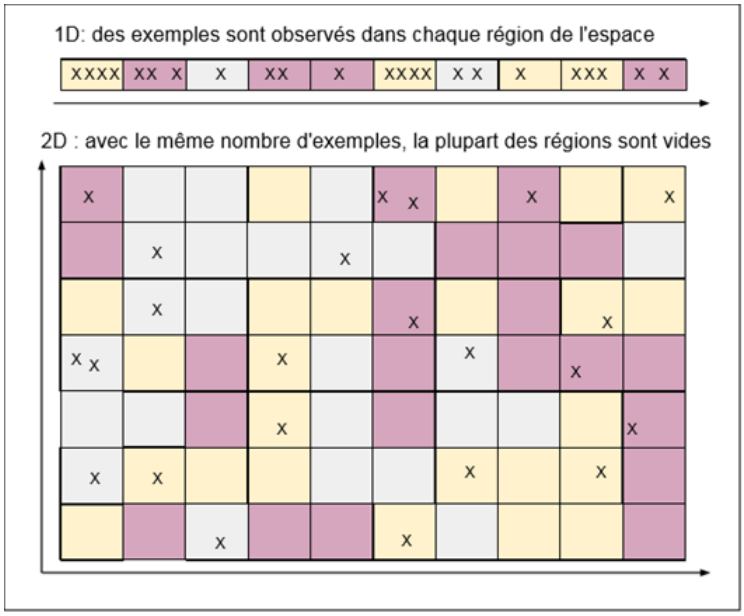

Figure II.5 – Malédiction de la dimensionalité : si l'algorithme partitionne l'espace en régions indépendantes, le nombre d'exemples nécessaires pour remplir ces régions augmente de manière exponentielle avec la dimension. Ici, la couleur d'une région représente la classe majoritaire dans cette région, et un tel algorithme pourrait bien généraliserà partir de 23 exemples d'entraînement pour le problème du haut (1D), mais pas pour celui du bas (2D).

II.5 Différents types de modèles

II.5.1 Modèles paramétriques

En apprentissage machine, un modèle paramétrique est défini par un ensemble o de paramètres de dimension fini, et l'algorithme d'apprentissage associé consiste à trouver la meilleure valeur possible de o. Prenons l'exemple typique de la régression linéaire : le modèle tente d'estimer $E_Y[Y|x]$ où y dans$R$ et x dansR^d,par une fonction $f(x) = w^T x + b$, avec w dansR^d et b dans$R$. On a alors $o = [w,b]$ et un algorithme d'apprentissage possible serait de minimiser le critère suivant (qui contient un terme de régularisation) [20] :(éq 1.5)

$$_(i = 1)^n (w^T x_i + b - y_i) + ||w||^2$$

La II.6 montre un exemple de régression linéaire en une dimension, sans régularisation. Les modèles paramétriques peuvent également être statistiquement inefficaces. S'il y a moins d'exemples d'apprentissage, alors le problème est sur-paramétré et on risque le sur-apprentissage : le modèle pourrait apprendre des paramètres taillés "sur mesure" pour les données d'entraînement, mais qui mèneront à une mauvaise généralisation. La conséquence de cette observation est qu'en général, un modèle avec un grand nombre de paramètres est statistiquement inefficace.

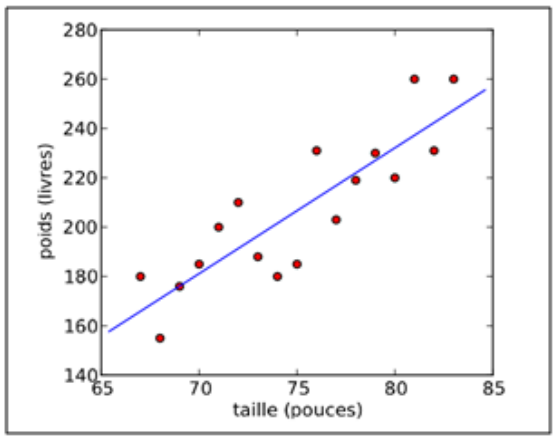

Figure II.6 – Modèle paramètrique : la règression linèaire. Les données (points rouges) sont un sous-ensemble des memes données que dans la figure II.2, et la tâche est ici de prédire le poids d'un joueur de baseball en fonction de sa taille. La prédiction du modéle minimisant le critère de leq. II.5 (ici avec = 0, c.'a.d. sans régularisation) est donnée par la ligne bleue.

II.5.2 Modelés non paramétriques

Un modèle non paramétrique n'a au contraire pas d'ensemble exacte de paramètres : le nombre de variables utilisées par le modèle augmente généralement avec le nombre d'exemples dans l'ensemble d'entrainement. Un exemple de modèle non paramétrique pour résoudre le même problème de régression que celui décrit en II.5.1 est l'algorithme des fenêtres de Parzen, aussi appelé régression de Nadaraya-Watson [19][27]. Il consiste à écrire la prédiction du modèle comme

$$f(x) = 1/(_(i = 1)^n K(x, x_i))_(i = 1)^n K(x, x_i) y_i$$

Où K(.,.) est appelée la fonction noyau ("kernel"en anglais). Il s'agit donc d'une moyenne pondérée des étiquettes observées dans l'ensemble d'entraînement D, où le poids est donné par le noyau K qu'on peut interpréter comme une fonction de similarité entre deux entrées. Les exemples (x_i, y_i)dans$D$ font donc partie des variables utilisées par le modèle pour faire sa prédiction, même après que l'apprentissage est terminé (l'algorithme d'apprentissage le plus simple consiste à uniquement mémoriser les paires (x_i, y_i), mais il serait aussi possible d'optimiser certains paramètres du noyau K, si besoin est). La II.7 montre un exemple de régression par fenêtres de Parzen en une dimension [20].

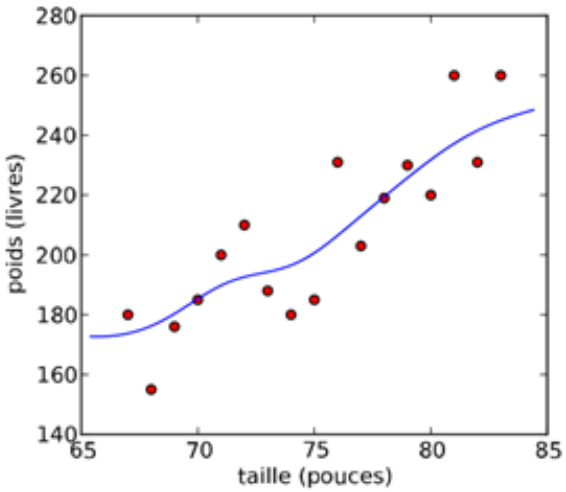

Figure II.7 – Modéle non paramétrique : régression par fenêtres de Parzen. Les données sont les mêmes que dans l'exemple de la régression linéaire II.6

II.6 Conclusion

L'apprentissage automatique (ML) est un sujet vaste en évolution permanente. Les algorithmes qu'il met en oeuvre ont des sources d'inspiration variées qui vont de la théorie des probabilités aux intuitions géométriques en passant par des approches heuristiques.
Dans le prochain chapitre, nous serons consacrés aux l'alghorithmes les plus importants de ML, a savoir Réseaux de neurones artificiels(RNC)

Chapitre III

Réseaux de neurones artificiels

III.1 Introduction

Nous allons présenter dans ce chapitre la méthode utilise dans de ce travail à savoir : Les réseaux de neurones convolutionnels(CNN).
Mais nous devons d'bord parler de Les réseaux de neurones artificiels et pourquois nous avons eu recours à cet pour abtenir des meilleurs résultats. Depuis plusieurs années ont été étudiées les méthodes de reconnaissance à base de réseaux de neurones dans le but de réaliser des performances proche de celles observées chez l'humain. Ces réseaux de neurones sont composés de plusieurs éléments (ou cellules) de calcul opérant en parallèle et arrangés à la manière des réseaux de neurones biologiques. Nous allons présentés aussi dans ce chapitre un modèle d'apprentissage automatique, qui appartient à une catégorie de modèles dits profonds. Ces modèles profonds ont suscités beaucoup d'intérêt lors des deux dernières décennies, surtout pour des applications 2D (classification d'images). (Cnn) Cet modèle ont été utilisé avec succè dans plusieurs applications de traitement d'images fixes.

III.2 Réseau de neurone artificiel :

Le réseau de neurone artificiel est un système de traitement de l'information nés il ya une cinquantaine d'années et sont toujours en cours de développement. Ce système est inspiré du fonctionnement de neurones humain. Ils se composent d'un grand nombre d'unités de traitement hautement reliées travaillent ensemble pour exécuter une tache de classification donnée .Le réseau de neurone caractérisé par leur robustesse au bruit et par leur capacité de mémorisation, de généralisation et d'une certaine forme d'apprentissage[29]

III.2.1 Du neurone biologique vers neurone artificiel :

Le neurone biologique se compose a le corps cellulaire qui permettant de traiter les informations, les dendrites ce qui les capteurs des signaux, les axones qui ce sont les lignes des transmissions des signaux entre les neurones, et les synapses ce qui la force de connexion entre la dendrite et l'axone de deux neurones

Le réseau de neurone artificiel ressemble le neurone biologique en trois aspect : la connaissance est acquise par le réseau par un processus d'apprentissage, La force de connexion reliées

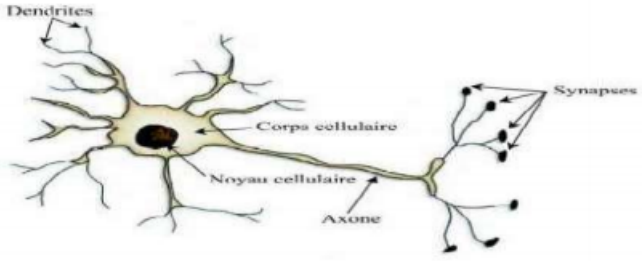

Figure III.1 – Représentation d'un neurone biologique.[28]

ensemble, connues sous le nom de poids synaptiques, et chaque neurone a un état interne appelé seuil ou fonction d'activation utilisée pour classifier les vecteurs [28]. On peut définir le neurone artificiel par les trois éléments suivants :

— La fonction d'entrée totale qui définit le prétraitement effectué sur les entrées.

— La fonction d'activation de neurone qui définit son état interne en fonction de son entrée totale

— La fonction de sortie qui calcule la sortie du neurone en fonction de son état d'activation [34].

Généralement le réseau de neurone reçoit les données X_i sur la couche d'entrée, chacun des ces données a un poids synaptiques W_ij sure l'importance de chaque entrée. Dan la couche cachées ou couche de traitement le réseau applique une fonction appelé fonction d'activation sur la somme pondérée de W_ij X_i Le résultat exprime la sortie S de ce réseau.

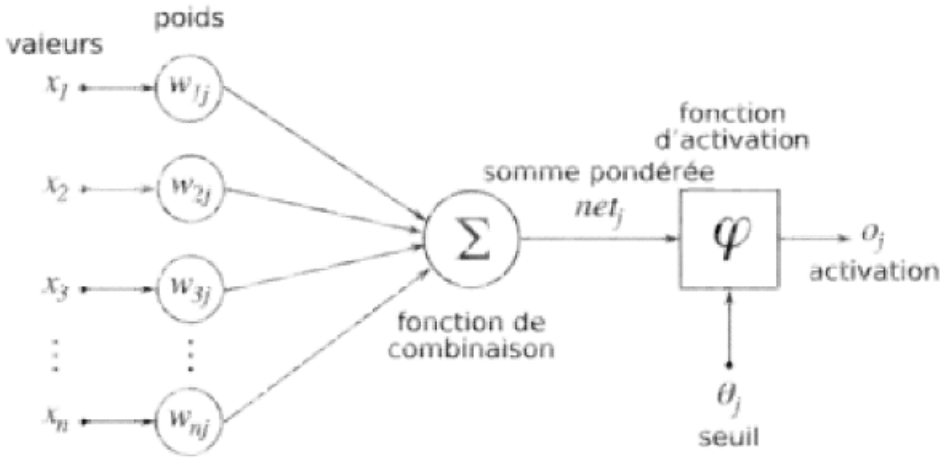

Figure III.2 – Représentation d'un neurone mathématique.[28]

Le seuil de tolérance θ_j est un paramètre critique déterminé par l'utilisateur. Pour détermine

la précision dans la réponse du réseau de neurones. On peut distinguer un réseau de neurones par ces trois caractéristiques principales :

— L'architecture du réseau qui caractérise un chemin particulier par lequel les éléments du réseau sont connectés et qui définit la direction de propagation des informations.

— La fonction d'activation des neurones qui dicte leur comportement

— L'algorithme d'apprentissage du réseau [30].

III.2.2 Architecture de réseau de neurone artificiel :

Les réseaux à couches sont les modèles connexionnistes les plus couramment utilisés. Leur architecture, organisée en couches successives, comprend une couche d'entrée, une couche de sortie et une ou plusieurs couches intermédiaires appelées couches cachées car elles ne sont pas vues de l'extérieur. Chaque couche est composée d'un certain nombre de neurones. Les connexions sont établies entre les neurones appartenant à des couches successives mais les neurones d'une même couche ne peuvent pas communiquer entre eux dans le cas des réseaux à couches. On distingue deux types de RNA : Les réseaux non bouclés et les réseaux bouclés :
Réseaux de neurones non bouclés : Dans un réseau de neurones non bouclée,l'information circulant des entrées vers les sorties sans "retour en arrière" ; si l'on représente le réseau graphiquement,le graphe d'un réseau non bouclé est acyclique : Si l'on se déplace dans le réseau, à partir d'un neurone quelconque, en suivant les connexions, on ne peut pas revenir au neurone de départ. dans le réseau, à partir d'un neurone quelconque, en suivant les connexions, on ne peut pas revenir au neurone de départ plusieurs couches dont certaines sont cachées.[32]

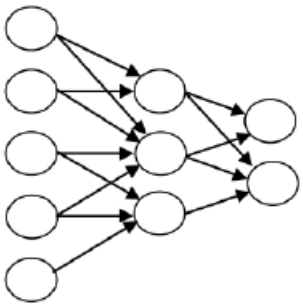

Figure III.3 – Architecture de réseau non bouclé[32]

Réseaux de neurones bouclés : Un réseau de neurones bouclé ou bien à connexions récurrentes signifie qu'une ou plusieurs sorties de neurones d'une couche aval sont connectées aux entrées des

neurones de la couche amont ou de la même couche. Ces connexions récurrentes ramènent l'information en arrière par rapport au sens de propagation défini dans un réseau non bouclé.Contrairement aux réseaux de neurones non bouclés, le graphe de connexions des réseaux de neurones bouclés est cyclique : lorsqu'on se déplace dans le réseau, en suivant le sens des connexions, il est possible de trouver au moins un chemin qui revient à son point de départ (un tel chemin est désigné sous le terme de "Cycle"). Comme pour les réseaux de neurone non bouclé, à chaque connexion d'un réseau de neurones bouclés est attaché un poids, un retard, multiple entier de l'unité de temps choisis.Un réseau de neurones bouclé à temps discret est donc régi par une (ou plusieurs) équations aux différences non linéaires, résultant de la composition des fonctions réalisées par chacun des neurones et des retards associés à chacune des connexions. [20]

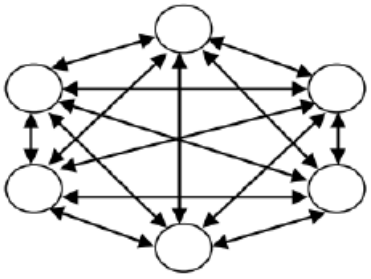

Figure III.4 – Architecture de réseau bouclé [35]

III.2.3 Les fonctions d'activation :

La fonction de transfert est en général, une fonction non linéaire monotone croissante ; Par ailleurs, les fonctions de transfert sont de qualités diverses : elles peuvent être déterministes, continues, discontinues ou aléatoires [29]. La figure suivante donne les modèles de fonctions d'activation le plus utilisées.

III.2.4 L'apprentissage :

L'apprentissage est la propriété la plus intéressante des réseaux de neurones, a pour but l'extraction des informations pertinentes à l'identification. Donc il est la phase du développement du réseau, durant laquelle le comportement de ce dernier est modifié jusqu'à l'obtention du comportement désiré [31]. Durant l'apprentissage de réseau, des certaines modifications des poids est réalisée dans le but d'avoir une meilleure réponse du réseau, la modification est atteinte

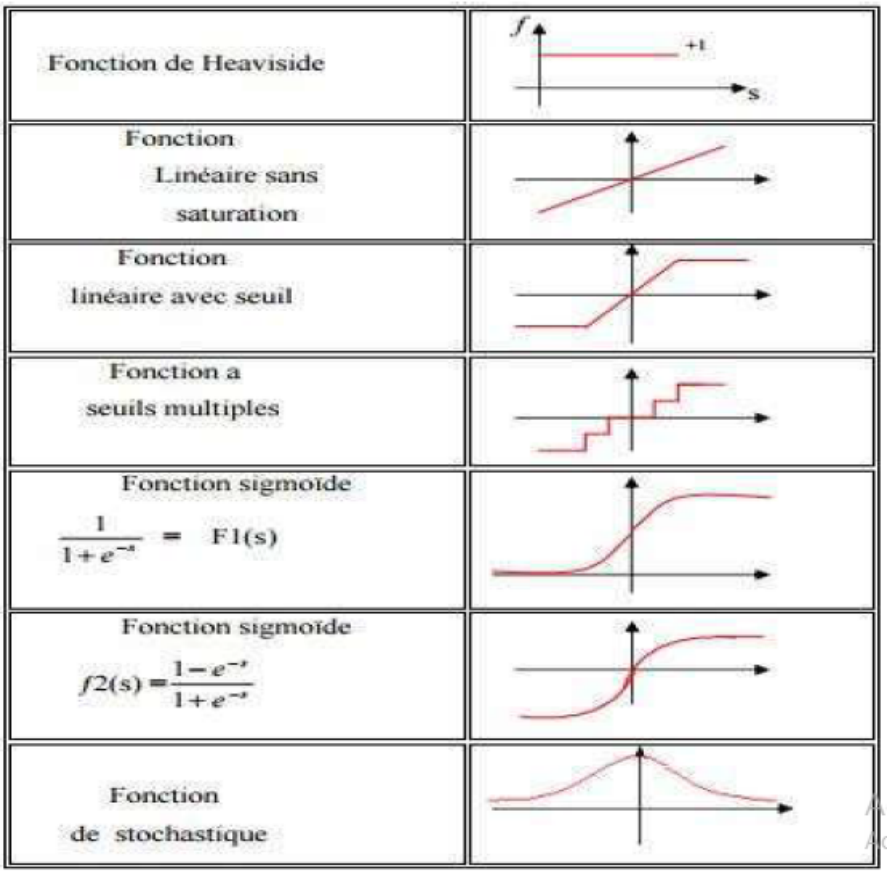

Figure III.5 – les différents types fonctions d'activations. [31]

lorsque le réseau se stabilise, il est souvent impossible de décider à priori des valeurs des poids des connexions d'un réseau pour une application donnée. A la fin de l'apprentissage les poids sont fixés, c'est alors la phase de généralisation. Le réseau peut ensuite dans un certaines mesure être capable de généraliser. C'est-à-dire de produire des résultats corrects sur de nouveaux cas qui ne lui avaient pas été présentes au cours de l'apprentissage. **Types d'apprentissage :** On peut distinguer trois types d'apprentissages :

— **Apprentissage supervisé :** est les plus utilisés. on présente au réseau des entrées et au même temps les sorties que l'on désirerait pour cette entrée. Le réseau doit alors se reconfigurer. C'est-à-dire calculer ses poids afin que la sortie qu'il donne corresponde bien à la sortie désirée.

— **Apprentissage semi-supervisé :** qui ne tiennent compte que d'une évaluation partielle ou qualitative des sorties.

— **Apprentissage non supervisé :** on présente une entrée au réseau et on le laisse évoluer librement jusqu'à ce qu'il se stabilise.

III.2.5 Type des réseaux de neurones :

Un réseau de neurones est constitué d'un assemblage d'éléments, d'unités ou de nœuds processeurs pour lequel un sous-groupe effectue un traitement indépendant et transmet le résultat à un deuxième sous-groupe et ainsi de suite (cas d'un réseau à couches multiples). Les capacités de traitement du réseau dépendent des poidsW_ij auxquels sont affectées des valeurs produisant un filtre affectant la capacité d'apprentissage du réseau. Dans un réseau de neurones, les neurones sont regroupés en couches. Habituellement, chaque neurone dans une couche est connecté à tous les neurones dans la couche précédente et la couche suivante (excepté dans la couche d'entrée et celle de sortie du réseau).L'information donnée à un réseau de neurones est propagée couche par couche de la couche d'entrée à la couche de sortie en passant par une ou plusieurs couches intermédiaires (couches cachées)[29].

III.2.5.1 Perceptron

Le perceptron est un réseau de neurone simple qui a été proposé par FRANK ROSENBLATT en 1958. Il est un réseau linéaire et monocouche ayant juste deux couches, une couche représente les entrées de système et l'autre pour les sorties. Les connexions entre ces deux couches sont modifiables et bidirectionnelles [30]. La procède d'apprentissage est supervisé et le réseau capable de résoudre des opérations logiques simples.

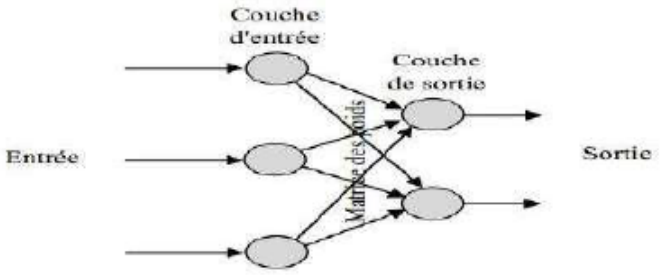

Figure III.6 – Architecture de perceptron [24]

Le perceptron prend en entrée n valeurs X1,X2,.....Xn, et calcule une sortie O, ce derniers est dépend de la somme de ces composants, pondérées par des poids réels W_n et le seuil θ La sortie O du ce réseau est sera de la forme suivante :

$$O_k = \begin{cases} 1 & if \quad W_n X_n \geq \theta_k \\ 0 & if \quad W_n X_n < \theta_k \end{cases}$$

L'algorithme d'apprentissage est la règle d'apprentissage de Hebb qui affecte le changement des poids en multipliant l'entrée d'un neurone par sa sortie et le taux d'apprentissage du réseau [33].

III.2.6 Perceptron multicouche :

Le perceptron multicouche (Multi Layer Perceptron) est une amélioration du perceptron comprenant une ou plusieurs couches cachées qui font le réseau MLP un outil robuste pour les tâches complexes. Il est largement utilisé pour la décision dans le domaine de reconnaissance faciale. Les réseaux MLP sont généralement des réseaux entièrement connectés. Les neurones de la première couche reçoivent le vecteur d'entrée, ils calculent leurs sorties qui sont transmises aux neurones de la seconde couche qui calculent eux même leurs sorties et ainsi de suite de couche en couche jusqu'à celle de sortie [33]. Dans le réseau MLP il n'y a aucune connexion entre les cellules d'une même couche. Les perceptrons multicouches sont utilisés avec apprentissage supervisé et aussi avec la technique de rétro- propagation (back-propagation) pour la correction de l'erreur.

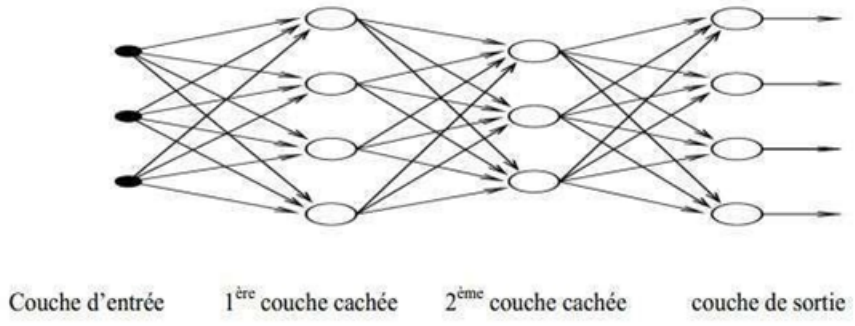

Figure III.7 – Architecture de perceptron multicouche

III.2.7 Le réseau à fonction de base radiale :

Le réseau à fonction de base radiale RBF (Radial Basis Function) est basé sur une architecture qui s'organise en deux couches seulement, une couche cachée et une couche de sortie. La couche cachée, constituée des noyaux (ou neurones) RBF effectue une transformation non linéaire de l'espace d'entrée. La couche de sortie calcule une combinaison linéaire des sorties de la couche cachée. Chaque noyau élémentaire calcule la distance entre l'entrée et son centre qu'il passe ensuite dans un non linéarité réalisée par une fonction d'activation qui est généralement de type gaussienne. La valeur que prend la sortie du noyau gaussien est d'autant plus importante que l'entrée est plus proche de son centre et tend vers zéro, lorsque la distance entrée centre devient importante. La sortie du réseau RBF est donnée par :

$$Y_i = \sum_{k=1}^{N1} W_{kj}\, \varphi_k(\|x \quad c_k\|) \text{et } \varphi(\varepsilon) = \exp\left(\frac{\varepsilon^2}{2\eta^2}\right)$$

. dénote la norme euclidienne, x le vecteur d'entrée, $C_k$est le centre associé au noyau k. N1 le nombre de noyaux de la couche cachée et W_kj les poids associés à la couche de sortie. Le paramètre ‘ ’ permet de contrôler la vitesse de décroissance de la fonction .

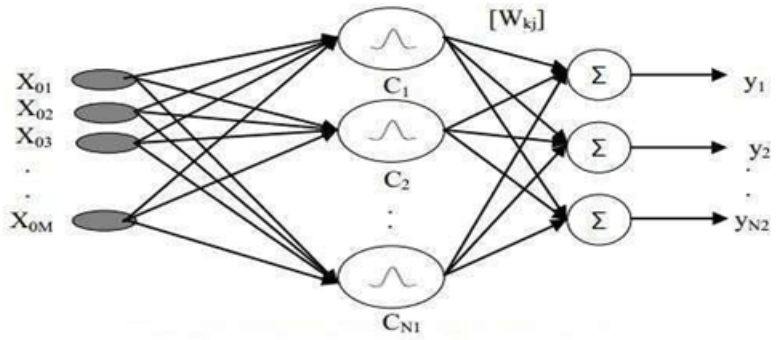

Figure III.8 – architecture du réseau RBF [34]

L'apprentissage des réseaux RBF est composé d'une règle non supervisée pour l'adaptation des centres et une règle d'apprentissage supervisé pour l'adaptation des poids. Le choix de la position des centres et le nombre de neurones reste généralement arbitraire [34].

III.3 Les réseaux de neurones convolutionnels

III.3.1 Pourquoi CNN ?

CNN, comme les réseaux de neurones, sont constitués de neurones dont le poids peut être appris et les préjugés. Chaque neurone reçoit plusieurs entrées, prend une somme pondérée sur eux, passez-le à travers une fonction d'activation et répond avec une sortie. L'ensemble du réseau a une fonction de perte et tous les trucs et astuces que nous avons développés pour les réseaux de neurones s'appliquent toujours sur les CNN.[37]

La seule différence notable entre les CNN et les RN traditionnels est que Les CNN sont principalement utilisés dans le domaine de la reconnaissance de formes avec Cela nous permet de coder des caractéristiques spécifiques à l'image dans l'architecture, rendre le réseau plus adapté aux tâches axées sur l'image

Ce choix a été motivé principalement par l'incorporation implicite d'une fonctionnalité phase d'extraction et a été utilisé avec succès dans de nombreuses applications. L'une des principales limites des formes traditionnelles d'IA est qu'elles ont tendance à avoir des difficultés avec la complexité informatique nécessaire pour calculer données d'image.

C'est la raison de la mise en œuvre de CNN, les propriétés que les CNN ont, par exemple, l'extraction de caractéristiques les rendent plus e cient quand manipulation des images.[37]

III.3.2 Les réseaux de neurones convolutionnels (CNN) :

Les réseaux de neurones convolutionnels (CNN pour Convolutional Neural Networks) proposés initialement par Le Cun. Ce choix a été motivé principalement par ce qu'il intègre implicitement une phase d'extraction de caractéristiques et il a été utilisé avec succès dans de nombreuses applications[**ref8**]. Ils sont réputés pour leur robustesse aux faibles variations d'entrée, le faible taux de prétraitement nécessaires à leur fonctionnement.
Le CNN est un réseau de neurone multicouche qui est spécialisés dans des tâches de reconnaissance de forme. Ces réseaux ont été inspirés par les travaux de Hubel et Wiesel sur le cortex visuel chez les mammifères qui combine trois idées principales :

— les champs récepteurs locaux.

— les poids partagés.

— le sous-échantillonnage.

L'architecture de CNN repose sur plusieurs réseaux de neurones profonds consistant en une succession de couches de convolution et d'agrégation (pooling) est dédié à l'extraction automatique de caractéristiques, tandis que la seconde partie, composée de couches de neurones complètement connectés, est dédiée à la classification. [37] [39]

Chaque cellule des couches de convolution est connectée à un ensemble de cellules regroupées dans un voisinage rectangulaire sur la couche précédente. Les champs récepteurs locaux permettent d'extraire des caractéristiques basiques. Les couches sont dites « à convolution » car les poids sont partagés et chaque cellule de la couche réalise la même combinaison linéaire (avant d'appliquer la fonction sigmoïde) qui peut être vue comme une simple convolution. Ces caractéristiques sont alors combinées à la couche suivante afin de détecter des caractéristiques de plus haut niveau. Entre deux phases d'extraction de caractéristiques, le réseau réduit la résolution de la carte des caractéristiques par un moyen de sous-échantillonnage. Cette réduction se justifie à deux titres : diminuer la taille de la couche et apporter de la robustesse par rapport aux faibles distorsions.

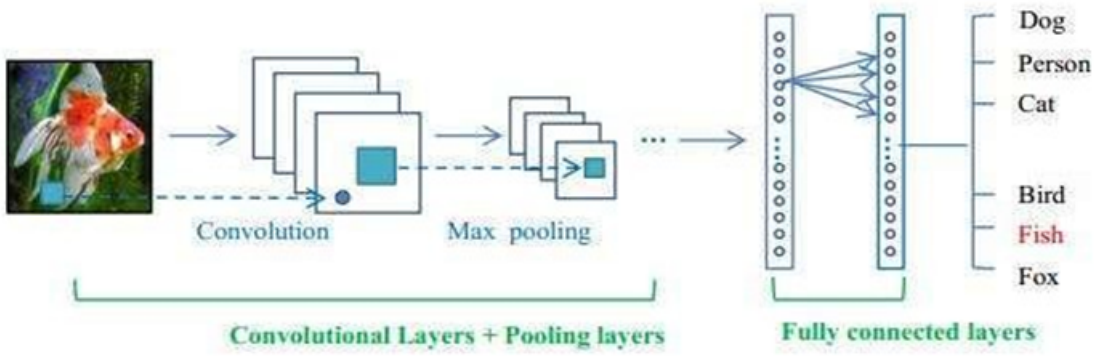

Figure III.9 – L'architecture d'un réseau de neurone convolutionnels

III.3.3 Couche de convolution :

La convolution est une opération mathématique comme l'addition et la multiplication, il est très utiles de simplifier des équations plus complexe, cette opération est largement utilisée dans le traitement du signal numérique. Lorsque on applique la convolution aux le traitement

d'image, on convoler (combiner) l'image d'entrée avec une sous-région de cette image (filtre). Le filtre est aussi connu sous le nom du noyau de convolution, il consiste en des poids de cette sous-région. La sortie de cette couche est l'image entrée avec des modifications qui est souvent appelée une carte de caractéristique (feature Map).

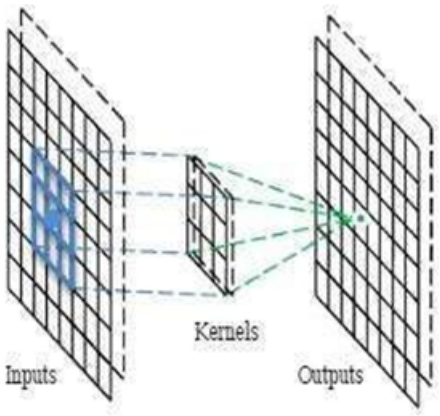

Figure III.10 – L'opération de convolution

En terme mathématique, Une couche de convolution C_i (couche i du réseau) est paramétrée par son nombre N de cartes de convolution M_ij(j 1,..., N), la taille des noyaux de convolution K_x K_y(souvent carrée), et le schéma de connexion à la couche précédente L^i Chaque carte de convolution M_jî est le résultat d'une somme de convolution des cartes de la couche précédente M_j^{i-1}par son noyau de convolution respectif. Un biais b_j^iest
ensuite ajouté et le résultat est passe à une fonction de transfert non-linéaire . Dans le cas d'une carte complètement connectée aux cartes de la couche précédente, le résultat est alors calculé par :[38]

$$M_j^i = (b_j^i + \sum_{n=1}^{n} M_j^{i-1} \quad K_j^i) \text{ [31]}$$

III.3.4 Couche de sous-échantillonnage (Pooling) :

Dans les architectures classiques de réseaux de neurones convolutionnels, les couches de convolution sont suivies par des couches de sous échantillonnage (couche d'agrégation). Cette dernière réduit la taille des cartes de caractéristique pour but de diminuer la taille de paramètre, et renvoie les valeurs maximales des régions rectangulaires de son entrée.[40] [37]

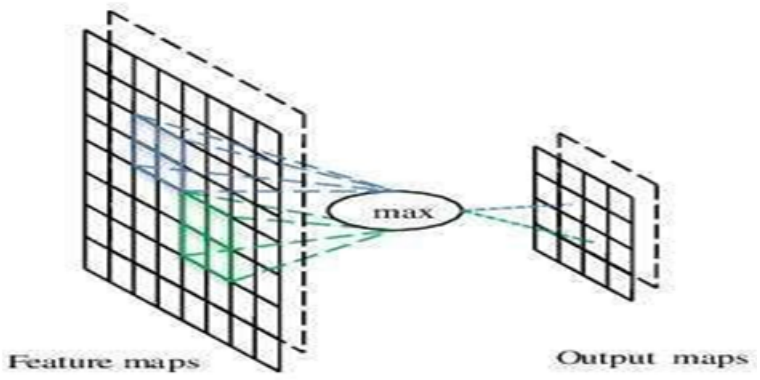

Figure III.11 – L'opération de sous échantillonnage

III.3.5 Couche entièrement connectée :

Les paramètres des couches de convolution et de max agrégation sont choisis de sorte que les cartes d'activation de la dernière couche soient de taille 1, ce qui résulte en un vecteur 1D d'attributs. Des couches classiques complètement connectées composées de neurones sont alors ajoutées au réseau pour réaliser la classification.[41] La dernière couche, dans le cas d'un apprentissage supervisé, contient autant de neurones que de classes désirées. Cette dernière couche contient N neurones (nombre des classes dans la base), et une fonction d'activation de type sigmoïde est utilisée afin d'obtenir des probabilités d'appartenance à chaque classe. [37] [41]

III.3.6 CNN Architectures

De nombreuses architectures CNN ont été utilisées dans la classification des images via les années, et chacun d'eux a maximisé la performance de l'image la classi cation à sa manière. certaines des architectures célèbres de CNN sont les Suivant [45] :

III.3.6.1 AlexNet (2012)

AlexNet utilise la fonction d'activation ReLu au lieu de tanh pour ajouter la non-linéarité, qui a accéléré la vitesse de formation (de 6 fois) et augmenté la précision. Il utilise également la régularisation des abandons (une technique évite les problèmes complexes). co-adaptations sur les données d'entraînement pour réduire le surmenage). Une autre caractéristique de AlexNet

fait en sorte que le pooling se chevauche pour réduire la taille du réseau. Il réduit les taux d'erreur parmi les premiers et les cinq premiers de 0,4% et 0,3%, respectivement

Le filet contient huit couches avec des poids ; les premiers sont convolutifs et les trois autres sont entièrement connectés. La sortie de la dernière couche entièrement connectée est transmise à un softmax à 1000 voies qui produit une distribution sur les 1000 étiquettes de classe. Notre réseau maximise la logistique multinomiale régression, ce qui équivaut à maximiser la moyenne sur l'ensemble des cas de formation de la log-probabilité de l'étiquette correcte sous la prévision Distribution. [43]

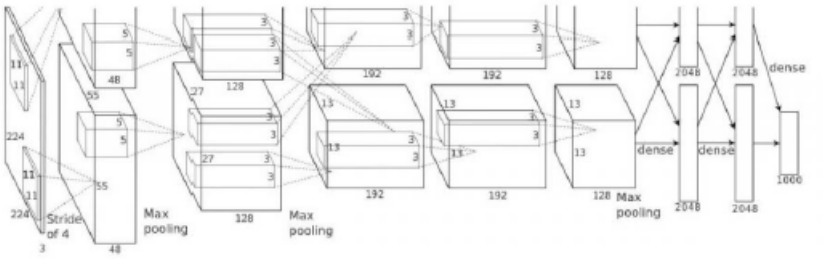

Figure III.12 – AlexNet Architecture

III.3.6.2 GoogLeNet/Inception(2014)

GoogLeNet est développé sur la base de l'idée que plusieurs connexions entre les couches sont inefficaces et contiennent des informations redondantes en raison de la corrélation qui existe entre elles. En conséquence, il utilise un "module de démarrage", une CNN, avec 22 couches dans un traitement parallèle ow et avantages de plusieurs classes auxiliaires au sein des couches intermédiaires pour améliorer la capacité de discrimination dans les couches inférieures. Contrairement aux CNN conventionnels tels que AlexNet et VGG, dans lesquels une opération de convolution ou de mise en commun peut être utilisée à chaque niveau, le module Inception pourrait bénéficier de les deux à chaque couche. De plus, des ltres (convolutions) de tailles différentes sont utilisé sur la même couche, fournissant des informations plus détaillées et extrayant modèles avec différentes tailles [44] .

Il est important de noter qu'une couche convolutionnelle 1 x 1, appelée couche de goulot d'étranglement, a été utilisé pour réduire à la fois la complexité informatique et le nombre de paramètres. Pour être plus précis, 1 x 1 couches convolutives ont été utilisées seulement avant un ltre convolutif de noyau plus grand (par exemple 3 x 3 et 5 x 5 convolutionnel) couches)

pour diminuer le nombre de paramètres à déterminer à chaque niveau (c'est-à-dire le processus de la fonctionnalité de mise en commun).

De plus, des couches convolutionnelles 1 x 1 rendent le réseau plus profond et ajoutent plus de non-linéarité en utilisant ReLU après chaque couche convolutionnelle 1 x 1. Dans ce réseau, les couches entièrement connectées sont remplacées par un pooling moyen couche. Cela diminue considérablement le nombre de paramètres depuis le début les couches connectées incluent un grand nombre de paramètres. Ainsi, ce réseau est capable d'apprendre des représentations plus profondes des entités avec moins de paramètres relatifs à AlexNet alors que c'est beaucoup plus rapide que VGG. [44] [46]

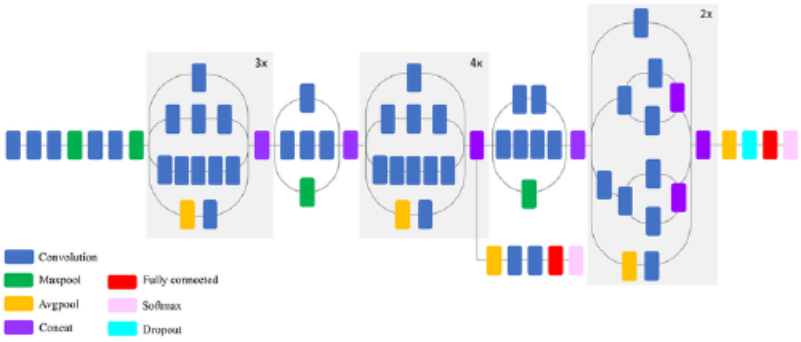

Figure III.13 – Vue comprimée de l'architecture de GoogLeNet (version 3)

III.4 Conclusion

Dans ce chapitre,nous avons présenté les réseaux de neurones (réseaux de neurones convolutionnels). Ces réseaux sont capables d'extraire des caractéristiques d'images présentées en entrée et de classifier ces caractéristiques, Ils sont fondés sur la notion de « champs récepteurs » (receptive fields), ils implémentent aussi l'idée de partage des poids qui permettant de réduire beaucoup de nombre de paramètres libres de l'architecture. Ce partage des poids permet en outre de réduire les temps de calcul, l'espace mémoire nécessaire, et également d'améliorer les capacités de généralisation du réseau.

Dans le chapitre suivant, on va proposer notre modèle et ensuite interpréter les résultats obtenus dans la phase d'apprentissage et de test .

Chapitre IV

Conception et Implémentation

IV.1 Conception

IV.1.1 Introduction

Ce chapitre est consacré à la conception et l'implémentation de notre système de reconnaissance des formes (les dattes) en utilisant une approche de de réseaux de neurones convolutionnels (CNN). Cette techniques est largement utilisés dans les problèmes de reconnaissance des formes et d'images car elle présente un certain nombre d'avantages par rapport aux autres techniques. Ce chapitre couvre aussi les bases de cette technique, y compris une description des différentes couches utilisées et finalement quelques astuces d'apprentissage profond. Notre application recevra une image qui contient un fruit (datte) comme entrée pour détecté sa catégorie.

IV.1.2 système proposé pour la reconnaissance des dattes

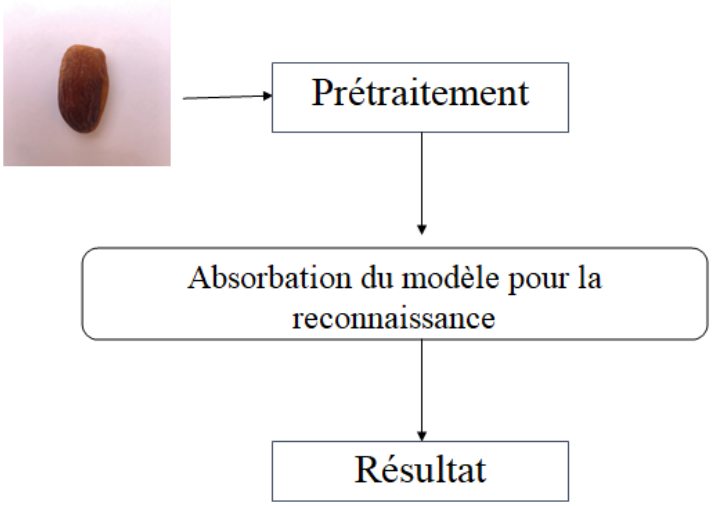

Figure IV.1 – système de reconnaissance des dattes

•prétraitment

Le traitement d'images (ou pré-traitement) regroupe l'ensemble des processus visant à améliorer les caractéristiques d'une image.

Dans cette phase on va prétraiter notre entrée(image) suivant c'est deux instruction pour la faire entrer dans le modèle de reconnaissance pour une meilleure reconnaissace :

— convertir les données en type numpy array et normalisé les images dans un intervale [0-1]

— remodler les images (reshape) 224*244

•Absorbation du modèle

Dans cette étape le modèle il va absorbé l'entrée qu'elle est maintenat sous forme d'un tableau numpy pour decider a qu'elle classe il aappartient.

•resultat

dans cette étape le modèle va nous afficher le resultat. cette phase permet de déterminer un vecteur dont les composantes caractérisent chaque type de fruit . La classification va permettre de déterminer la classe d'appartenance du fruit (datte) à l'aide de ce vecteur de caractéristiques.

IV.1.3 Architecture du système proposé

Notre système de classification de datte a partir d'une image a un nombre d'étape résumer dans la figure **IV.2**.

Nous pouvons voir que notre système comporte deux étapes principales qui sont le Front-end et le Back-end.

1.Front-end : est synonyme d'interface utilisateur, c'est ce qui se passe dans le navigateur tout ce que l'utilisateur voit et avec lequel il interagit. Dans le cadre de notre travail, l'utilisateur télécharge une image et demande une prédiction, puis obtient une réponse du label de classe de l'arrière-plan.

2.Back-end : c'est la base de notre travail, à ce stade nous allons rassembler les données des images et nettoyer les données en effectuant également une technique de prétraitement des images, d'augmentation des données et en concevant l'architecture de CNN pour la formation avec différents modèles jusqu'à ce que nous ayons un modèle précis, puis en le sauvegardant pour faire des prédictions plus tard si nous avons une demande de l'utilisateur.

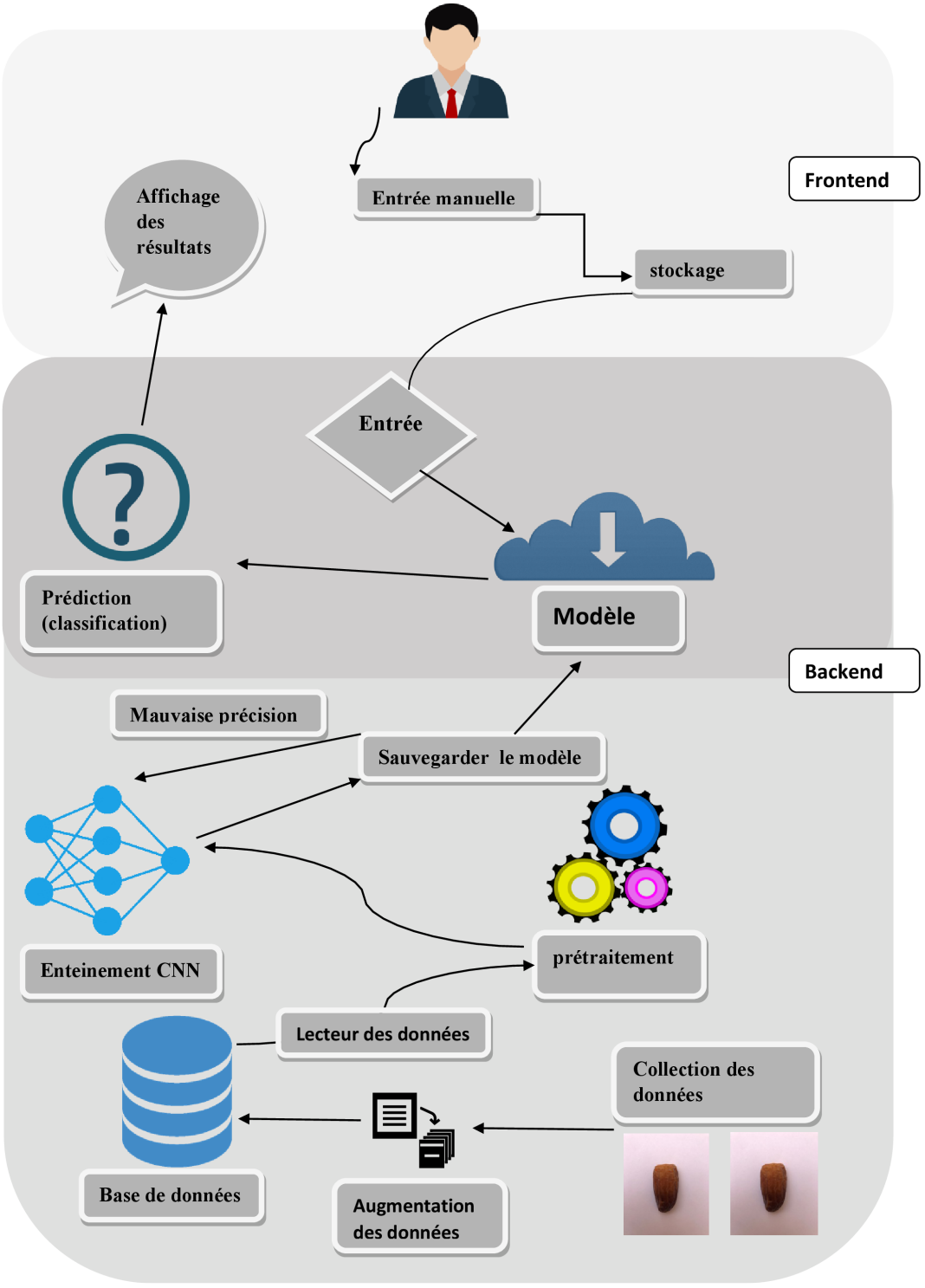

Figure IV.2 – architecture générale du système proposé

IV.1.4 Back-end

Le back-end est le cœur du système et il passe par un pipline comme suit :

1. collection et préparation des données
2. prétraitement des données
3. entrainement des modèles
4. evaluation des modèles
5. hyperparamètres
6. prédiction

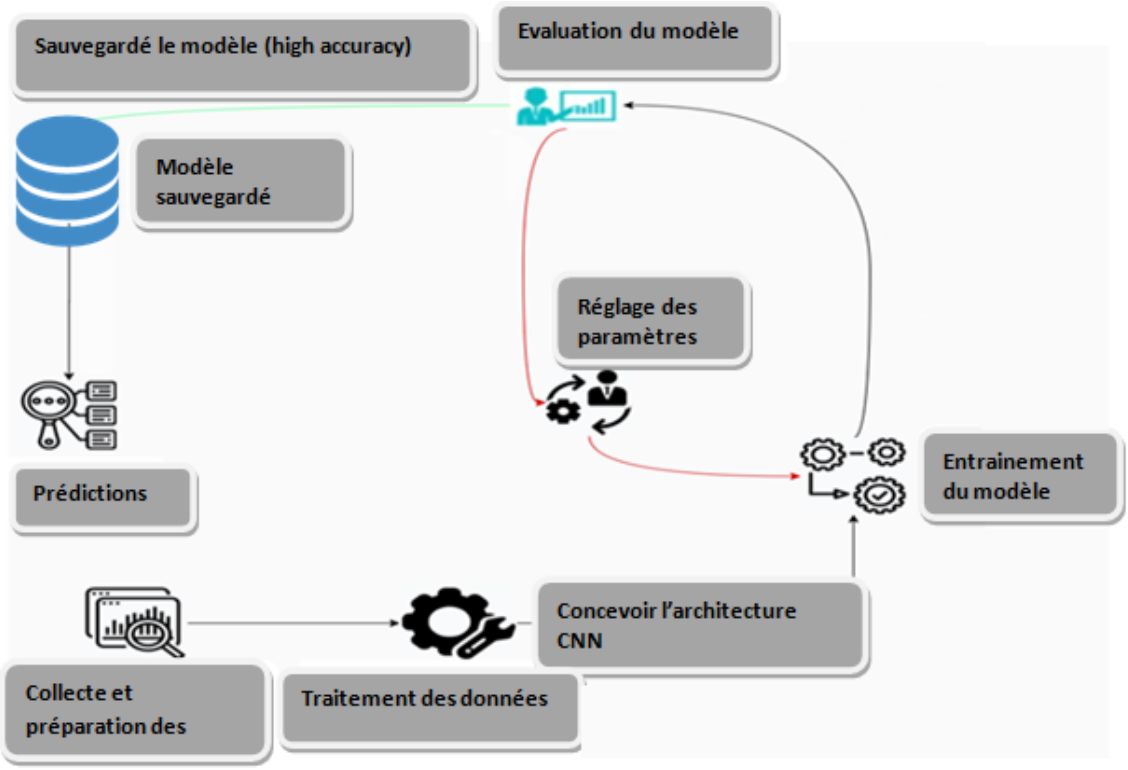

Figure IV.3 – Back-end architecture

•collection et preparation des données(Dataset)

Dans notre travail on va utilser la base de donnée DATTE que j'ai moi meme collectionné c'est photos apres une visite sur le terain (oasis tolga) j'ai reussi a avoir plusieurs échantillons de 6 type de differents et les photographier avec une une appareil photo (smartphone iphone 8 plus),parce que j'ai pas trouver une base de donné fiable sur internet donc j'ai décidier de créer la mienne.

notre base de donneés contient 1235 images de datte répartis sur 6 classes suivantes :

•**Dagla**(210 images)

•**Daghla.blanc**(193 images)

•**Daghlet nour**(201 images)

•**Gharess**(229 images)

•**Mech daghla**(208 images)

•**Ton Tbouch**(194 images)

Figure IV.4 – visualisation de ma base de données

•prétraitement des données

Le prétraitement des données est une technique qui permet d'améliorer la qualité des données afin de favoriser l'extraction d'informations utiles à partir de celles-ci. De manière simple, le prétraitement des données dans l'apprentissage automatique est une technique d'exploration des données qui transforme les données brutes en un format compréhensible et lisible.

Pourquoi avons-nous besoin du prétraitement des données ?

Lorsqu'il s'agit de créer un modèle d'apprentissage automatique, le prétraitement des données est la première étape marquant le début du processus. Généralement, les données du monde réel sont incomplètes, incohérentes, inexactes (contiennent des erreurs ou des aberrations) et manquent souvent de valeurs ou de tendances d'attributs spécifiques. C'est là que le prétraitement des données entre en jeu : il permet de nettoyer, de formater et d'organiser les données brutes,

les rendant ainsi prêtes à être utilisées dans des modèles d'apprentissage automatique.

Il y a un pipeline à suivre dans le prétraitement des données, comme le montre la figure **IV.5**.

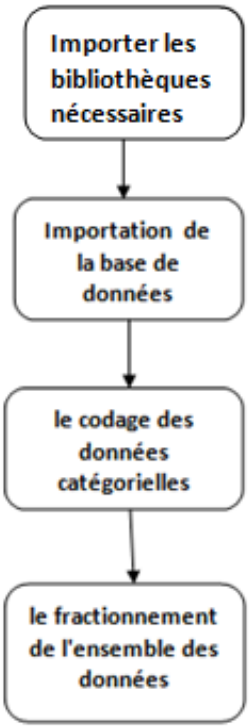

Figure IV.5 – prétraitement pipeline

Étape 1 :Importer les bibliothéques nécessaires

C'est la première étape du prétraitement des données. Les bibliothèques sont particulièrement utiles pour stocker les routines fréquemment utilisées car nous n'avons pas besoin de les relier explicitement à chaque programme qui les utilise, ce qui nous aide à traiter simplement notre ensemble de données. C'est ce que python nous fournit d'utiles bibliothèques de traitement d'images qui nous aident aussi pour le simple chargement de notre ensemble de données, nous allons les découvrir dans la section implémentation.

Étape 2 :Importation de la base des données

Nous pouvons importer l'ensemble de données de plusieurs façons, cela dépend des fichiers de format de l'ensemble de données tels que (data.npy, data.csv ...) chaque format d'ensemble de données peut être chargé avec une bibliothèque spécifique. Et il est utile de placer notre dataset dans le même répertoire de travail pour y accéder plus rapidement.

Étape 3 :le codage des données catégorielles

Les données catégorielles font référence aux informations qui ont des catégories spécifiques dans l'ensemble de données. Comme les modèles d'apprentissage machine sont basés sur des équations mathématiques, vous pouvez intuitivement comprendre que cela poserait un problème si nous

pouvions conserver les données catégorielles dans les équations, car nous ne voudrions que des nombres dans les équations. Il existe plusieur méthodes pour effectuer ce travail. C'est une représentation des variables catégorielles sous forme de vecteurs binaires. Cela nécessite d'abord que les valeurs catégorielles soient mises en correspondance avec des valeurs entières. Ensuite, chaque valeur entière est représentée par un vecteur binaire qui est constitué de toutes les valeurs zéro, sauf l'indice de l'entier, qui est marqué par un 1. Pour notre ensemble de données (images) sont déjà numériques, mais nous avons besoin de cette étape pour nos étiquettes (Dagla, Daghla.blanc, Daghlet nour, Gharess,Mech daghla et Ton Tbouch).

Étape 4 :fractionnement de l'ensemble des données

Chaque ensemble de données pour le modèle d'apprentissage automatique doit être divisé en ensemble de apprentissage/tests. L'ensemble de données d'appretnissage est utilisé pour la l'apprentissage de notre modèle. Ici, le modèle connaît le résultat. Un ensemble de test, d'autre part, est utilisé pour tester le modèle afin de vérifier la précision du modèle. Le modèle de ML utilise l'ensemble de tests pour prédire les résultats.

Habituellement, nous prenons 70 ou 80% des données pour l'apprentissage du modèle, en laissant de côté les 30 ou 20 % restants pour le test durant la phase d'apprentissage

• entrainement des modéles

Le cœur de notre processus est l'apprentissage du modèle qui exige de la patience et de l'expérimentation. L'essentiel de l'apprentissage se fait à ce stade. La première chose à faire pour notre processus d'apprentissage est de mettre en place l'architecture du modèle de réseaux neuronaux convolutifs (choix du modèle) ; ensuite, nous utiliserons nos données prétraitées pour améliorer progressivement la capacité de notre modèle .

•evaluation des modéles

Dans cette étape, nous devons évaluer notre modèle d'apprentissage avec les données du monde réel (données que le modèle n'avait jamais vues auparavant). Cependant, grâce à son apprentissage, le modèle devrait être suffisamment capable d'extrapoler les informations et de déterminer si le type de datte est Dagla ou Daghla.blanc ou Daghlet nour ou Gharess ou Mech daghla ou Ton Tbouch . Sinon, si notre modèle n'est pas assez bon avec les données réelles, nous devons ajuster les **hyperparamètres**(réglage des paramètres) ou revoir l'architecture de CNN, puis réadapter notre modèle avec les nouveaux paramètres.

— Precision(Accuracy) est une mesure d'évaluation commune pour les problèmes de classi-

fication. Il s'agit du nombre de prédictions correctes faites par rapport à l'ensemble des prédictions faites.

— Une matrice de confusion fournit une ventilation plus détaillée des classifications correctes et incorrectes pour chaque classe.

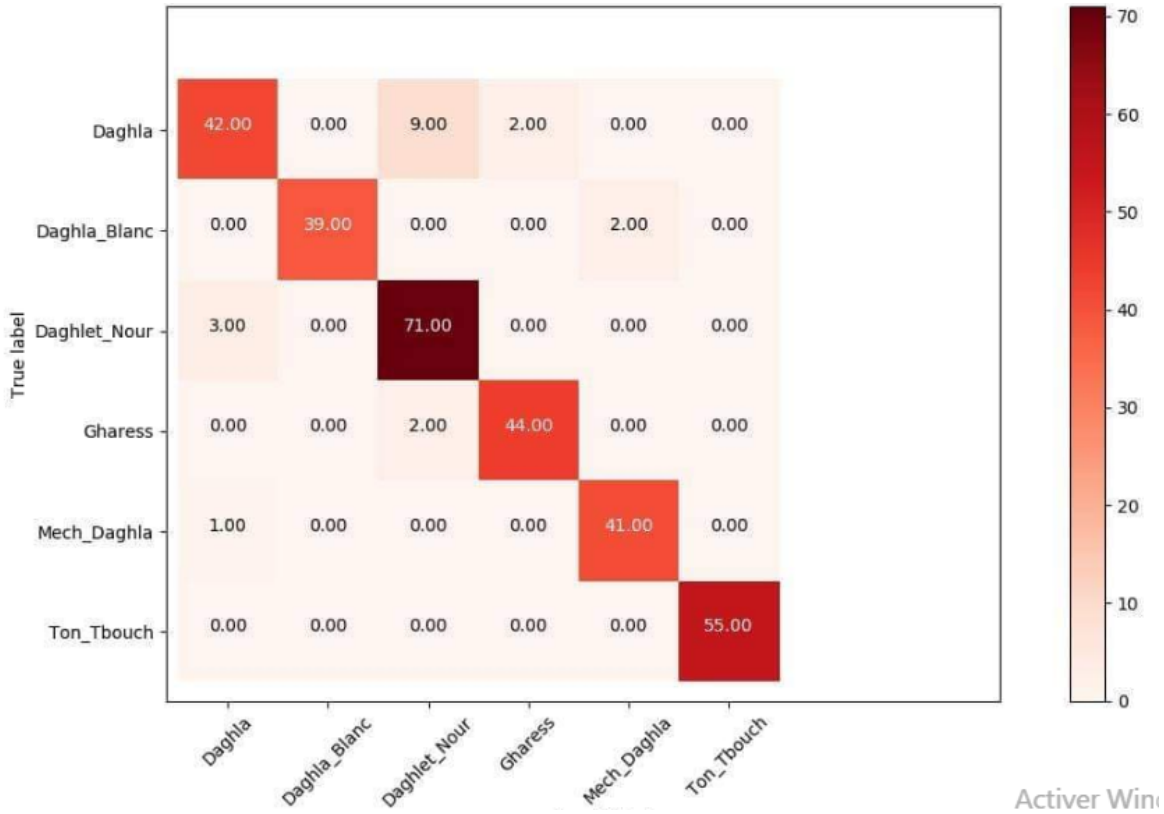

Figure IV.6 – matrice de confusion

•hyperparamètres

Le réglage des hyperparamètres est le problème du choix d'un ensemble d'hyperparamètres optimaux pour un algorithme d'apprentissage. L'objectif est de trouver une combinaison optimale d'hyperparamètres qui minimise une fonction de perte prédéfinie afin d'améliorer les performances du modèle.

hyperparamètres de CNN : taux d'apprentissage, taille du lot, numéro d'époque, fonction d'activation, etc.

•prédiction

La dernière étape de notre travail est la prédiction. Sachez que nous considérons que notre modèle est prêt pour des applications pratiques. C'est le point de tout ce travail, où la valeur de notre modèle CNN est réalisée.

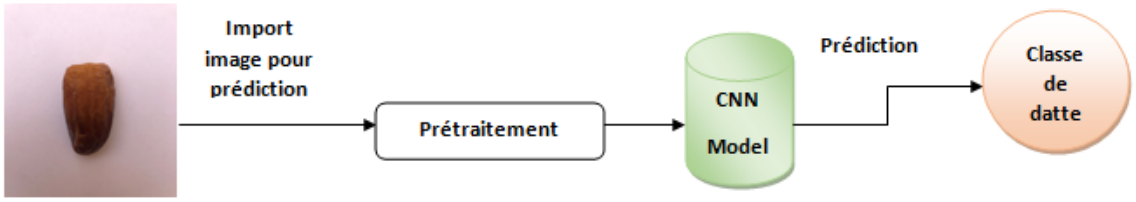

Figure IV.7 – exemple d'une prédiction

IV.1.5 L'architecture de CNN proposée

L'image en entrée est de taille 224 224, l'image passe d'abord a la première couche de convolution. Cette couche est composée de 32 filtres de taille 3 3, la fonction d'activation ReLU est utilisée pour forcer les neurones a retourner des valeurs positives.

Cette convolution produit 32 features maps de taille 111x111 chacune. Ensuite, les 32 feature maps sont présentées en entrée a la deuxieme couche de convolution qui est composée aussi de 32 filtres de taille 3 3, mais avant il passe par une couche de maxpooling 2x2.

La fonction d'activation ReLU est appliquée sur les couches de convolutions.apres deux couche de convolutions seron appliqués de suite avec une dimension de 3 3, avant que la couche max-pooling sera appliquée , la meme étape et encore appliqué avec deux couche convolutif et et une derniere max pooling finalment deux couche FC sont appliqués doté d'une fonction softamx pour la probabilté des vecteurs de classe plus sur.

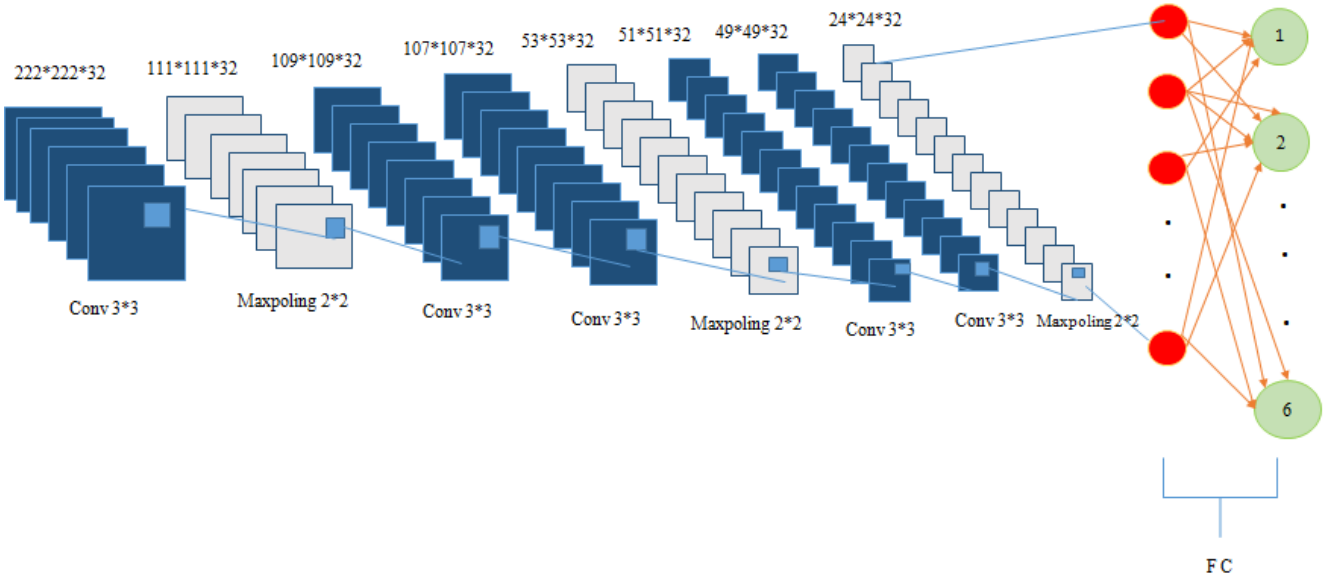

Figure IV.8 – L'architecture de CNN proposée

IV.2 Implémentation

Dans cette phase on va implémenter notre conception , on va demarrer par l'etape de l'apprentissage (entriner notre modèle) sur les 6 classes de notre fruit , avec le language de programmation pyhton et avec l'aide de certain API'

IV.2.1 Environnement de travail

pour obtenir des meilleurs résultats de reconnaissance et tant que les images sont de grande capacité de taille et resolution (3024x4032) on a besoin d'un matriel sophistiqué pour avoir un meilleur traitement (segmentation , extraction de caractéristiques,classification). Pour mettre en œuvre cette application, les matériaux ayant les caractéristiques suivantes ont été utilisés :

— processeur : Intel(R) CoreTM i7-7200U CPU @4.20 GHZ

— RAM : 32.00 GB

— SE : Windows 10 Pro

IV.2.2 Paquets et bibliothéques requis

Pour développer mon application, j'ai utilisé un environnement de programmation , et des outils et des librairies différentes pour la programmation en arrière-plan (back-end), exécutable sur plusieurs IDE (environment)

Python

Python est un langage de programmation interprété, multi-paradigme et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. C'est est un langage qui peut s'utiliser dans de nombreux contextes et s'adapter à tout type d'utilisation grâce à des bibliothèques spécialisées. Il est cependant particulièrement utilisé comme langage de script pour automatiser des tâches simples mais fastidieuses

Tensorflow

TensorFlow est un outil open source d'apprentissage automatique développé par Google.

Figure IV.9 – python logo

Figure IV.10 – Tensorflow logo

Le code source a été ouvert le 9 novembre 2015 par Google et publié sous licence Apache. initiée par Google en 2011, et est doté d'une interface pour Python et Julia. TensorFlow est l'un des outils les plus utilisés en IA dans le domaine de l'apprentissage machine.

Keras

Figure IV.11 – Keras Logo

La bibliothèque Keras permet d'interagir avec les algorithmes de réseaux de neurones profonds et d'apprentissage automatique, notamment Tensorflow,Conçue pour permettre une expérimentation rapide avec les réseaux de neurones profonds, elle se concentre sur son ergonomie, sa modularité et ses capacites d'extension. Elle a été développée dans le cadre du projet ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System). Elle a été initialement écrite par François Chollet.

Numpy

Figure IV.12 – Numpy Logo

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.

Matplotlib

Matplotlib est une bibliothèque du langage de programmation Python destinée à tracer et visualiser des données sous formes de graphiques. Elle peut être combinée avec les bibliothèques

Figure IV.13 – matplotlib logo

python de calcul scientifique NumPy et SciPy.
Plusieurs points rendent cette bibliothèque intéressante :

— Export possible en de nombreux formats matriciels (PNG, JPEG...) et vectoriels (PDF, SVG...).

— Documentation en ligne en quantité, nombreux exemples disponibles sur internet.

— Forte communauté très active.

•Pour ce qui concerne l'environnement de developpement j'ai utilsé L'IDE

Pycharme

Figure IV.14 – Pycharme Logo

PyCharm est un environnement de développement intégré utilisé pour programmer en Python. Il permet l'analyse de code et contient un débogueur graphique. Il permet également la gestion des tests unitaires, l'intégration de logiciel de gestion de versions, et supporte le développement web avec Django. Développé par l'entreprise tchèque JetBrains, c'est un logiciel

multi-plateforme qui fonctionne sous Windows, Mac OS X et Linux. Il est décliné en édition professionnelle, diffusé sous licence propriétaire, et en édition communautaire diffusé sous licence Apache..

IV.2.3 importation des librairies

La premiére étape est d'importer des bibliothéques python pour commencer notre apprentissage approfondi.

```
import matplotlib.pyplot as plt
import numpy as np
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dropout, Flatten, Dense
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint
import time
```

IV.2.4 Importation de la base de données et prétraitement

Dans notre cas , la base de donnée n'est pas un fichier docs ou bien un csv ,c'est un dossier qui contient des fichiers qui represente les 6 classes , donc on va essayer de les fracturées on deux categories , entrainement et test (cross validation), et aussi assurer que chaque image est assicier à sa label correcte.
Et comme on a mentionné dans la partie conception on doit prétraiter notre dataset avant de l'envoyer au CNN suivant 3 étapes :

— Normalisation des données : nous devons redimensionner nos images de la plage [0-255] à la plage [0-1] en utilisant la méthode de normalisation min-max en divisant la valeur de chaque pixel de l'image par 255.

— ncodage des étiquettes catégorielles : dans nos fichiers npy d'étiquettes, nous avons des valeurs catégorielles et c'est un problème dans les modèles d'apprentissage machine car ce sont des modèles mathématiques, nous devons donc encoder nos étiquettes. Nous utilisons

OneHotEncoder de la bibliothèque Scikit-learn, la sortie serait un vecteur binaire pour chaque classe.

— Remodelage des images d'entrée : nos images avec entrée (224x224x3), avant de les envoyer à CNN pour la formation, nous devons les remodeler en tenseurs 4D. Les données d'entrée ont donc une forme de (nombre d'échantillons, 224, 224, 3), où la première dimension représente la taille du lot de l'image.

```
train_data_path = 'C:/Users/HP/PycharmProjects/Abdelatif/Training'
validation_data_path = 'C:/Users/HP/PycharmProjects/Abdelatif/validation'
img_width, img_height = 224, 224
batch_size = 64
steps_epoch = 20
steps_validation = 5
epochs = 100
classes_num = 6

train_datagen = ImageDataGenerator(
    rescale=1. / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

test_datagen = ImageDataGenerator(rescale=1. / 255)

train_generator = train_datagen.flow_from_directory(
    train_data_path,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical')

validation_generator = test_datagen.flow_from_directory(
    validation_data_path,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical')
```

IV.2.5 Architecture du CNN

Puisque notre ensemble de données est maintenant prêt, nous allons concevoir une architecture de modèle de réseau neuronal convolutif en utilisant le modèle séquentiel de Keras avec des configurations de réglage comme indiqué dans le code suivant :

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),input_shape=(img_height, img_width, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu'))
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu'))
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(512, activation='relu'))img_width, img_height = 224, 224
batch_size = 64
steps_epoch = 20
steps_validation = 5
epochs = 100
classes_num = 6

model.add(Dropout(0.7))
model.add(Dense(classes_num, activation='softmax'))
```

Aprés plusieurs tentatives et manupilations , je trouve que cette architecture et la meilleure pour donner des meilleurs resultats , on executant le code en bas on peut avoir une vue globale sur notre modéle avant le lancemant de l'apprentissage.

```
model = load_model(model_path)
print(model.summary())
```

La prochaine chose à faire est de commencer notre session de d'apprentissage,comme suit :

```
history = model.fit_generator(
    train_generator,
    steps_per_epoch=steps_epoch,
    epochs=epochs,
    callbacks= my_callbacks,
    validation_data=validation_generator,
    validation_steps=steps_validation)

model.save('model/cherif.h5')
```

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 222, 222, 32)	896
max_pooling2d_1 (MaxPooling2	(None, 111, 111, 32)	0
conv2d_2 (Conv2D)	(None, 109, 109, 32)	9248
conv2d_3 (Conv2D)	(None, 107, 107, 32)	9248
max_pooling2d_2 (MaxPooling2	(None, 53, 53, 32)	0
conv2d_4 (Conv2D)	(None, 51, 51, 32)	9248
conv2d_5 (Conv2D)	(None, 49, 49, 32)	9248
max_pooling2d_3 (MaxPooling2	(None, 24, 24, 32)	0
flatten_1 (Flatten)	(None, 18432)	0
dense_1 (Dense)	(None, 512)	9437696
dropout_1 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 6)	3078

Figure IV.15 – Architecture du CNN

IV.2.6 Evaluation du model

Aprés la phase d'apprentissage qu'elle a durée 22h on a obtenu les resultats suivants : :

Precision : 0.93

Loss : 0.17

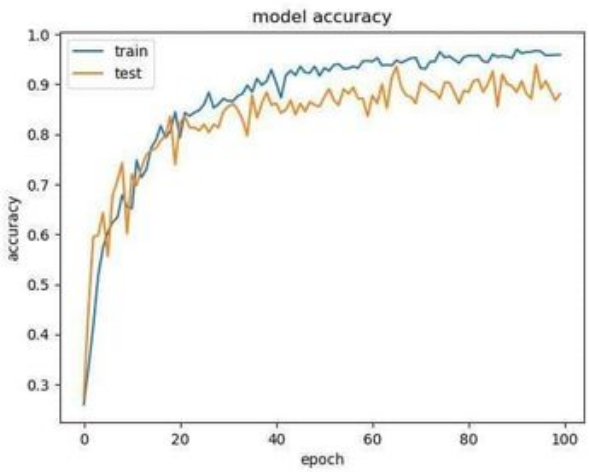

Figure IV.16 – Precision

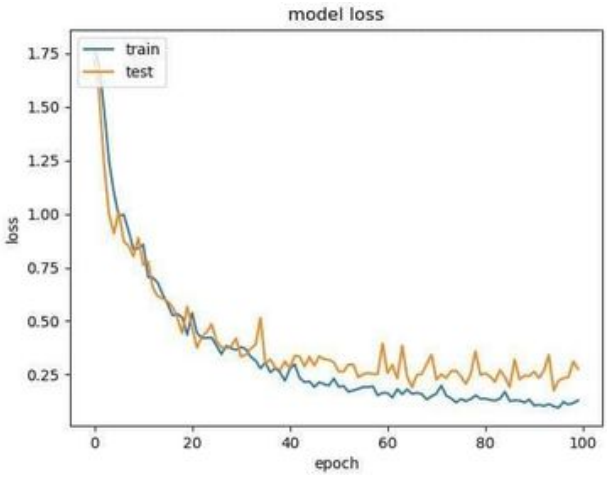

Figure IV.17 – Loss

.

IV.3 Test et résultat

Dans cette partie on va essayer de tester notre modele sur des nouvelles photos de datte pour voir dans quelle classe il va les classifier et si c'est correct.

On va tester notre système avec une photo de la classe **Mech daghla** et voir s'il va donner la bonne prédiction.

Figure IV.18 – Mech daghla

On va tester notre système avec la photo en dessus avec le code Ci-dessous :

```python
from keras.models import load_model
import cv2
from keras.preprocessing.image import ImageDataGenerator
model_path = 'model/checkPoint0.chkpt'
test_path = 'C:/Users/HP/PycharmProjects/Abdelatif/testing'
validation_data_path = 'C:/Users/HP/PycharmProjects/Abdelatif/validation'
model = load_model(model_path)
img_width, img_height = 224, 224
img_class = ['Daghla_1', 'Daghla_Blanc', 'Daghlet_Nour', 'Gharess', 'Mech_daghla', 'Ton_Tbouch']

def predict():
    test_datagen = ImageDataGenerator(rescale=1. / 255)
    test_generator = test_datagen.flow_from_directory(
        directory=test_path,
        target_size=(img_width, img_height),
        batch_size=10,
        class_mode=None,
        color_mode='rgb',
        shuffle=False,
        save_format='jpg'
    )
    test_generator.reset()
    images = test_generator[0]
    sample0 = images[7]  # choose from 0 to 9 ( number of images in test folder)
    sample = sample0.reshape(1, img_width, img_height, 3)
    pred = model.predict(sample)
    print(pred)
    predicted_class_indices = pred.argmax()
    print(img_class[predicted_class_indices])
    predict_result= img_class[predicted_class_indices]

    image = cv2.cvtColor(sample0, cv2.COLOR_BGR2RGB)
    cv2.putText(image,str(predict_result),(5,15),cv2.FONT_HERSHEY_SIMPLEX,0.5,(0,180,0),1)
    ct = cv2.resize(image,(300,300))
    cv2.imshow('testing', ct)
    cv2.waitKey(0)


predict()
```

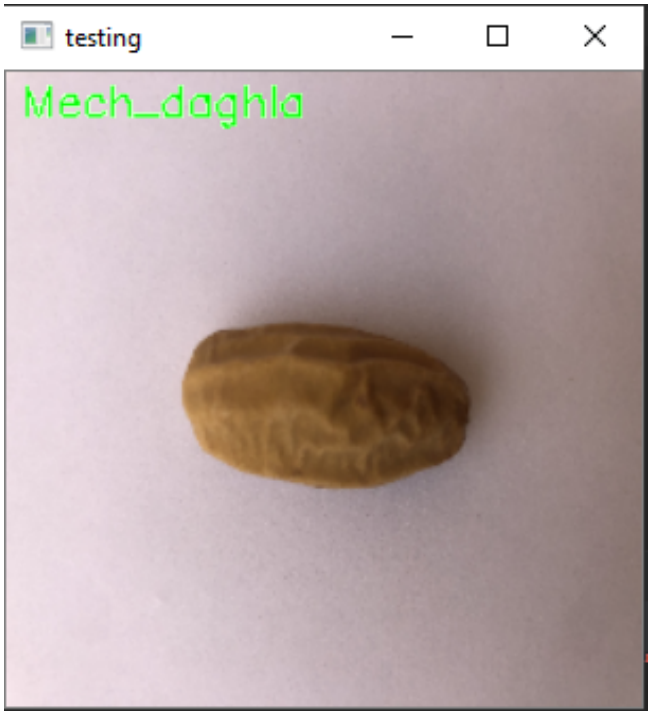

Figure IV.19 – Mech daghla prédiction

Conclution générale

Au cours des dernières années, principalement en raison des progrès de l'apprentissage en profondeur, plus de Sur les réseaux spécifiquement convolutifs, la qualité de la reconnaissance d'image et de la détection d'objets a considérablement augmenté. Une des nouvelles encourageantes est que la plupart de ces progrès ne sont pas seulement le résultat d'un matériel plus puissant, De plus grands ensembles de données et de plus grands modèles, mais principalement une conséquence de nouvelles idées, ainsi que des architectures de réseau améliorées. Aucune des nouvelles sources de données ont été utilisés, par exemple, pour les meilleures contributions au concours ILSVRC, en plus du jeu de données de classement du même concours au le même temps. Cela nous motive et nous encourage à entreprendre ce voyage d'apprentissage en profondeur afin de surmonter les défis. Le but de ce mémoire est de mettre en œuvre une application de reconnaissance de formes capable de reconnaître le motif (fruit datte) en utilisant le réseau de neurones convolutionnels .

Bibliographie

[1] ‘’Mise au Point d’une Application de Reconnaissance de Formes’’, Mémoire de fin d’études pour l’obtention du diplôme de Master en Informatique, DJABEUR DJEZZAR Mohammed Rafik, BENKADA Feth-a

[2] Watanabe, Satosi. *Pattern recognition: human and mechanical*. John Wiley & Sons, . Inc., 1985.

[3] **Theodoridis et Koutroumbas,** S. Theodoridis, K. Koutroumbas, “Pattern . . Recognition, Second Edition”, Academic *Press, Elsevier*, 2003.

[4] https://interstices.info/jcms/p_88807/marvin-minsky-un-des-cerveaux-de-l-intelligence-artificielle

[6] https://www.interstices.info/jcms/c_5952/histoire-du-traitement-d-images

[7] V. Bjorn, “One Finger at a Time: Best Practices for Biometric Security,” Banking Information Source (Document ID: 1697301411), April, 2009 .

[8] **Chung et al,** K.W. Cheung, J.T. Kwok, M.H. Law, K.C. Tsui,"Mining customer product ratings for personalized marketing", *Decision Support Systems*, vol. 35, issue 2, pp. 231-243, 2003

[9] **Kaastra et Boyd,** I. Kaastra, M. Boyd, "Designing a neural network for forecasting financial and economic time series", *Neurocomputing*. Vol. 10, no. 3, pp. 215-236. 1996.

[10] **Gupta et al,** S.K. Gupta, W.C. Regli, D.S. Nau, "Manufacturing Feature Instances: Which Ones to Recognize?", *Symposium on Solid Modeling and Applications*, pp. 141-152, 1995.

[11] **Hippert et al,** H.S. Hippert, C.E. Pedreira, R.C. Souza, "Neural networks for short-term load forecasting: a review and evaluation", *IEEE Transactions on Power Systems*, vol. 16, Part 1, pp. 44-55, 2001.

[12] **Munich et Perona,** M.E. Munich, P. Perona, "Visual Input for Pen-Based Computers", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, issue 3, 2002.

[13] **Yang et al,** M.H. Yang, D.J. Kriegman, N. Ahuja, "Detecting Faces in Images: A 51

Survey", *IEEE Transactions On Pattern Analysis And Machine Intelligence PAMI*, vol.24; part 1, pp. 34-58, 2002.

[14] **Woznica et Menal,** P. Woznica, M. Mennal, "Reconnaissance des formes", 2003.

[15] **Theodoridis et Koutroumbas,** S. Theodoridis, K. Koutroumbas, “Pattern Recognition, Second Edition”,
Academic Press, Elsevier, 2003.

[16] Hoffman, Richard, and Anil K. Jain. "Segmentation and classification of range images." *IEEE transactions on pattern analysis and machine intelligence* 5 (1987): 608-620.

[17] **J.P. Cocquerez et S. Philipp**, Analyse d’images : filtrage et segmentation Ed. Masson, Année 1995.

[18] **Phan Viet Anhe,** Développement d’un module de segmentation pour un système de reconnaissance biométrique basé sur l’iris, novembre 2008.

[19] Weizenbaum, J. 1966,“ELIZA-a computer program for the study of natural language communication between man and machine”, Commun. ACM, vol. 9, no 1, p. 36–45.

[20] ‘’Apprentissage machine efficace : théorie et pratique’’, par Olivier Delalleau , D´epartement d’informatique et de recherche op´erationnelle Facult´e des arts et des sciences

[21] Blumer, A., A. Ehrenfeucht, D. Haussler et M. Warmuth. 1987, “Occam’srazor”, Inf. Proc. Let., vol. 24, p. 377–380.

[22] Watson, G. S. 1964, “Smooth regression analysis”, Sankhya - The Indian Journal of Statistics, vol. 26, p. 359–372.

[23] Solomonoff, R. J. 1964,“A formaltheoryof inductive inference”, Information and Control, vol. 7, p. 1–22, 224–254. Sutton, R. et A. Barto. 1998, Reinforcement Learning : An Introduction, MIT Press.

[24] Kolmogorov, A. N. 1965, “Three approaches to the quantitative definition of information”, Problems of Information and Transmission, vol. 1, no 1, p. 1–7.

[25] Li, M. et P. Vit´anyi. 2008, An Introduction to Kolmogorov Complexity and Its Applications, 3e ´ed., Springer, New York, NY.

[26] Bellman, R. 1961, Adaptive Control Processes : A Guided Tour, Princeton University Press, New Jersey.

[27] Nadaraya, E. A. 1964,“On estimating regression”, Theory of Probability and its Applications, vol. 9, p. 141–142.

[28] Kaastra, Iebeling, and Milton Boyd. "Designing a neural network for forecasting financial." *Neurocomputing* 10 (1996): 215-236.

[29] Hippert, Henrique Steinherz, Carlos Eduardo Pedreira, and Reinaldo Castro Souza. "Neural networks for short-term load forecasting: A review and evaluation." *IEEE Transactions on power systems* 16.1 (2001): 44-55.

[30] Coulibaly, Paulin, François Anctil, and Bernard Bobée. "Prévision hydrologique par réseaux de neurones artificiels: état de l'art." *Canadian Journal of civil engineering* 26.3 (1999): 293-304.

[31] Ganascia, Jean-Gabriel, and N. I. V. E. A. U. DE LECTURE. "Marvin Minsky: un des cerveaux de l’intelligence artificielle." (2016).

[32] Martin, Philippe. *Réseaux de neurones artificiels: Application à la reconnaissance optique de partitions musicales*. Diss. 1992.

[33] Borne, Pierre, Mohamed Benrejeb, and Joseph Haggège. *Les réseaux de neurones: présentation et applications*. Vol. 15. Editions OPHRYS, 2007.

[34] Koiran, P. (1993). *Puissance de calcul des réseaux de neurones artificiels* (Doctoral dissertation, Lyon 1).

[35] Solaiman, Basel, and Lepage Richard. *Les réseaux de neurones artificiels et leurs applications en imagerie et en vision par ordinateur*. 2003.

[36] Touzet, Claude. *les réseaux de neurones artificiels, introduction au connexionnisme*. 1992.

[37] Mokri, Mohammed Zakaria. *Classification des images avec les réseaux de neurones convolutionnels*. Diss. 09-01-2018, 2017.

[38] Buyssens, Pierre, and Abderrahim Elmoataz. "Réseaux de neurones convolutionnels multi-échelle pour la classification cellulaire." 2016.

[39] Du Terrail, Jean Ogier. *Réseaux de neurones convolutionnels profonds pour la détection de petits véhicules en imagerie aérienne*. Diss. Normandie Université, 2018.

[40] Pibre, Lionel, et al. "Étude des réseaux de neurones sur la stéganalyse." 2016.

[41] Toufik, LAKHDARI Ahmed, and A. B. B. A. C. I. Salih. "La reconnaissance des caractères arabes manuscrits par les réseaux des neurones convolutionnels." (2017).

[42] Savard, F. (2012). Réseaux de neurones à relaxation entraînés par critère d'autoencodeur débruitant.

[43] Douillard, Arthur, Yifu Chen, and Matthieu Cord. "Visualisation des réseaux de neurones." (2019).

[44] Douillard, Arthur, Yifu Chen, and Matthieu Cord. "Réseaux convolutionnels pour l’image."

[45] Robert, Thomas, et al. "Réseaux convolutionnels pour l’image."

[46] Marty, Jean-Marc, et al. "Analyse d’opinions de tweets par réseaux de neurones convolutionnels." *Actes de DEFT, Caen, France: TALN* (2015).