

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mohamed Sadik Benyahia de Jijel

Faculté des Sciences Exactes et informatique

Département d'Informatique



Mémoire de fin d'étude
pour obtention du diplôme Master de
Recherche en Informatique
Option : Intelligence Artificielle

Thème

**Reconnaissance d images par les réseaux de neurones
convolutifs**

Préparé par

Hani Zerzaihi et Fouad Zarour

Encadré par

Mr Later Azzedine

Année Universitaire 2019/2020

Remerciements

Je tiens à remercier en premier lieu le Dieu le tout puissant et miséricordieux, qui m'a donné la force et la patience d'accomplir ce modeste travail.

Le travail présenté dans ce mémoire a été effectué sous la direction de Mr **Later Azzedine** à qui je tiens à adresser mes plus vifs remerciements, pour sa patience, sa disponibilité et surtout ses judicieux conseils, qui ont contribué à alimenter ma réflexion et ses encouragements lors de la réalisation de cette mémoire.

Je tiens aussi à remercier les membres du jury pour avoir accepté d'examiner et d'évaluer ce travail.

Dédicaces

Je dédie ce modeste travail

- ❖ mes chers parents, pour tous leurs sacrifices, leur amour, leur tendresse, leur soutien et leurs prières tout au long de mes études .
- ❖ A mes frères Abd elkader Djaber Walid Yasser El muTasim b Allah Djihan.
- ❖ A tout mes amies surtout Ayoub Salah eddine Oussama mouad aala eddine aymen aala eddin mouhamed el saleh omar mohamed .

Fouad Zarour

Dédicaces

Je dédie ce modeste travail

- ❖ *mes chers parents, pour tous leurs sacrifices, leur amour, leur tendresse, leur soutien et leurs prières tout au long de mes études*
- ❖ *A mes frères Hicham mouhamed Amine et Abla.*
- ❖ *A mon oncel ali pour tous leurs sacrifices, leur amour*
- ❖ *A tout mes amies surtout abd latif Mouad Ayoub Salah eddine Oussama mouad aala aymen omar halim houssam mouhamed Atman abd rahman anis*

Hani Zerzaihi

Table des matières

Table des matières.....	1
Table des figures	4
Liste des Abbreviation	5
Introduction générale	6
Chapitre I : Généralités sur la classification d’images.....	8
I.1 Introduction	9
I.2 notions de base.....	9
I.2.1 Définition d'une image.....	9
I.2.2 Caractéristiques de l'image	9
I.2.2.1 Pixel	9
I.2.2.2 Dimension & Résolution	10
I.2.2.3 Voisinage	10
I.2.2.4 Niveau de gris.....	11
I.2.2.5 Contraste	11
I.2.2.6 Luminance	11
I.2.2.7 Bruit	12
I.2.2.8 Contour	12
I.2.3 Les Types d'image	12
I.3 Méthodes de classification d’images	13
I.3.1 Définition de la classification	13
I.3.2 Les différentes méthodes de la classification	14
I.3.2.1 Méthodes hiérarchiques	14
I.3.2.2 Méthodes non hiérarchiques	14
I.3.2.3 Méthodes paramétriques	14
I.3.2.4 Méthodes non paramétriques	15
I.3.2.5 Méthodes structurelles	15
I.3.2.6 Méthodes supervisées	15
I.3.2.7 Méthodes non-supervisées.....	16
I.3.3 L'objectif de la classification	17

I.3.4 Domaines d'application de la classification	17
I.3.5 Présentation de certaines techniques de la classification	18
I.3.5.1 k plus proches voisin	18
I.3.5.2 k-means.....	18
I.3.5.3 Fuzzy c-means	19
I.3.5.4 Machine à vecteurs de support.....	19
I.4 conclusion.....	20
Chapitre II : Réseaux de neurones	21
II.1 Introduction	22
II.2 Le neurone formel	22
II.2.1 Perceptron.....	24
II.2.2 Perceptron multi-couches	24
II.3 Le Rétro propagation du gradient (RPG)	25
II.4 La Fonction d'activation	27
II.5 Deep learning (L'apprentissage profond)	26
II.5.1 Définition de l'apprentissage profond (deeplearning).....	28
II.5.2 Fonctionnement du deep Learning	29
II.5.3 Applications du deep Learning.....	29
II.5.4 Quelques algorithmes de Deep Learning	30
II.5.5 Les réseaux neuronaux convolutifs (CNN).....	30
II.5.6 Architecture de réseaux de neurone convolutionnel	30
II.5.6.1 Couche de convolution(CONV)	31
II.5.6.2 Couche de pooling (POOL).....	31
II.5.6.3 Couche de correction (RELU).....	32
II.5.6.4 Couche entièrement connectée (FC)	32
II.5.6.5 Couche de perte (LOSS).....	33
II.5.6.6 Couche Dropout	33
II.5.6.7 Exemples de modèles de CNN	30
II.5.6.8 Choix des paramètres	34
II.6 Conclusion.....	35
Chapitre III : Implémentation et réalisation.....	36
III.1 Introduction.....	37

III. 2 Environnement de développement logiciel	37
III. 2.1 TensorFlow	37
III. 2.2 Keras	37
III. 2.3 Python	38
III. 2.4 Scikit-learn	38
III. 2.5 Anaconda	38
III. 2.6 Pycharm IDE.....	39
III. 2.7 PyQt	39
III. 2.8 OpenCv	39
III. 3 Configuration matériel	40
III.3.1 Configuration matérielle distante (Google Colab)	40
III.3.2 Configuration matérielle local	41
III.3.3 Les base d'images	41
III.3.4 Architecture de notre réseau	41
III.3.5 Résultats obtenus et discussion	46
Conclusion générale.....	51
Bibliographie.....	52
Annexe	54
A Installation de l'Environnement de développement logiciel sous Windows 10 (64 bits)	55
A.1 Python Environment Setup with Anaconda Python.....	55
A.2 Install Opencv, Numpy et Matplotlib.....	56
A.3 Install Keras et Tensorflow	56
A.4 Install CUDA et cuDNN	56

Table des figures

Figure I.1 Voisinage à 4.....	10
Figure I.2 Voisinage à 8.....	11
Figure I.3 couleur RVB.....	12
Figure I.4 Image en niveaux de gris.....	13
Figure I.5 Image binaire.....	13
Figure I.6 L'apprentissage supervise.....	16
Figure I.7 Extrait de la classification taxinomique de Linné.....	17
Figure II.1 Neurone formel.....	23
Figure II.2 Modèle du perceptron.....	24
Figure II.3 Perceptron multi-couches.....	25
Figure II.4 Fonctions d'activations classiques.....	27
Figure II.5 montre exemple de conv2d.....	31
Figure II.6 Exemples de max et moyenne pooling sur une fenêtre 2 x 2 et pas 2.....	32
Fig. II.7 Exemple d'Utilisation Dropout avec une probabilité 0.5 (50%).....	33
Figure II.8 Exemples de modèles de CNN.....	34
figure III.1 Configuration du modèle 1.....	42
figure III.2 Configuration du modèle 2.....	44
figure III.3 Configuration du modèle 3.....	45
Figure III.4 Précision et Erreur pour le Modèle 01.....	46
Figure III.5 Matrice de Confusion pour le Modèle 01.....	47
Figure III.6 Précision et Erreur pour le Modèle 02.....	47
Figure III.7 Matrice de Confusion pour le Modèle 02.....	48
Figure III.8 Précision et Erreur pour le Modèle 03.....	48
Figure III.9 Matrice de Confusion pour le Modèle 03.....	49

Liste des Abbreviation

MNT : Modèle Numérique de Terrain

AI : Artificielle Intelligence

ML : Machine learning

CNN : Convolutional Neural Network

ReLU : Rectified Linear Unit

MLP: Multilayer perceptron

RNN: Recurrent Neural Network

GPU: Graphics processing unit

API: Application programming interface

RVB Rouge, Vert, Bleu

KNN k plus proches voisin

SVM Machine à vecteurs de support

OpenCV Open Source Computer Vision

Introduction générale

Dans la fin des années 80 Yan le Cun a développé un type de réseau particulier qui s'appelle le réseau de neurones convolutionnel, ces réseaux sont une forme particulière de réseau neuronal multicouche dont l'architecture des connexions est inspirée de celle du cortex visuel des mammifères. Par exemple, chaque élément n'est connecté qu'à un petit nombre d'éléments voisins dans la couche précédente. En 1995, Yan le cun et deux autres ingénieurs ont développé un système automatique de lecture de chèques qui a été déployé largement dans le monde. À la fin des années 90, ce système lisait entre 10 et 20 % de tous les chèques émis aux États-Unis. Mais ces méthodes étaient plutôt difficiles à mettre en œuvre avec les ordinateurs de l'époque, et malgré ce succès, les réseaux convolutionnels et les réseaux neuronaux plus généralement ont été délaissés par la communauté de la recherche entre 1997 et 2012.

Aujourd'hui, cette technique est partout. Depuis les assistants virtuels, tels que Home de Google ou Alexa d'Amazon, qui commencent à remplacer efficacement le travail d'assistants et de secrétaires humains ou les chatbots intelligents qui remplacent les opérateurs des services après-vente dans l'assistance des clients. En Chine, les réseaux profonds sont utilisées quotidiennement pour reconnaître les visages de millions de citoyens qui passent devant les caméras de surveillance, permettant ainsi une baisse drastique de la délinquance, même si le procédé soulève des polémiques quant à l'utilisation de l'IA par des régimes répressifs. Dans un contexte plus positif, les prochaines années verront la sortie officielle de la première voiture autonome grâce aux algorithmes de deep learning capables de reconnaître les piétons, les panneaux de signalisation et d'anticiper les trajectoires des objets sur la route.

Plusieurs entreprises proposent des outils d'aide au diagnostic et au suivi du COVID-19 basés sur l'interprétation automatique de radiographies ou de scanners thoraciques par des algorithmes d'intelligence artificielle.

Déjà capable de détecter certaines pathologies sur les radiographies et les scanners des poumons, IA pourrait aider à mieux diagnostiquer et orienter les patients atteints de COVID-19 en automatisant l'interprétation des examens ou en prédisant l'évolution de la maladie. Alors que les laboratoires universitaires du monde entier mobilisent leurs ressources dans cet objectif, certaines entreprises ont déjà dégagé des solutions.

Dans notre projet nous avons utilisé les CNN pour classifier les images. L'un des avantages majeur ces réseaux est l'utilisation d'un poids unique associé aux signaux entrant dans tous les neurones d'un même noyau de convolution. Cette méthode réduit l'empreinte mémoire, améliore les performances et permet une invariance du traitement par translation. Donc notre choix est justifié par la simplicité et l'efficacité de la méthode. Nous avons créé différents modèles avec différentes architectures qui sont appliqués sur le Scanner thoracique du COVID-19.

Pour ce faire, nous avons structuré notre mémoire en trois chapitres :

- Dans le premier chapitre on va présenter les notions de base de la classification des images, les différents types des images et les caractéristiques, ainsi que l'utilisation des réseaux de neurones dans la classification des images.
- Le deuxième chapitre est consacré à la description des réseaux de neurones convolutionnels ainsi que leurs l'intérêt dans le domaine de la classification des images.
- Dans le troisième chapitre, on va montrer la partie expérimentale de notre travail suivie par une discussion des différents résultats obtenus et en fin on termine par une conclusion générale.

Chapitre I

Généralités sur la classification d'images

Chapitre I : Généralités sur la classification d'images

I.1 Introduction

La classification des images consiste à attribuer une classe à une image à l'aide d'un système de classification en se basant sur ce qu'on appelle l'apprentissage. On retrouve ainsi la classification d'objets, de scènes, de textures, la reconnaissance de visages, d'empreintes digitale et de caractères. Il existe deux principaux types d'apprentissage : l'apprentissage supervisé et l'apprentissage non-supervisé. Dans l'approche supervisée, chaque image est associée à une étiquette qui décrit sa classe d'appartenance. Dans l'approche non supervisée les données disponibles ne possèdent pas d'étiquettes. Dans notre travail on s'intéresse de l'approche supervisée.

I.2 notions de base

I.2.1 Définition d'une image

Une image est une représentation planaire d'une scène ou d'un objet situé en général dans un espace tridimensionnel, elle est issue du contact des rayons lumineux provenant des objets formants la scène avec un capteur (caméra, scanner, rayons X, ...). Il ne s'agit en réalité que d'une représentation spatiale de la lumière.

L'image est considérée comme un ensemble de points auquel est affectée une grandeur physique (luminance, couleur). Ces grandeurs peuvent être continues (image analogique) ou bien discrètes (images digitales). Mathématiquement, l'image représente une fonction continue IF, appelée fonction image, de deux variables spatiales représentée par $IF(x, y)$ mesurant la nuance du niveau de gris de l'image aux coordonnées (x, y) . [1]

I.2.2 Caractéristiques de l'image

L'image est un ensemble structuré d'information caractérisé par les paramètres suivants :

I.2.2.1 Pixel

Le pixel est l'abréviation du mot « Picture élément », est une unité de surface permettant de définir la base d'une image numérique. Il matérialise un point donné (x, y) du plan de l'image. L'information présentée par le pixel est le niveau de gris (ou la couleur) prélevée à l'emplacement correspondant dans l'image réelle. La différence entre image monochrome et image couleur réside dans la quantité d'informations contenue dans chaque pixel, par exemple

Chapitre I : Généralités sur la classification d'images

dans une image couleur (RVB : Rouge, Vert, Bleu) la valeur d'un pixel est représentée sur trois octets pour chaque couleur.

I.2.2.2 Dimension & Résolution

La dimension est la taille de l'image. Elle se présente sous forme d'une matrice dont les éléments sont des valeurs numériques représentatives des intensités lumineuses (pixels). Le nombre de lignes de cette matrice multipliée par le nombre de colonnes nous donne le nombre total de pixels dans une image.

Par contre, la résolution est la clarté ou la finesse de détails atteinte par un moniteur ou une imprimante dans la production d'images. Sur les moniteurs d'ordinateur, la résolution est exprimée en nombre de pixels par unité de mesure (pouce ou centimètre). On utilise aussi le mot résolution pour désigner le nombre total de pixels horizontaux et verticaux sur un moniteur. Plus ce nombre est grand, plus la résolution est meilleure

I.2.2.3 Voisinage

Le plan de l'image est divisé en termes de formes rectangulaires ou hexagonales permettant ainsi l'exploitation de la notion de voisinage (voir figure I.1). Le voisinage d'un pixel est formé par l'ensemble des pixels qui se situent autour de ce même pixel. On définit aussi l'assiette comme étant l'ensemble de pixels définissant le voisinage pris en compte autour d'un pixel.

On distingue deux types de voisinage :

1-Voisinage à 4: On ne prend en considération que les pixels qui ont un coté commun avec le pixel considéré.

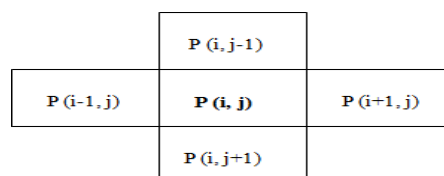


Figure I.1 Voisinage à 4

Chapitre I : Généralités sur la classification d'images

2-Voisinage à 8 : On prend en compte tous les pixels qui ont au moins un point en liaison avec le pixel considéré.

$P(i-1, j-1)$	$P(i, j-1)$	$P(i+1, j-1)$
$P(i-1, j)$	$P(i, j)$	$P(i+1, j)$
$P(i-1, j+1)$	$P(i, j+1)$	$P(i+1, j+1)$

Figure I.2 Voisinage à 8

I.2.2.4 Niveau de gris

C'est la valeur d'intensité lumineuse d'un pixel. Cette valeur peut aller du noir (0) jusqu'au blanc (255) en passant par les nuances qui sont contenues dans l'intervalle $[0, 255]$. Elle correspond en fait à la quantité de la lumière réfléchie.

Pour 8 bits, on dispose de 256 niveaux de gris dont 40 sont reconnus à l'œil nue. Plus le nombre de bit est grand plus les niveaux sont nombreux et plus la représentation est fidèle. [02]

I.2.2.5 Contraste

C'est l'opposition marquée entre deux régions d'une image. Une image contrastée présente une bonne dynamique de la distribution des valeurs de gris sur tout l'intervalle des valeurs possibles, avec des blancs bien clairs et des noirs profonds. Au contraire une image peu contrastée a une faible dynamique, la plupart des pixels ayant des valeurs de gris très proches.

Si $L1$ et $L2$ sont les degrés de luminosité respectivement de deux zones voisines $A1$ et $A2$ d'une image, le contraste est défini par le rapport : $C = \frac{L1-L2}{L1+L2}$

I.2.2.6 Luminance

C'est le degré de luminosité des points de l'image. Elle est définie aussi comme étant le quotient de l'intensité lumineuse d'une surface par l'aire apparente de cette surface, pour un observateur lointain, le mot luminance est substitué au mot brillance, qui correspond à l'éclat d'un objet.

Une bonne luminance se caractérise par :

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

Chapitre I : Généralités sur la classification d'images

- L'absence de parasites .

I.2.2.7 Bruit

Un bruit (parasite) dans une image est considéré comme un phénomène de brusque variation de l'intensité d'un pixel par rapport à ses voisins, il provient de l'éclairage des dispositifs optiques et électroniques du capteur. C'est un parasite qui représente certains défauts (poussière, petits nuages, baisse momentanée de l'intensité électrique sur les capteurs, ...etc.). Il se traduit par des taches de faible dimension et dont la distribution sur l'image est aléatoire. [03]

I.2.2.8 Contour

Les contours représentent la frontière entre les objets de l'image, ou la limite entre deux pixels dont les niveaux de gris représentant une différence significative.

Dans une image numérique, les contours se situent entre les pixels appartenant à des régions ayant des intensités moyennes différentes ; il s'agit de contours de type « saut d'amplitude ». Un contour peut également correspondre à une variation locale d'intensité présentant un maximum ou un minimum ; il s'agit alors de contour « en toit »

I.2.3 Les Types d'image

Il y a 3 types d'image :

Image couleur RVB : L'œil humain analyse la couleur à l'aide de trois types de cellules photo 'les cônes'. Ces cellules sont sensibles aux basses, moyennes, ou hautes fréquences (rouge, vert, bleu). Pour représenter la couleur d'un pixel, il faut donc donner trois nombres, qui correspondent au dosage de trois couleurs de base : Rouge, Vert, Bleu. On peut ainsi représenter une image couleur par trois matrices chacune correspondant à une couleur de base.



Figure I.3 couleur RVB

Chapitre I : Généralités sur la classification d'images

Image d'intensités :C'est une matrice dans laquelle chaque élément est un réel compris entre 0 (noir) et 1 (blanc). On parle aussi d'image en niveaux de gris, car les valeurs comprises entre 0 et 1 représentent les différents niveaux de gris.



Figure I.4 Image en niveaux de gris

Image binaire :Une image binaire est une matrice rectangulaire dans l'élément valent 0 ou 1. Lorsque l'on visualise une telle image, les 0 sont représentés par du noir et les 1 par du blanc.
[01]



Figure I.5 Image binaire

I.3 Méthodes de classification d'images

I.3.1 Définition de la classification

La classification est une tâche qui tente à identifier les objets en se basant sur certaines de leurs caractéristiques. Par exemple, pour pouvoir utiliser les images pour les analyses complémentaires ou pour la cartographie, il est souvent important de traduire l'information de fréquence contenue dans les images en information thématique portant sur l'occupation du sol ou

Chapitre I : Généralités sur la classification d'images

la couverture végétale. On a généralement le choix entre deux approches : la classification supervisée et non supervisée

I.3.2 Les différentes méthodes de la classification

I.3.2.1 Méthodes hiérarchiques

En classification hiérarchique ascendante. Le procédé consiste à grouper les observations individuelles en classes par a partie de la même classe.

Les méthodes se distinguent par le choix de la distance entre les observations et la définition de la stratégie d'agrégation .Dans l'algorithme de base, le calcul de la distance (il s'agit plus exactement d'une quantité critère que l'on appelle distance par abus de langage) fait par récurrence à partir de la matrice des distances entre observations.

I.3.2.2 Méthodes non hiérarchiques

La classification non hiérarchique ou partitionnement, aboutissant à la décomposition de l'ensemble de tous les individus en m ensemble disjoints ou classes d'équivalence, le nombre m de classes est fixé [04]. Le résultat obtenu est alors une partition de l'ensemble des individus, un ensemble de parties, ou classes de l'ensemble I des individus telles que :

- Toute classe soit non vide.
- Deux classes distinctes sont disjointes.
- Tout individu appartient à une classe.

Cet algorithme porte le nom de "agrégation autour de centres variables". Une version égèrement différente, connue sous le nom de "nuées dynamiques" consiste à représenter chaque groupe non pas par son centre, mais par un ensemble de points (noyau) choisis aléatoirement à l'intérieur de chaque groupe. [04]On calcule alors une distance "moyenne" entre chaque observation et ces noyaux et l'on procède à l'affectation.

I.3.2.3 Méthodes paramétriques

Un classifieur est dit paramétrique s'il associe à la signature spectrale (ou profil) une distribution statistique connue, le plus fréquemment pour le traitement d'images, la loi normale ou multi

Chapitre I : Généralités sur la classification d'images

normale. Cette association offre la possibilité d'affecter à chaque pixel une probabilité d'appartenance à une classe donnée

I.3.2.4 Méthodes non paramétriques

Un classifié est dit non paramétrique si aucune distribution statistique paramétrique n'est exploitée, seule la distance spectrale sera alors prise en compte. Cette catégorie comprend notamment les méthodes fondées sur la minimisation de distance (hyper boîte , la distance minimale et la distance de Mahalanobis, K plus proches voisins, K-means, ISODATA, etc.), de nouvelles méthodes apparues récemment s'ajoutent à cette catégorie comme les réseaux neuronaux et les Machines à Support Vecteurs (SVM)

I.3.2.5 Méthodes structurelles

Ce type de méthodes exploite des informations structurelles et contextuelle d'un objet, elles analysent l'objet en termes de ses composantes (primitives) et de leurs propriétés, on trouve par exemple l'analyse syntaxique d'une forme ou un objet à partir d'une grammaire, la distance d'arbres, la distance de graphes (isomorphismes de graphes, de sous-graphes, avec correction d'erreurs, etc.). Dans la méthode structurelle la classe se présente principalement sous la forme de petites régions rondes. [05]

I.3.2.6 Méthodes supervisées

Dans le cas de l'apprentissage supervisé, on dispose d'un ensemble de données étiquetées, ou d'exemples qui se sont vus associés une classe par un professeur ou un expert. Cet ensemble d'exemples constitue la base d'apprentissage .Les méthodes d'apprentissage supervisé se donnent alors comme objectif général déconstruire à partir de la base d'apprentissage, ou fonctions de classement. Une telle fonction permet, à partir de la description d'un objet, de reconnaître un attribut particulier, la classe (Figure I.6) [06]

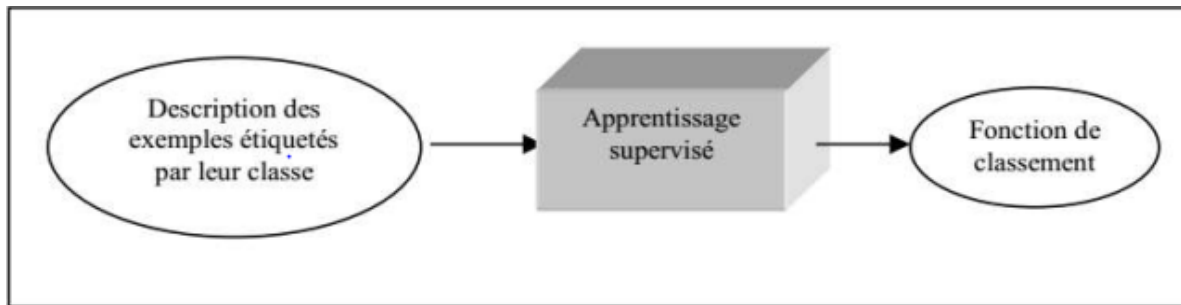


Figure I.6-L'apprentissage supervisé

l'inférence inductive est définie comme un processus qui à partir d'une connaissance spécifique observée sur certains objets et d'une hypothèse inductive initiale ,permet d'obtenir une assertion inductive impliquant ou rendant compte fortement ou faiblement des observations. Dans le cas de l'apprentissage inductif supervisé, qui est un sous domaine de l'inférence inductive, la connaissance spécifique consiste en un ensemble d'objets appartenant à des classes connues. L'assertion inductive est exprimée par une règle de classification qui assigne une classe à chaque objet. L'implication forte est satisfaite si la règle classe correctement tous les objets connus [07].

I.3.2.7 Méthodes non-supervisées

L'apprentissage non-supervisé, encore appelé apprentissage à partir d'observations ou découverte, consiste à déterminer une classification« sensée » à partir d'un ensemble d'objets ou de situations données (des exemples non étiquetés).On dispose d'une masse de données indifférenciées, et l'on désire savoir si elles possèdent une quelconque structure de groupes. Il s'agit d'identifier une éventuelle tendance des données à être regroupées en classes. Ce type d'apprentissage, encore appelé Cluster ING ou Cluster Analysais, se trouve en classification automatique et en taxinomie numérique. Cette forme de classification existe depuis des temps immémoriaux. Elle concerne notamment les sciences de la nature (Figure I.7), les classifications des documents et des livres mais également la classification des sciences élaborées au cours des siècles par les philosophes [08]

Chapitre I : Généralités sur la classification d'images

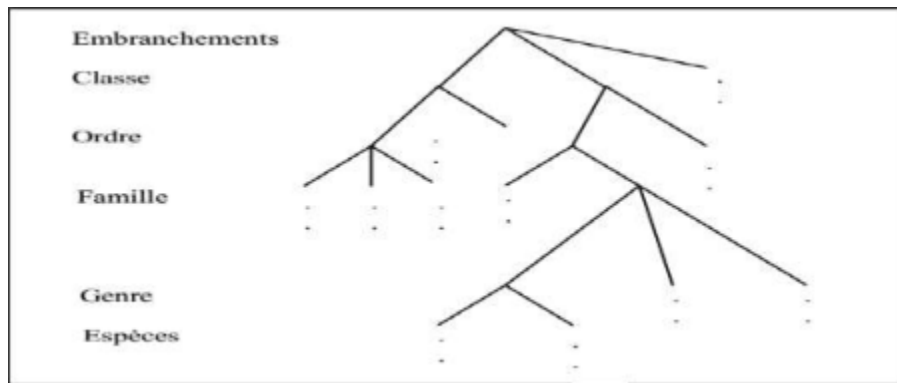


Figure I.7 Extrait de la classification taxinomique de Linné

L'automatisation de la construction de classification constitue aujourd'hui un véritable domaine de recherche. La notion clé utilisée pour créer des classes d'objets est une mesure de la similarité entre les objets. Les classes ou concepts sont construits de façon à maximiser la similarité intra-classes et à minimiser la similarité interclasses.

L'apprentissage non supervisé correspond également à la classification conceptuelle, où une collection d'objets forme une classe si cette classe peut être décrite par un concept, compte tenu d'un ensemble de concepts prédéfinis [09].

I.3.3 L'objectif de la classification

L'objectif de la classification d'images est d'élaborer un système capable d'affecter un classement automatique d'images. Ainsi, ce système permet d'effectuer une tâche d'expertise qui peut s'avérer coûteuse à acquérir pour un être humain en raison notamment de contraintes physiques comme la concentration, la fatigue et le temps nécessaire pour un volume important de données images.

I.3.4 Domaines d'application de la classification

La classification joue un rôle important dans toutes les sciences et techniques qui font appel à la statistique multidimensionnelle. Citons tout d'abord les sciences biologiques : botanique, zoologie, écologie,... Ces sciences utilisent également le terme de "taxinomie" pour désigner l'art de la classification. De même les sciences de la terre et des eaux : géologie, pédologie, géographie, étude des pollutions, font grand usage de classifications.

Chapitre I : Généralités sur la classification d'images

I.3.5 Présentation de certaines techniques de la classification

I.3.5.1 k plus proches voisin

L'algorithme KNN figure parmi les plus simples algorithmes d'apprentissage artificiel. Dans un contexte de classification d'une nouvelle observation x , l'idée fondatrice simple est de faire voter les plus proches voisins de cette observation. La classe de x est déterminée en fonction de la classe majoritaire parmi les k plus proches voisins de l'observation x . Donc la méthode du plus proche voisin est une méthode non paramétrique où une nouvelle observation est classée dans la classe d'appartenance de l'observation de l'échantillon d'apprentissage qui lui est la plus proche, au regard des covariables utilisées. La détermination de leur similarité est basée sur des mesures de distance.

I.3.5.2 k-means

L'algorithme k-means est l'algorithme de regroupement le plus connu et le plus utilisé, du fait de sa simplicité de mise en œuvre. Il partitionne les données d'une image en K clusters. Contrairement à d'autres méthodes dites hiérarchiques, qui créent une structure en « arbre de clusters » pour décrire les groupements, k-means ne crée qu'un seul niveau de clusters. L'algorithme renvoie une partition des données, dans laquelle les objets à l'intérieur de chaque cluster sont aussi proches que possible les uns des autres et aussi loin que possible des objets des autres clusters. Chaque cluster de la partition est défini par ses objets et son centroïde. Le k-means est un algorithme itératif qui minimise la somme des distances entre chaque objet et le centroïde de son cluster. La position initiale des centroïdes conditionne le résultat final, de sorte que les centroïdes doivent être initialement placés le plus loin possible les uns des autres de façon à optimiser l'algorithme. K-means change les objets de cluster jusqu'à ce que la somme ne puisse plus diminuer. Le résultat est un ensemble de clusters compacts et clairement séparés, sous réserve qu'on ait choisi la bonne valeur K du nombre de clusters. Les principales étapes de l'algorithme k-means sont :

1. Choix aléatoire de la position initiale des K clusters.
2. (Ré-Affecter les objets à un cluster suivant un critère de minimisation des distances (généralement selon une mesure de distance euclidienne).

Chapitre I : Généralités sur la classification d'images

3. Une fois tous les objets placés, recalculer les K centroïdes.
4. Réitérer les étapes 2 et 3 jusqu'à ce que plus aucune réaffectation ne soit faite.

I.3.5.3 Fuzzy c-means

Fuzzy C-Means(FCM) est un algorithme de classification non-supervisée floue. Issu de l'algorithme des C-moyennes (C-means), il introduit la notion d'ensemble flou dans la définition des classes : chaque point dans l'ensemble des données appartient à chaque cluster avec un certain degré, et tous les clusters sont caractérisés par leur centre de gravité. Comme les autres algorithmes de classification non supervisée, il utilise un critère de minimisation des distances intra-classe et de maximisation des distances interclasse, mais en donnant un certain degré d'appartenance à chaque classe pour chaque pixel. Cet algorithme nécessite la connaissance préalable du nombre de clusters et génère les classes par un processus itératif en minimisant une fonction objective. Ainsi, il permet d'obtenir une partition floue de l'image en donnant à chaque pixel un degré d'appartenance (compris entre 0 et 1) à une classe donnée. Le cluster auquel est associé un pixel est celui dont le degré d'appartenance sera le plus élevé. Les principales étapes de l'algorithme Fuzzy C-means sont :

1. La fixation arbitraire d'une matrice d'appartenance.
2. Le calcul des centroïdes des classes.
3. Le réajustement de la matrice d'appartenance suivant la position des centroïdes.
4. Calcul du critère de minimisation et retour à l'étape 2 s'il y a non convergence de critère.

I.3.5.4 Machine à vecteurs de support

Les machines à vecteurs de support ou séparateurs à vaste marge (en anglais Support Vector Machine, SVM) sont un ensemble de techniques d'apprentissage supervisé destinées à résoudre des problèmes de classification, en cherchant à séparer les classes d'une façon optimale. Les SVM ont été développés dans les années 1990 à partir des considérations théoriques de Vladimir Vapnik sur le développement d'une théorie statistique de l'apprentissage : la Théorie de Vapnik Chervonenkis. Les SVM ont rapidement été adoptés pour leur capacité à travailler avec des données de grandes dimensions, le faible nombre d'hyper paramètres, leurs garanties théoriques, et leurs bons résultats en pratique. Les SVM ont été appliqués à de très nombreux domaines (bioinformatique, recherche d'information, vision par ordinateur, finance...). Selon les données,

Chapitre I : Généralités sur la classification d'images

la performance des machines à vecteurs de support est de même ordre, ou même supérieure, à celle d'un réseau de neurones ou d'un modèle de mixture gaussienne.

L'hyperplan optimal qui sépare les deux classes est le lieu des points x satisfaisant $\langle w, x \rangle + b = 0$ où w et b sont des paramètres à déterminer. On voit que le vecteur w définit la pente de l'hyperplan (w est perpendiculaire à l'hyperplan), le terme b quant à lui permet de translater l'hyperplan parallèlement à lui-même. L'obtention de cet hyperplan est basée sur la recherche de deux autres hyperplans canoniques passant par les vecteurs supports. La règle de décision $h(x)$ consiste à déterminer l'appartenance d'une forme x à une classe en observant de quel côté de l'hyperplan se trouve x .

I.4 conclusion

Nous avons consacré ce chapitre à la présentation des notions des bases d'images et leur classification et on a présenté aussi certaines techniques de la classification. Dans le deuxième chapitre on va bien détailler les réseaux de neurones et plus précisément l'utilisation des réseaux de neurones convolutionnels dans la classification des images.

Chapitre II

Les réseaux de neurones

Chapitre II : Les réseaux de neurones

II.1 Introduction

Les réseaux de neurones sont des structures cellulaires artificielles et constituent une approche permettant d'aborder sous des angles nouveaux les problèmes de perception, de mémoire, d'apprentissage et de raisonnement (en d'autres termes... d'intelligence artificielle ou abrégée "I.A.") au même titre que les algorithmes génétiques. Ils s'avèrent aussi des alternatives très prometteuses pour contourner certaines des limitations des ordinateurs classiques. Grâce à leur traitement parallèle de l'information et à leurs mécanismes inspirés des cellules nerveuses (neurones), ils infèrent des Propriétés émergentes permettant de solutionner des problèmes jadis qualifiés de complexes. [10]

De nombreux termes sont utilisés pour désigner le domaine des réseaux de neurones artificiels, comme connexionnisme ou neuromimétique. Connexionnisme et neuromimétique sont tous deux des domaines de recherche à part entière, qui manipulent chacun des modèles de réseaux de neurones artificiels, mais avec des objectifs différents [11]

II.2 Le neurone formel

Un neurone formel est un modèle mathématique. Il est utilisé dans le cadre des recherches sur l'intelligence artificielle, principalement en association avec d'autres neurones formels pour former un réseau de neurones. [12]

Il contient plusieurs formules mathématiques afin de reproduire le plus fidèlement possible le fonctionnement d'un neurone avec ses différentes entrées (dendrites), leurs importances (coefficient de pondération) et sa sortie (axone) et ainsi comprendre son potentiel d'interaction avec les autres neurones.

Il est caractérisé par :

- Des entrées e_i et leur poids synaptique w_i
- Une fonction d'activation (de transfert) f
- Une sortie y

Chapitre II : Les réseaux de neurones

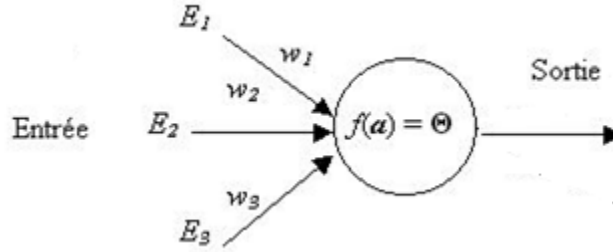


Figure II.1 : Neurone formel

- Les entrées a_j du neurone i proviennent soit d'autres neurones soit de l'environnement.
- Les poids W_{ji} associés à chaque neurone j déterminent l'influence de chaque entrée i .
- Le seuil S permet de contrôler l'entrée de la fonction d'activation f du neurone.
- Le potentiel d'activation est exprimé sous forme d'une combinaison linéaire des entrées a_j pondérées par les coefficients W_{ji} .
- La fonction d'activation f reçoit en entrée la somme pondérée pour produire la valeur de sortie du neurone qui sera transmise aux neurones suivants. Ainsi, à chaque signal entrant est associé un poids car les synapses n'ont pas toutes la même valeur. La valeur de la sommation est comparée à un seuil et la sortie du neurone est une fonction non linéaire du résultat:

$$in_i = \sum_{j=1}^n w_{ji} a_j - \theta \quad (1)$$

$$a_i = f(in_i) \quad (2)$$

Plusieurs fonctions d'activation f peuvent être utilisées. Toutefois dans la pratique, les deux fonctions de transfert les plus utilisées sont la fonction de Heaviside et la fonction sigmoïde.

- fonction de Heaviside: $\forall x \in R, f(x) = \begin{cases} 1 & \text{si } x \geq 0 \\ 0 & \text{sinon} \end{cases} \quad (3)$

- tangente hyperbolique : $\forall x \in R, f(x) = \frac{2}{1+e^{-2x}} - 1 \quad (4)$

- fonction gaussienne : $\forall x \in R, f(x) = e^{-\frac{x^2}{2}} \quad (5)$

- fonction sigmoïde : $\forall x \in R, f(x) = \frac{1}{1+e^{-x}} \quad [13] \quad (6)$

Chapitre II : Les réseaux de neurones

II.2.1 Perceptron

Le perceptron a été introduit en 1958 par Franck Rosenblatt. Il s'agit d'un neurone artificiel inspiré par la théorie cognitive de Friedrich Hayek et celle de Donald Hebb. Dans sa version la plus simple, le perceptron n'a qu'une seule sortie y à laquelle toutes les entrées x_i sont connectées (voir Figure II.2), ses entrées et sorties étant booléennes. [14]

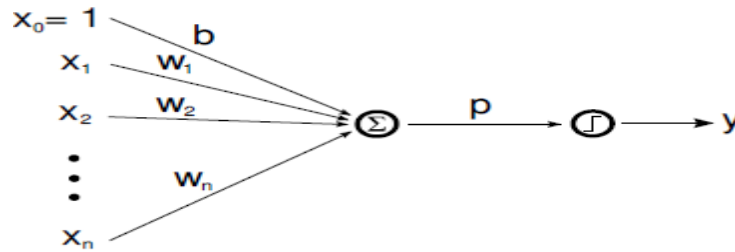


Figure II.2 : Modèle du perceptron

La somme pondérée des entrées par les poids W_i associés aux entrées est appelée potentiel, noté p .

$$p = \sum W_i X_i \quad (7)$$

Ce potentiel est alors soumis à une fonction seuil de type Heaviside :

$$y = \begin{cases} 0 & \text{si } s < 0 \\ 1 & \text{si } s \geq 0 \end{cases} \quad (8)$$

Cependant Minsky et Papert relèvent que le perceptron échoue pour des problèmes de classification simples, comme ceux où les classes ne sont pas linéairement séparables. Le cas est la classification du XOR, où les motifs (0 ; 0) Et (1 ; 1) appartiennent à une classe tandis que les motifs (1 ; 0) et (0 ; 1) appartiennent à une autre classe. Ces problèmes suggèrent l'utilisation de plusieurs perceptrons, qui organisés en couches forment le modèle du perceptron multicouches. Ce modèle est capable de traiter des problèmes non linéaires.

II.2.2 Perceptron multi-couches

Dans le modèle du Perceptron Multicouches, les perceptrons sont organisés en couches. Les perceptrons multicouches sont capables de traiter des données qui ne sont pas linéairement séparables. Avec l'arrivée des algorithmes de rétro propagation, ils deviennent le type de réseaux

Chapitre II : Les réseaux de neurones

de neurones le plus utilisé. Les MLP sont généralement organisés en trois couches, la couche d'entrée, la couche intermédiaire (dite couche cachée) et la couche de sortie. L'utilité de plusieurs couches cachées n'a pas été démontrée. (La figure II.3) illustre la structure d'un MLP présentant quatre neurones en entrée, trois neurones sur la couche cachée et deux en sortie.

Lorsque tous les neurones d'une couche sont connectés aux neurones de la couche suivante, on parle alors de couches complètement connectées.[14]

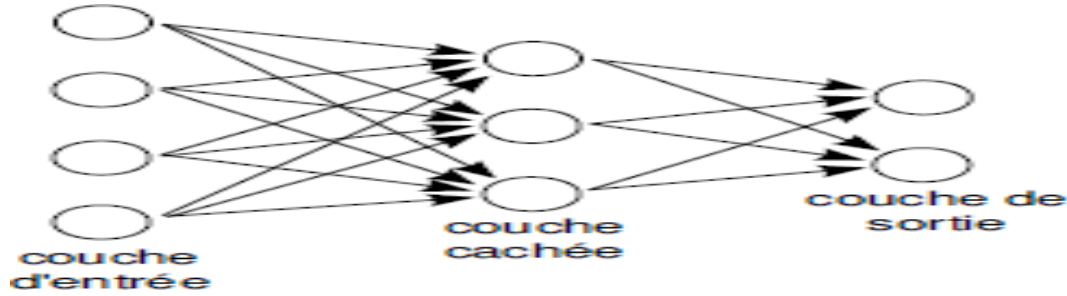


Figure II.3 Perceptron multi-couches

Les équations de Propagation décrites plus haut s'appliquent à tous les neurones. Cependant, le passage d'une couche à l'autre peut être formalisé sous forme matricielle. Soit un MLP dont le nombre de neurones sur la couche d'entrée est n_0 , n_1 sur la couche cachée et n_2 sur la couche de sortie. Les poids de la couche cachée peuvent s'écrire sous la forme d'une matrice $W^{n_1 \times n_0}$. Pour un vecteur X^{n_0} présenté en entrée, le vecteur potentiel V et le vecteur de sortie Y pour la couche cachée s'écrivent donc.

$$V = WX \quad (9)$$

$$Y = \Phi(V) \quad (10)$$

Avec Φ une fonction d'activation. La sortie de la couche cachée devient ensuite l'entrée de la dernière couche.

II.3 Rétro propagation du gradient (RPG)

La rétro propagation du gradient s'applique de manière récursive de la couche de sortie à la couche d'entrée tout en minimisant l'erreur quadratique $\frac{1}{2}(Y - D)^2$. Elle consiste à faire évoluer les poids W dans la direction inverse du gradient $w = w - \lambda dw$ où λ est le taux d'apprentissage.

Chapitre II : Les réseaux de neurones

Pour une couche dont X est le vecteur d'entrée, V le potentiel, W la matrice de poids et Y la sortie, la rétro propagation s'écrit donc :

$$\frac{\partial L}{\partial V} = \frac{\partial L}{\partial Y} \frac{\partial Y}{\partial V} \quad (11)$$
$$= \Phi'(V) \frac{\partial L}{\partial Y}$$

$$\frac{\partial L}{\partial X} = \frac{\partial L}{\partial V} \frac{\partial V}{\partial X} \quad (12)$$
$$= W^T \frac{\partial L}{\partial V}$$

II.4 La Fonction d'activation

La fonction d'activation est la fonction mathématique qui permet de traiter l'information qui arrive à un neurone artificiel en machine Learning, comme le fait ceux du cerveau avec les signaux électriques qu'ils reçoivent.

Les principales fonctions d'activation Φ sont : la fonction linéaire, la fonction sigmoïde et la fonction tangente hyperbolique :

Linéaire $\Phi(p) = p$ (13)

Sigmoïde $\Phi(p) = \frac{1}{1+e^{-cp}}$ (14) $(c > 0)$

Tangente hyperbolique $\Phi(p) = \frac{1-e^{-cp}}{1+e^{-cp}}$ (15)

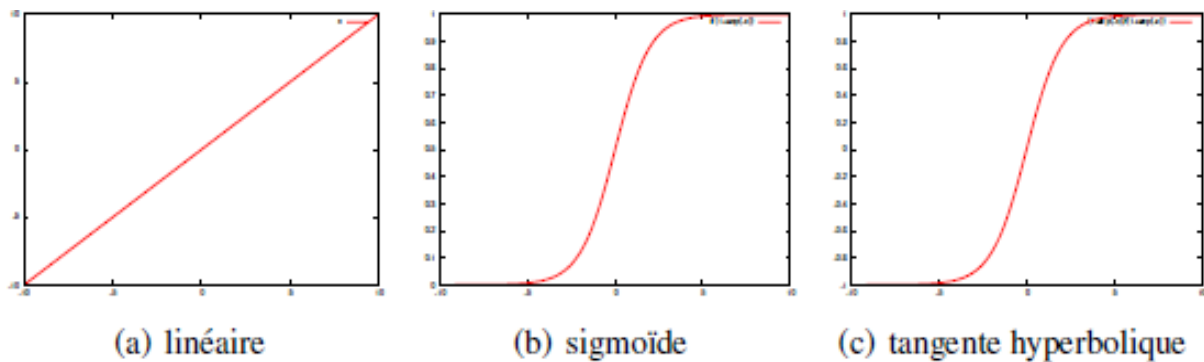


Figure II.4 Fonctions d'activations classiques

Chapitre II : Les réseaux de neurones

La figure II.4 montre les trois fonctions d'activation classiques. Il faut remarquer que la fonction linéaire est dans $]-\infty ; +\infty [$, la fonction sigmoïde dans $]0 ; 1[$ et la fonction tangente hyperbolique dans $] -1 ; 1[$.

La propagation d'un vecteur d'entrée (x_i) à travers un perceptron s'écrit donc :

$$p = \sum \omega_i x_i \quad (16)$$

$$y = \Phi(p) \quad (17)$$

L'apprentissage classique d'un perceptron est la régression où la fonction de Coût est de la forme:

$$L = \frac{1}{2} (o - d)^2 \quad (18)$$

Où o est la sortie obtenue pour un échantillon et d est la sortie désirée u (label)

Le gradient de l'erreur pour le potentiel est :

$$\begin{aligned} \frac{\partial L}{\partial p} &= \frac{\partial L}{\partial y} \frac{\partial y}{\partial p} \\ &= \frac{\partial L}{\partial y} \Phi'(p) \end{aligned} \quad (19)$$

Et le gradient de l'erreur pour l'entrée x_i est :

$$\begin{aligned} \frac{\partial L}{\partial x_i} &= \frac{\partial L}{\partial p} \frac{\partial p}{\partial x_i} \\ &= \frac{\partial L}{\partial p} w_i \end{aligned} \quad (20)$$

Une fois la rétro propagation du gradient effectuée pour toutes les couches, les matrices de poids sont mises à jour .

L'algorithme de la rétropropagation

1. Initialiser les poids W , avec des valeurs aléatoires entre $[-1, 1]$.
2. Mélanger les exemples .
3. Pour tout exemple x faire :

Chapitre II : Les réseaux de neurones

- a. Calculer les sorties de réseaux en propageant les entrées vers les sorties.
- b. Corriger les poids en rétropropageant l'erreur :

$$w_{(x)ij} = w_{(x-1)ij} + \alpha \delta_{(x)j} o_{(x)i} \quad (22)$$

Avec :

$$\delta_{(x)j} = \begin{cases} e_{(x)j} o_{(x)j} [1 - o_{(x)j}] & \text{si } j \in \text{la couche de sortie} \\ o_{(x)j} [1 - o_{(x)j}] \left[\sum_{k \in c} \delta_{(x)k} W_{(x)jk} \right] & \text{si } j \in \text{la couche de cachée} \end{cases} \quad (23)$$

Ou :

x : l'exemple courant.

$x - 1$: l'exemple précédent.

i : neurone i de la couche précédent.

j : neurone j de la couche suivante.

α : représente le taux ou le pas d'apprentissage.

4. Répéter les étapes 2 et 3 jusqu'à remplir le critère d'arrêt.

II.5 Deep learning (L'apprentissage profond)

II.5.1 Définition de l'apprentissage profond (deep learning)

L'apprentissage profond (deep learning) est une classe de méthodes dont les principes sont connus depuis la fin des années 80, mais dont l'utilisation ne s'est vraiment généralisée que depuis environ 2012 [15]. L'apprentissage profond peut être vu comme un réseau multicouche avec de nombreuses couches intermédiaires [16][15]. L'architecture profonde est capable d'extraire automatiquement des caractéristiques de l'image, par exemple les premières couches extrairont des contours que les couches suivantes forment en des concepts de plus en plus complexes et abstraits : des formes, des objets, de parties d'objets en objets, etc [15].

Le deep learning ou apprentissage profond est un type d'intelligence artificielle dérivé de machine learning (apprentissage automatique) où la machine est capable d'apprendre par elle-

Chapitre II : Les réseaux de neurones

même, contrairement à la programmation où elle se contente d'exécuter à la lettre des règles prédéterminées

II.5.2 Fonctionnement du deep Learning

Le deep Learning s'appuie sur un réseau de neurones artificiels s'inspirant du cerveau humain. Ce réseau est composé de dizaines voire de centaines de « couches » de neurones, chacune recevant et interprétant les informations de la couche précédente. Le système apprendra par exemple à reconnaître les lettres avant de s'attaquer aux mots dans un texte, ou détermine s'il y a un visage sur une photo avant de découvrir de quelle personne il s'agit à chaque étape, les « mauvaises » réponses sont éliminées et renvoyées vers les niveaux en amont pour ajuster le modèle mathématique. Au fur et à mesure, le programme réorganise les informations en blocs plus complexes. Lorsque ce modèle est par la suite appliqué à d'autres cas, il est normalement capable de reconnaître un chat sans que personne ne lui ait jamais indiqué qu'il n'ai jamais appris le concept de chat. Les données de départ sont essentielles : plus le système accumule d'expériences différentes, plus il sera performant.

II.5.3 Applications du deep Learning

Le deep Learning est utilisé dans de nombreux domaines :

- reconnaissance d'image.
- traduction automatique .
- voiture autonome.
- diagnostic médical.
- recommandations personnalisées.
- modération automatique des réseaux sociaux.
- prédiction financière et trading automatisé.
- identification de pièces défectueuses.
- détection de malwares ou de fraudes .
- chatbots (agents conversationnels).
- exploration spatiale.
- robots intelligents.

II.5.4 Quelques algorithmes de Deep Learning

Il existe différents algorithmes d'apprentissage profond qui utilisent les méthodes:

- **Les réseaux de neurones profonds :** (Deep Neural Networks). Ces réseaux sont similaires aux réseaux MLP mais avec plus de couches cachées. L'augmentation du nombre de couches, permet à un réseau de neurones de détecter de légères variations du modèle d'apprentissage, favorisant le sur-apprentissage ou sur-ajustement (« overfitting »).
- **Les réseaux de neurones convolutionnels :** (CNN ou Convolutional Neural Networks). Le problème est divisé en sous parties, et pour chaque partie, un «cluster» de neurones sera créé afin d'étudier cette portion spécifique. Par exemple, pour une image en couleur, il est possible de diviser l'image sur la largeur, la hauteur et la profondeur (les couleurs).
- **La machine de Boltzmann profonde :** (Deep Belief Network): Ces algorithmes fonctionnent suivant une première phase non supervisée, suivie de l'entraînement classique supervisé. Cette étape d'apprentissage non-supervisée, permet, en outre, de faciliter l'apprentissage supervisé.

II.5.5 Les réseaux neuronaux convolutifs (CNN)

Les réseaux de neurones convolutifs (CNN) sont des variantes des réseaux profonds et permettent d'extraire les caractéristiques d'une image [17]. Leur fonctionnement est inspiré par les processus biologiques. Les réseaux neuronaux convolutifs ont de larges applications dans la reconnaissance d'image et vidéo, les systèmes de recommandation.

II.5.5.1 Architecture de réseaux de neurone convolutionnel

Une architecture CNN est formée par un empilement de couches de traitement indépendantes :

- La couche de convolution (CONV) qui traite les données d'un champ récepteur
- La couche de pooling (POOL), qui permet de compresser l'information en réduisant la taille de l'image intermédiaire (souvent par sous-échantillonnage)

Chapitre II : Les réseaux de neurones

- La couche de correction (ReLU), souvent appelée par abus 'ReLU' en référence à la fonction d'activation (Unité de rectification linéaire)
- La couche "entièrement connectée" (FC), qui est une couche de type perceptron.
- La couche de perte (LOSS)

II.5.5.1.1 Couche de convolution(CONV)

La couche de convolution est le bloc de construction de base du réseau convolutif. Ces couches se composent d'une grille rectangulaire de neurones qui ont un petit champ réceptif, mais s'étend à travers toute la profondeur du volume d'entrée. Ainsi, la couche de convolution est juste une convolution d'image de la couche précédente, où les poids spécifient le filtre de convolution [15]. La Figure II.5 montre un exemple de conv2d.

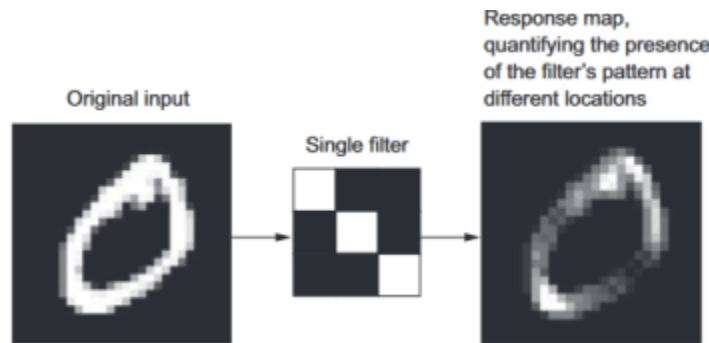


Figure II.5 montre exemple de conv2d

Les trois hyperparamètres permettent de dimensionner le volume de la couche de convolution :

- Profondeur : nombre de noyaux de convolution.
- Le pas : contrôle le chevauchement des champs récepteurs.
- Zéro padding : mettre des zéros à la frontière du volume d'entrée.

II.5.5.1.2 Couche de pooling (POOL)

Après chaque couche de convolution, il peut y avoir une couche de pooling. La couche de pooling est une forme de sous-échantillonnage de l'image entrée. (La Figure II.6) montre la mise en commun couramment utilisés, à savoir, pooling moyen et max pooling [18] [19]. Le pooling réduit la taille spatiale d'une image intermédiaire, réduisant ainsi la quantité de paramètres et de calcul dans le réseau.

Chapitre II : Les réseaux de neurones

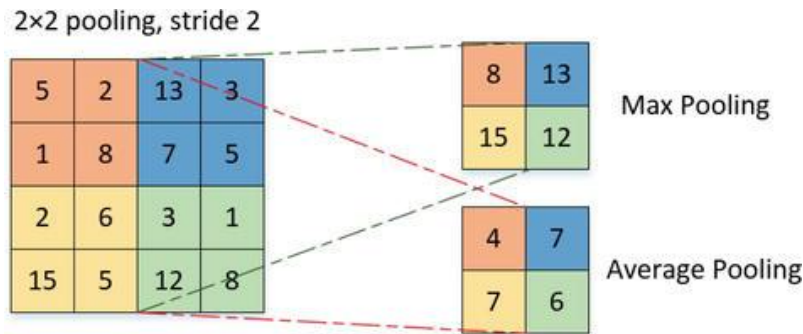


Figure II.6 Exemples de max et moyenne pooling sur une fenêtre 2 x 2 et pas 2

II.5.5.1.3 Couche de correction (RELU)

Souvent, il est possible d'améliorer l'efficacité du traitement en intercalant entre les couches de traitement une couche qui va opérer une fonction mathématique (fonction d'activation) sur les signaux de sortie. On a notamment :

- La correction ReLU (abréviation de Unités Rectifié linéaires): $f(x) = \max(0, x)$. Cette fonction, appelée aussi « fonction d'activation non saturante », augmente les propriétés non linéaires de la fonction de décision et de l'ensemble du réseau sans affecter les champs récepteurs de la couche de convolution.
- La correction par tangente hyperbolique $f(x) = \tanh(x)$.
- La correction par la tangente hyperbolique saturante : $f(x) = |\tanh(x)|$.
- La correction par la fonction sigmoïde.

Souvent, la correction Relu est préférable, car il en résulte la formation de réseau neuronal plusieurs fois plus rapide, sans faire une différence significative à la généralisation de précision.

II.5.5.1.4 Couche entièrement connectée (FC)

Après plusieurs couches de convolution et de max dans le réseau neuronal se fait via des couches entièrement connectées. Les neurones dans une couche entièrement connectée ont des connexions vers toutes les sorties de la couche précédente (comme on le voit régulièrement dans les réseaux réguliers de neurones). Leurs fonctions d'activations peuvent donc être calculées avec une multiplication matricielle suivie d'un décalage de polarisation.

Chapitre II : Les réseaux de neurones

II.5.5.1.5 Couche de perte (LOSS)

La couche de perte spécifie comment l'entraînement du réseau pénalise l'écart entre le signal prévu et réel. Elle est normalement la dernière couche dans le réseau. Diverses fonctions de perte adaptées à différentes tâches peuvent y être utilisées. La perte « Soft max » est utilisée pour prédire une seule classe parmi K classes mutuellement exclusives. La perte par entropie croisée sigmoïde est utilisée pour prédire K valeurs de probabilité indépendante dans $[0,1]$. La perte euclidienne est utilisée pour régresser vers des valeurs réelles.

II.5.5.1.6 Couche Dropout

Dropout est l'une des techniques de régularisation des réseaux de neurones les plus efficaces et les plus couramment utilisées [20], développé par Srivastava et al. [21]. Dropout, appliqué à une couche, consiste à abandonner de manière aléatoire (mettre à zéro) un certain nombre (précisé par une probabilité donnée à Dropout) d'entités en sortie de la couche pendant l'entraînement [20] (voir figure II.7).

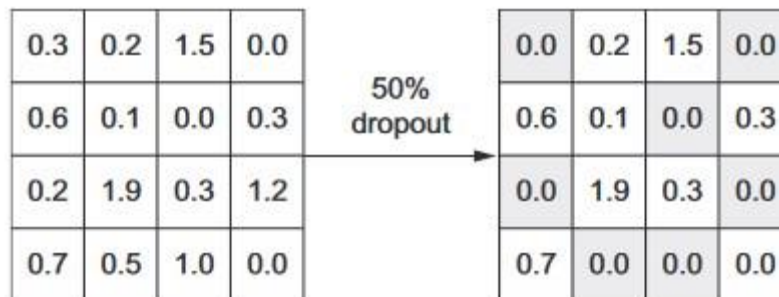


Fig. II.7 Exemple d'Utilisation Dropout avec une probabilité 0.5 (50%)[20]

II.5.5.1.7 Exemples de modèles de CNN

La forme la plus commune d'une architecture de réseau de neurones convolutifs empile quelques couches Conv-ReLU, les suit avec des couches Pool, et répète ce schéma jusqu'à ce que l'entrée soit réduite dans un espace d'une taille suffisamment petite. À un moment, il est fréquent de placer des couches entièrement connectées (FC). La dernière couche entièrement connectée est reliée vers la sortie. Voici quelques architectures communes de réseau de neurones convolutifs qui suivent ce modèle :

Chapitre II : Les réseaux de neurones

- INPUT -> CONV -> RELU -> FC
- INPUT -> [CONV -> RELU -> POOL] * 2 -> FC -> RELU -> FC Ici, il y a une couche de CONV unique entre chaque couche POOL
- INPUT -> [CONV -> RELU -> CONV -> RELU -> POOL] * 3 -> [FC -> RELU] * 2 -> FC Ici, il y a deux couches CONV empilées avant chaque couche POOL.

L'empilage des couches CONV avec de petits filtres de pooling (plutôt un grand filtre) permet un traitement plus puissant, avec moins de paramètres.

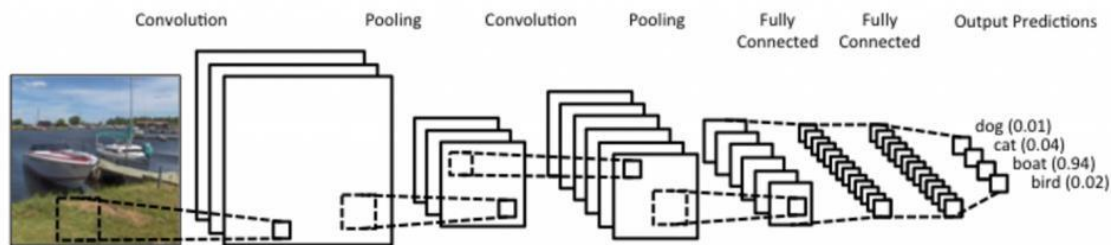


Figure II.8 : Exemples de modèles de CNN

II.5.5.1.8 Choix des paramètres

Les réseaux de neurones convolutifs utilisent plus d'hyper paramètres qu'un perceptron multicouche standard. Même si les règles habituelles pour les taux d'apprentissage et des constantes de régularisation s'appliquent toujours, il faut prendre en considération les notions de nombre de filtres, leur forme et la forme du max pooling. [22]

Nombre de filtres : Comme la taille des images intermédiaires diminue avec la profondeur du traitement, les couches proches de l'entrée ont tendance à avoir moins de filtres tandis que les couches plus proches de la sortie peuvent en avoir davantage. Pour égaliser le calcul à chaque couche, le produit du nombre de caractéristiques et le nombre de pixels traités est généralement choisi pour être à peu près constant à travers les couches. Pour préserver l'information en entrée, il faudrait maintenir le nombre de sorties intermédiaires (nombre d'images intermédiaires multiplié par le nombre de positions de pixel) pour être croissante (au sens large) d'une couche à l'autre. Le nombre d'images intermédiaires contrôle directement la puissance du système, dépend du nombre d'exemples disponibles et la complexité du traitement

Chapitre II : Les réseaux de neurones

Forme du filtre : Les formes de filtre varient grandement dans la littérature. Ils sont généralement choisis en fonction de l'ensemble de données. Les meilleurs résultats sur les images de MNIST (28x28) sont habituellement dans la gamme de 5x5 sur la première couche, tandis que les ensembles de données d'images naturelles (souvent avec des centaines de pixels dans chaque dimension) ont tendance à utiliser de plus grands filtres de première couche de 12x12, voire 15x15. Le défi est donc de trouver le bon niveau de granularité de manière à créer des abstractions à l'échelle appropriée et adaptée à chaque cas

Forme du Max Pooling : Les valeurs typiques sont 2x2. De très grands volumes d'entrée peuvent justifier un pooling 4x4 dans les premières couches. Cependant, le choix de formes plus grandes va considérablement réduire la dimension du signal, et peut entraîner la perte de trop d'information.

II.6 Conclusion

Dans ce chapitre on a présenté les notions importantes qui sont en relation avec l'apprentissage profond (définition, Architectures...etc.). Aussi qu'une vision générale sur l'apprentissage profond, tout en donnant en détail la méthode choisie dans notre travail de recherche qui est le CNNs. Le prochain chapitre, traite les détails de la conception, ainsi que la méthode et les outils utilisés pour la réalisation de notre application.

Chapitre III

Implémentation et réalisation

Chapitre III: Implémentation et réalisation

III.1 Introduction

Dans ce chapitre, nous allons présenter les expérimentations effectuées sur notre modèle afin de démontrer son efficacité à détecter covid 19 à partir de Scanner thoracique. Nous détaillerons d'abord les environnements matériels et logiciels que nous avons utilisés pour développer notre application, puis nous comparerons les résultats.

III. 2 Environnement de développement logiciel

III. 2.1 TensorFlow

TensorFlow¹ est un framework de programmation pour le calcul numérique qui a été rendu Open Source par Google en Novembre 2015. Depuis son release, TensorFlow n'a cessé de gagner en popularité, pour devenir très rapidement l'un des frameworks les plus utilisés pour le Deep Learning et donc les réseaux de neurones. Son nom est notamment inspiré du fait que les opérations courantes sur des réseaux de neurones sont principalement faites via des tables de données multi-dimensionnelles, appelées Tenseurs (Tensor). Un Tensor à deux dimensions est l'équivalent d'une matrice. Aujourd'hui, les principaux produits de Google sont basés sur TensorFlow: Gmail, Google Photos, Reconnaissance de voix.

III. 2.2 Keras

Keras² est une API de réseaux de neurones de haut niveau, écrite en Python et capable de fonctionner sur TensorFlow ou Theano. Il a été développé en mettant l'accent sur l'expérimentation rapide. Être capable d'aller de l'idée à un résultat avec le moins de délai possible est la clé pour faire de bonnes recherches. Il a été développé dans le cadre de l'effort de recherche du projet ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), et son principal auteur et mainteneur est François Chollet, un ingénieur Google.

En 2017, l'équipe TensorFlow de Google a décidé de soutenir Keras dans la bibliothèque principale de TensorFlow. Chollet a expliqué que Keras a été conçue comme une interface plutôt

-
1. <https://www.tensorflow.org/>
 2. <https://keras.io/>

Chapitre III: Implémentation et réalisation

que comme un cadre d'apprentissage end to end. Il présente un ensemble d'abstractions de niveau supérieur et plus intuitif qui facilitent la configuration des réseaux neuronaux indépendamment de la bibliothèque informatique de backend. Microsoft travaille également à ajouter un backend CNTK à Keras aussi.

III. 2.3 Python

Python³ est un langage de programmation de haut niveau interprété (il n'y a pas d'étape de compilation) et orienté objet avec une sémantique dynamique. Il est très sollicité par une large communauté de développeurs et de programmeurs. Python est un langage simple, facile à apprendre et permet une bonne réduction du cout de la maintenance des codes. Les bibliothèques (packages) python encouragent la modularité et la réutilisabilité des codes.

III. 2.4 Scikit-learn

Scikit-learn est une bibliothèque libre Python dédiée à l'apprentissage automatique. Elle est développée par de nombreux contributeurs notamment dans le monde académique par des instituts français d'enseignement supérieur et de recherche comme Inria et Télécom ParisTech. Elle comprend notamment des fonctions pour estimer des forêts aléatoires, des régressions logistiques, des algorithmes de classification, et les machines à vecteurs de support. Elle est conçue pour s'harmoniser avec des autres bibliothèques libre Python, notamment NumPy et SciPy.

III. 2.5 Anaconda

Anaconda⁴ est une distribution libre et open source des langages de programmation Python et R appliquée au développement d'applications dédiées à la fouille de données et à l'apprentissage automatique (traitement de données à grande échelle, analyse prédictive, calcul scientifique), qui vise à simplifier la gestion des paquets et de déploiement. Les versions dépaquetages sont gérées par le système de gestion de paquets conda. La distribution Anaconda est utilisée par plus de 6 millions d'utilisateurs et comprend plus de 250 paquets populaires fouille de des données adaptés pour Windows, Linux et MacOS.

3. <https://www.python.org/>

4. <https://www.anaconda.com/>

Chapitre III: Implémentation et réalisation

III. 2.6 Pycharm IDE

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-plate forme 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.

III. 2.7 PyQt

PyQt⁵ est l'une des liaisons Python les plus populaires pour le framework C++ multiplateforme Qt. PyQt a été développé par Riverbank Computing Limited. Qt lui-même est développé dans le cadre du projet Qt. PyQt fournit des liaisons pour Qt 4 et Qt 5. PyQt est distribué sous un choix de licences: GPL version 3 ou une licence commerciale.

PyQt est disponible en deux éditions: PyQt4 qui compilera contre Qt 4.x et 5.x et PyQt5 qui ne compilera que contre 5.x. Les deux éditions peuvent être construites pour Python 2 et 3. PyQt contient plus de 620 classes qui couvrent les interfaces utilisateur graphiques, la gestion XML, la communication réseau, les bases de données SQL, la navigation Web et d'autres technologies disponibles dans Qt.

III. 2.8 OpenCV

OpenCV⁶ (Open source Computer Vision) est une bibliothèque libre de vision par ordinateur, qui possède plus de 2500 algorithmes optimisés, La bibliothèque comprend un ensemble complet d'algorithmes de vision par ordinateur et d'algorithmes d'apprentissage. Elle existe depuis une décennie et est publiée sous la licence Berkeley Software Distribution (BSD), ce qui facilite l'utilisation et la modification du code par les utilisateurs.

Les modules intégrés d'OpenCV sont suffisamment puissants et polyvalents pour résoudre la plupart des problèmes de vision informatique pour lesquels des solutions bien établies sont

5. <https://pypi.org/project/PyQt5/>

6. <https://opencv.org/>

Chapitre III: Implémentation et réalisation

disponibles. Ils sont destinés à des applications en temps réel et conçus pour être exécutés très rapidement.

OpenCV a été officiellement lancé en tant que projet de recherche au sein d'Intel Research afin de faire progresser les technologies dans les applications gourmandes en ressources. Parmi les principaux contributeurs au projet figuraient des membres d'Intel Research Russia et de l'équipe Performance Library d'Intel.

III. 3 Configuration matérielle

III.3.1 Configuration matérielle distante (Google Colab)

Google Collaboratory⁷ ou Colab¹, un outil Google simple et gratuit pour vous initier au Réseaux profonds ou collaborer avec vos collègues sur des projets en science des données .Colab permet d'améliorer vos compétences de codage en langage de programmation Python, de développer des applications en Réseaux Profonds en utilisant des bibliothèques populaires telles que Keras, TensorFlow, PyTorch et OpenCV sans installation ainsi que d'utiliser un environnement de développement (Jupyter Notebook) qui ne nécessite aucune configuration. Cependant, chaque 12 heures, la machine virtuelle mise à disposition par Google est réinitialisée nécessitant un mécanisme de sauvegarde des données en cours .De plus, les documents Colab (Jupyter Notebook) sont enregistrés directement votre compte Google Drive.

L'infrastructure distante mise à disposition par Google Colab et utilisée pour l'entraînement possède cette configuration :

- Processeur Intel Core Xeon CPU @2.3Ghz, 45MB Cache.
- Processeur graphique NVIDIA Tesla K80, ayant 2496 cœurs CUDA, Compute 3,7, 12 Go (11.439 Go utilisable) GDDR5 VRAM.
- Mémoire vive de 12.6 Go.
- Disque dur de capacité 320 Go.
- Jupyter Notebook
- Système d'exploitation Linux x64 bits.

7. <https://colab.research.google.com>

Chapitre III: Implémentation et réalisation

III.3.2 Configuration matérielle local

Pour la reconnaissance, nous avons utilisé un pc portable personnel possédant cette configuration:

- Processeur Intel Core i5-9400f CPU @3.5Ghz, 5.0 GhZ.
- Processeur graphique NVIDIA GEFORCE GTX 1060 3Go VRAM
- Mémoire vive de 8 Go
- Disque dur hybride SSD de capacité 256 Go et HDD de capacité 1 To
- Système d'exploitation Windows 10 x64 bits

III.3.3 Les base d'images

Kaggle⁸ est une collection de données scientifiques. Kaggle permet aux utilisateurs de rechercher et de publier des ensembles de données et des explorateurs, de créer des modèles dans un environnement de science des données basé sur le Web, de travailler avec d'autres données scientifiques et des moteurs de correspondance automatique, et de participer à des concours pour résoudre les failles de données.

III.3.4 Architecture de notre réseau

Au cours de nos expérimentations, nous avons créé trois modèles (modèle 1, modèle 2 et modèle 3) avec différentes architectures, où on a appliqué les trois modèles sur la base d'images

Dans ce qui suit on présente l'architecture des trois modèles :

Architecture du modèle 01

Le premier modèle que nous présentons dans la figure III.1 est composé de cinq couches de convolution et deux couches de maxpooling et trois couches de fullyconnected.

L'image en entrée est de taille 244*244, l'image passe d'abord à la première couche de convolution. Cette couche est composée de 32 filtres de taille 3*3, Chacune de nos couches de convolution est suivie d'une fonction d'activation ReLU cette fonction force les neurones à retourner des valeurs positives, après cette convolution 32 featuresmaps de taille 242*242 seront créés.

8. <https://www.kaggle.com/>

Chapitre III: Implémentation et réalisation

Les 32 featuremaps qui sont obtenus auparavant sont donnés en entrée de la deuxième couche de convolution qui est composée aussi de 32 filtres, une fonction d'activation RELU est appliquée sur la couche de convolution, ensuite on applique Maxpooling pour réduire la taille de l'image, nombre des paramètres et donc le temps de calcul. À la sortie de cette couche, nous aurons 32 featuremaps de taille 120*120.

On répète la même chose avec les couches de convolutions trois, quatre et cinq, ces couches sont composées de 64 filtres, la fonction d'activation ReLU est appliquée toujours sur chaque convolution. Une couche de Maxpooling est appliquée après la couche de convolution cinq. À la sortie de cette couche, nous aurons 64 featuremaps de taille 57*57. Le vecteur de caractéristiques issu des convolutions a une dimension de 207936.

Après ces cinq couches de convolution, nous utilisons un réseau de neurones composé de 3 couches fullyconnected. La première couches est 1 024 neurones où la fonction d'activation utilisée est le ReLU, La 2ème couches est 512 neurones où la fonction d'activation utilisée est le ReLU, et la 3ème couche est un sigmoid qui permet de calculer la distribution de probabilité des 2 classes (nombre de classe dans la base d'image).

Layer (type)	Output Shape	Param #
conv2d_9 (Conv2D)	(None, 242, 242, 32)	896
conv2d_10 (Conv2D)	(None, 240, 240, 32)	9248
max_pooling2d_6 (MaxPooling2D)	(None, 120, 120, 32)	0
conv2d_11 (Conv2D)	(None, 118, 118, 64)	18496
conv2d_12 (Conv2D)	(None, 116, 116, 64)	36928
conv2d_13 (Conv2D)	(None, 114, 114, 64)	36928
max_pooling2d_7 (MaxPooling2D)	(None, 57, 57, 64)	0
flatten_2 (Flatten)	(None, 207936)	0
dense_6 (Dense)	(None, 1024)	212927488
dropout_4 (Dropout)	(None, 1024)	0
dense_7 (Dense)	(None, 512)	524800
dropout_5 (Dropout)	(None, 512)	0
dense_8 (Dense)	(None, 1)	513

figure III.1 Configuration du modèle 1

Chapitre III: Implémentation et réalisation

Architecture du modèle 02

Le deuxième modèle que nous présentons dans la figure III.2 est composé de Trois couches de convolution et trois couches de maxpooling et deux couches de fullyconnected. L'image en entrée est de taille 244×244 , l'image passe d'abord à la première couche de convolution. Cette couche est composée de 32 filtres de taille 5×5 , Chacune de nos couches de convolution est suivie d'une fonction d'activation ReLU cette fonction force les neurones à retourner des valeurs positives, après cette convolution 32 featuremaps de taille 240×240 seront créés.

Ensuite on applique Maxpooling pour réduire la taille de l'image ainsi la quantité de paramètres et de calcul. À la sortie de cette couche, nous aurons 32 featuremaps de taille 120×120

Les 32 featuremaps qui sont obtenus auparavant ils sont donnés en entrée de la deuxième couche de convolution qui est composée aussi de 64 filtres, une fonction d'activation RELU est appliquée sur la couche de convolution, ensuite on applique Maxpooling pour réduire la taille de l'image ainsi que le nombre de paramètres et le temps de calcul. À la sortie de cette couche, nous aurons 60featuremaps de taille 58×58 .

On répète la même chose avec la couche de convolution trois, Cette couche est composée de 128 filtres, la fonction d'activation ReLU est appliquée toujours sur chaque convolution. Une couche de Maxpooling est appliquée après la couche de convolution trois. À la sortie de cette couche, nous aurons 128 featuremaps de taille 27×27 . Le vecteur de caractéristiques issu des convolutions a une dimension de 93312.

Après ces trois couches de convolution, nous utilisons un réseau de neurones composé de deux couches fullyconnected. la première couche est composé 1 024 neurones où la fonction d'activation utilisée est le ReLU, et la Deuxième couche est un sigmoid qui permet de calculer la distribution de probabilité des 2 classes (nombre de classe dans la base d'image).

Chapitre III: Implémentation et réalisation

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 240, 240, 32)	2432
max_pooling2d (MaxPooling2D)	(None, 120, 120, 32)	0
conv2d_1 (Conv2D)	(None, 116, 116, 64)	51264
max_pooling2d_1 (MaxPooling2D)	(None, 58, 58, 64)	0
conv2d_2 (Conv2D)	(None, 54, 54, 128)	204928
max_pooling2d_2 (MaxPooling2D)	(None, 27, 27, 128)	0
flatten (Flatten)	(None, 93312)	0
dense (Dense)	(None, 1024)	95552512
dropout (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 2)	2050
=====		
Total params: 95,813,186		
Trainable params: 95,813,186		
Non-trainable params: 0		

figure III.2 Configuration du modèle 2

Architecture du modèle 03

Le troisième modèle que nous présentons dans la figure III.3 est composé de six couches de convolution et trois couches de maxpooling et de trois couches de fullyconnected.

L'image en entrée est de taille 244*244, l'image passe d'abord à la première couche de convolution. Cette couche est composée de 32 filtres de taille 3*3, la fonction d'activation ReLU est utilisé, cette fonction d'activation force les neurones à retourner des valeurs positives, après cette convolution 32 featuremaps de taille 242*242 seront créés.

Ensuite, les 32 featuremaps qui sont obtenus ils sont données en entrée de la deuxième couche de convolution qui est composée aussi de 32 filtres. La fonction d'activation ReLU est appliquée sur les couches de convolutions. Le Maxpooling est appliqué après pour réduire la taille de l'image et des paramètres. À la sortie de cette couche, nous aurons 32 featuremaps de taille 120*120.

Chapitre III: Implémentation et réalisation

On répète la même chose avec les couches de convolutions trois, quatre, cinq et six (les couches trois et quatre sont composées de 64 filtres et les couches cinq et six sont composées de 128 filtres), la fonction d'activation ReLU est appliquée toujours sur chaque convolution. Une couche de Maxpooling est appliquée après les couches de convolutions quatre et six. À la sortie de la dernière couche Maxpooling, nous aurons 128 featuremaps de taille 27*27. Le vecteur de caractéristiques issu des convolutions a une dimension de 93312.

Après ces six couches de convolution, nous utilisons un réseau de neurones composé de 4 couches fullyconnected La première couches est 1 024 neurones où la fonction d'activation utilisée est le ReLU, La 2ème couches est 512 neurones où la fonction d'activation utilisée est le ReLU, La 3ème couches est 128 neurones où la fonction d'activation utilisée est le ReLU, et la 4ème couche est un sgmoid qui permet de calculer la distribution de probabilité des 2 classes (nombre de classe dans la base d'image).

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 242, 242, 32)	896
conv2d_1 (Conv2D)	(None, 240, 240, 32)	9248
max_pooling2d (MaxPooling2D)	(None, 120, 120, 32)	0
conv2d_2 (Conv2D)	(None, 118, 118, 64)	18496
conv2d_3 (Conv2D)	(None, 116, 116, 64)	36928
max_pooling2d_1 (MaxPooling2D)	(None, 58, 58, 64)	0
conv2d_4 (Conv2D)	(None, 56, 56, 128)	73856
conv2d_5 (Conv2D)	(None, 54, 54, 128)	147584
max_pooling2d_2 (MaxPooling2D)	(None, 27, 27, 128)	0
flatten (Flatten)	(None, 93312)	0
dense (Dense)	(None, 1024)	95552512
dropout (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 512)	524800
dropout_1 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 128)	65664
dropout_2 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 1)	129

figure III.3 Configuration du modèle 3

Chapitre III: Implémentation et réalisation

III.3.5 Résultats obtenus et discussion

Résultats obtenus pour le modèle 1

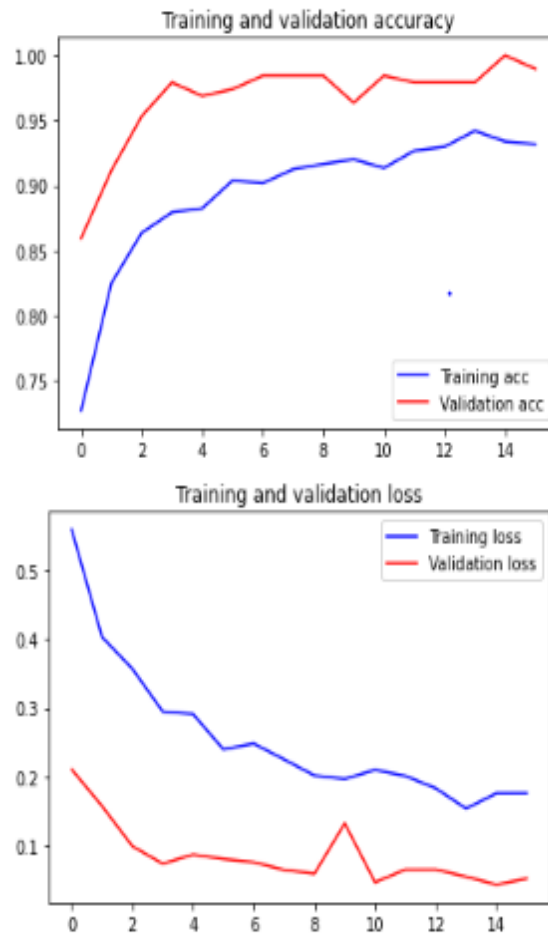


Figure III.4 Précision et Erreur pour le Modèle 01

Chapitre III: Implémentation et réalisation

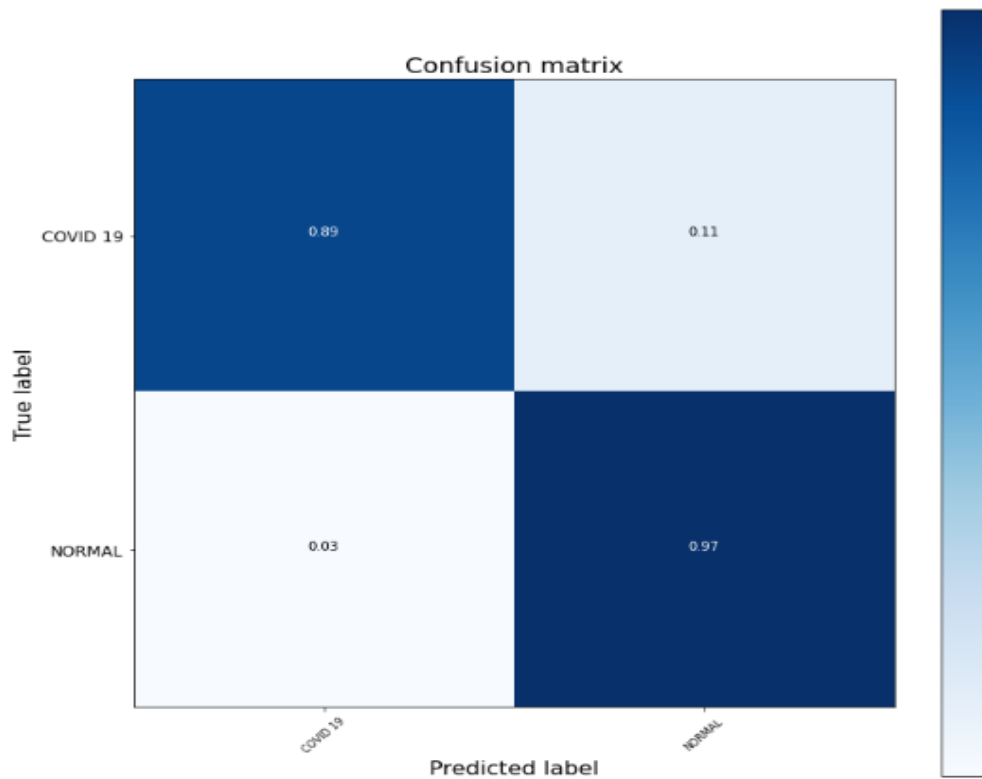


Figure III.5 Matrice de Confusion pour le Modèle 01
Résultats obtenus pour le modèle 2

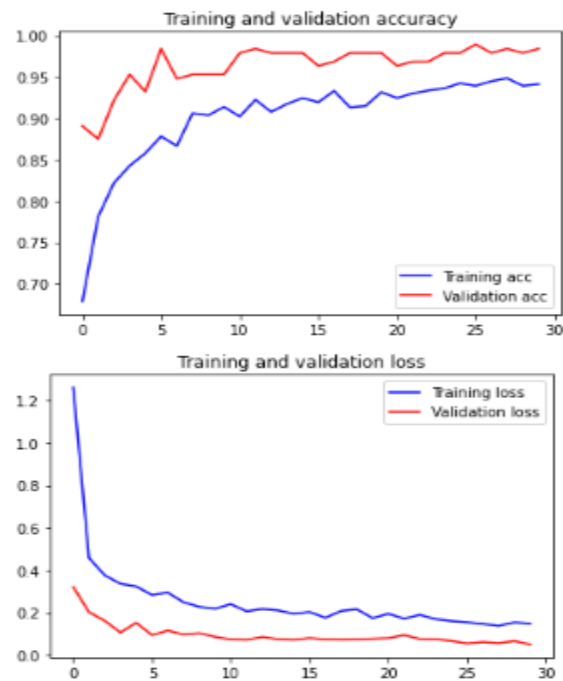


Figure III.6 Précision et Erreur pour le Modèle 02

Chapitre III: Implémentation et réalisation

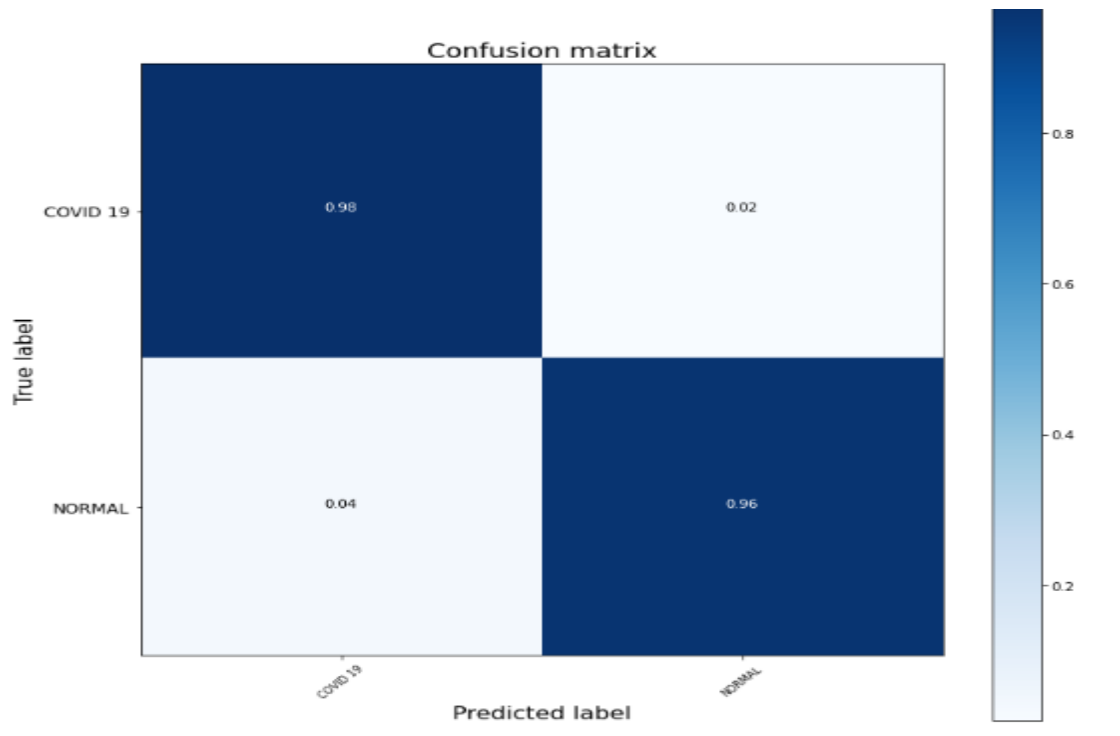


Figure III.7 Matrice de Confusion pour le Modèle 02

Résultats obtenus pour le modèle 3

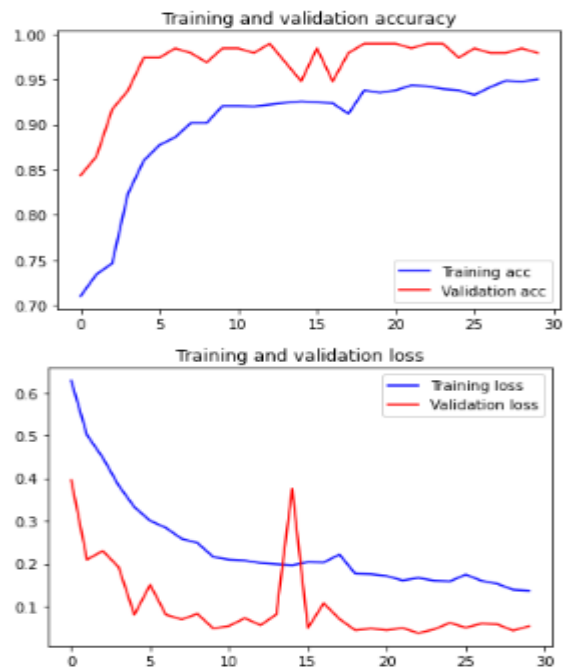


Figure III.8 Précision et Erreur pour le Modèle 03

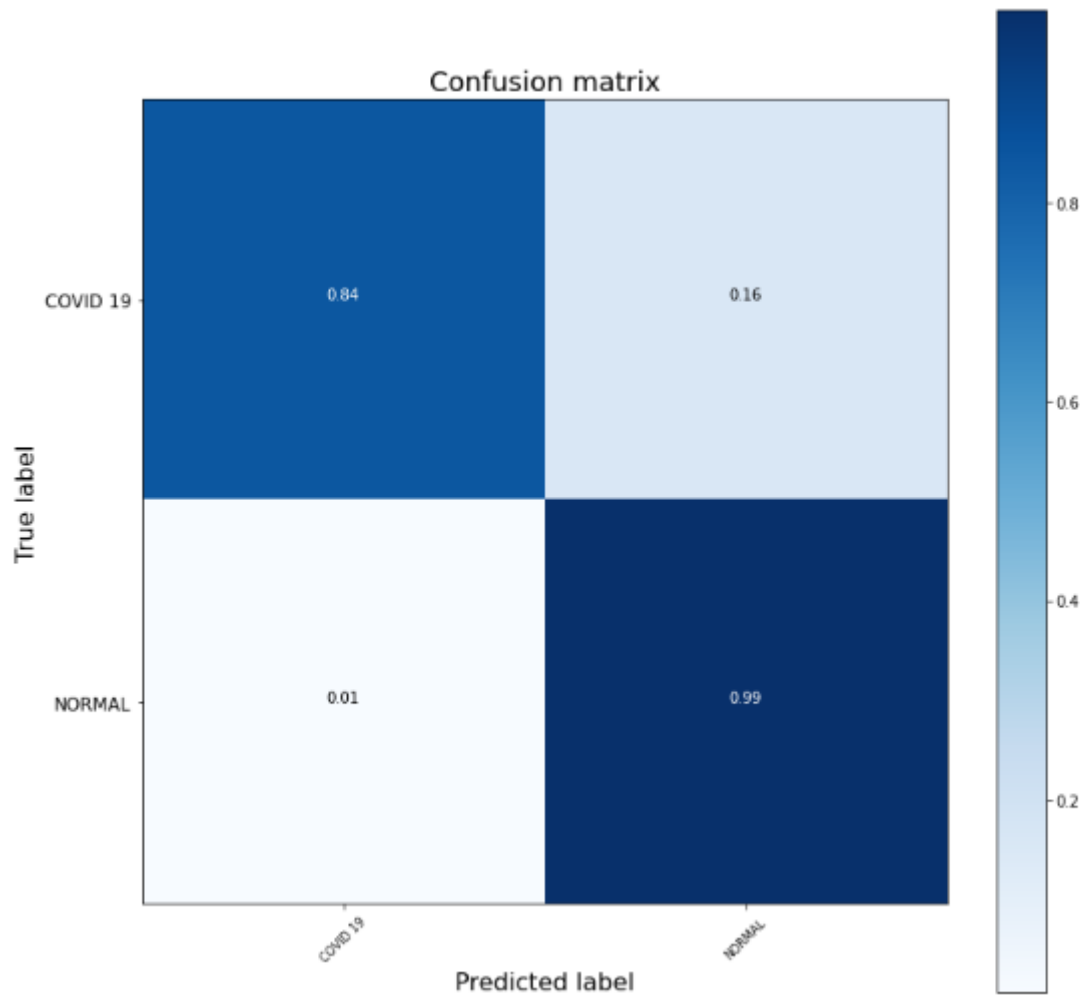


Figure III.9 Matrice de Confusion pour le Modèle 03

Après l'analyse des résultats obtenus, On constate les remarques suivantes :

La précision de l'apprentissage et de test augmente avec le nombre d'époques, ceci reflète qu'à chaque époque le modèle apprend plus d'informations. Si la précision est diminuée alors on aura besoin de plus d'information pour faire apprendre notre modèle et par conséquent on doit augmenter le nombre d'époques et vice versa. De même, l'erreur d'apprentissage et de la validation diminue avec le nombre d'époques.

Nous remarquons aussi que la totalité des images mal classées est de 53 images à partir de 337 pour le modèle 1, 21 images à partir de 337 pour le modèle 2 et 58 images à partir de 337 pour le modèle 3.

Chapitre III: Implémentation et réalisation

Les matrices de confusion permettent d'évaluer la performance de nos modèles, puisqu'elle reflète les métriques du Vrai Positif, Vrai Négatif, Faux Positif et Faux Négatif

	Architecture utilisée			Nombre époques	Précision obtenu Sur la base d'apprentissage	Précision obtenu Sur la base de validation	Temps d'exécution
	Couche de conv	Couche de Pooling	Fully Connected				
Model 01	5	2	3	10	91%	71%	900 s 15 min
Model 02	3	3	2	30	95%	83%	2040 s 30 min
Model 03	6	3	4	20	93%	74%	1320 s 22min

Le tableau montre l'architecture utilisée dans chaque modèle ainsi que le nombre d'époques (passages sur l'ensemble des exemples de la base d'apprentissage). Les résultats obtenus sont exprimés en termes de précision d'apprentissage, de validation, et enfin la dernière colonne représente le temps d'exécution. Nous remarquons que le modèle 2 présente les meilleurs résultats trouvés. Le nombre d'époques et de couches de convolution reflètent ces bons résultats, cependant le temps d'exécution était très coûteux (à cause du nombre d'époques)

Les résultats obtenus se sont améliorés au fur et à mesure que nous avons approfondie notre réseau et augmenté le nombre d'époques. La base d'apprentissage est également un élément déterminant dans les réseaux de neurones convolutionnels, il faut avoir une base d'apprentissage de grande taille pour aboutir à des meilleurs résultats.

III.4 Conclusion

Dans ce chapitre, nous avons expliqué la mise en œuvre de notre application ainsi que la configuration matérielle et logicielle utilisée. Nous avons également présenté nos résultats expérimentaux obtenus en faisant une étude comparative sur les trois modèles utilisés.

Conclusion générale

Conclusion générale

La classification d'images est une tâche importante dans le domaine de la vision par ordinateur, la reconnaissance d'objets et l'apprentissage automatique. Grâce à l'apprentissage en profondeur (Deep Learning), l'avenir de l'IA se développe très rapidement.

Dans ce projet nous avons entamé les notions fondamentales sur les réseaux de neurones en générale et les réseaux de neurones convolutionnels en particulier. Nous avons introduit ces réseaux de neurones convolutionnels en présentant les différents types de couches utilisées dans la classification: la couche convolutionnelle, la couche de correction, la couche de pooling et la couche de dropout. Nous avons parlé aussi sur les méthodes de régularisation (dropout et data augmentation) utilisées pour éviter le problème de sur apprentissage.

Pour réaliser notre travail de classification on a utilisé le deep learning, la méthode d'apprentissage qui a montré ses performances ces dernières années et nous avons choisi les CNNs comme réseau de neurones artificiels acycliques (feed-forward). Parmi les avantages des réseaux convolutifs est l'utilisation d'un poids unique associé aux signaux entrant dans tous les neurones d'un même noyau de convolution. Cette particularité nous a permis de réduire l'empreinte mémoire, améliorer les performances et garder une invariance du traitement par translation. Donc notre choix est justifié par la simplicité et l'efficacité de la méthode. Les résultats obtenus lors de la phase de test confirment l'efficacité de notre approche.

Notre travail n'est que dans sa version initiale, on peut dire que ce travail reste ouvert pour des travaux de comparaison et/ou d'hybridation avec d'autres méthodes de classification.

Bibliographie

Bibliographie

- [01] Naciri H., Chaoui N, Conception et Réalisation d'un système automatique d'identification des empreintes digitales, Mémoire de PFE, Université de Tlemcen, 2003.
- [02] Hadjila F. & Bouabdellah R, Reconnaissance des visages par les réseaux de neurones, Mémoire de PFE, Université de Tlemcen, 2003.
- [03] Rafael C. Gonzalez & Richard E. Woods, Digital Image Processing, Pearson Education Inc, 2008
- [04] Nasrin, Abdllaoui Nezha. la classification on hiérarchique axendante. 2013/2014.
- [05] Hirotk Suzuki, Pascal Matsakis. Exploitation de connaissances structurelles en classification on d'image: Utilisation de méthodes heuristiques d'optimisation combinatoire.\
- [06] BORG A., AKDAG H. Supervised Learning and Approximate. 2001.
- [07] R. S. Michalski, R.L. Chilausky. Knowledge acquisition by encoding expert rules versus computer induction from examples : a case study involving. 1981.
- [08] Parrochia. Classifications, histoire et problèmes formels. Cinquièmes Rencontres de la société Francophone de classification SFC'97. Lyon : s.n., Septembre 1997.
- [09] R.E. Stepp, R.S. Michalski. Learning from Observation: Conceptual Clustering, Machine Learning, An Artificial Intelligence Approach. s.l. : Morgan Kaufman Publishers, 1983.
- [10]. Cours d'analyse numérique : réseaux de neurones formels (<http://informatique.coursgratuits.net/methodes-numeriques/reseaux-deneurones-formels.php>)
- [11]. LNIA - Laboratoire de Neurosciences intégratives et adaptatives Claude Touze
- [12]. Jean-Paul Haton, Modèles connexionnistes pour l'intelligence artificielle, 1989.
- [13]. Léon Personnaz et Isabelle Rivals, Réseaux de neurones formels pour la modélisation, la commande et la classification, CNRS Éditions, 2003.
- [14] Pierre Buysens, Fusion de différents modes de capture pour la reconnaissance du visage appliquée aux e_transactions DOCTORAT de l'UNIVERSITÉ de CAEN Le 4 Janvier 2011
- [15] Yann LeCun. Les enjeux de la recherche en intelligence artificielle. Interstices, 2016.
- [16] Grégory Gelly. Réseaux de neurones récurrents pour le traitement automatique de la parole. PhD thesis, Université Paris-Saclay, 2017.
- [17] Sébatien Frizzi, Rabeb Kaabi, Moez Bouchouicha, Jean-Marc Ginoux, Farhat Fnaiech, and Eric Moreau. Détection de la fumée et du feu par réseau de neurones convolutifs. In Conférence Nationale sur les Applications Pratiques de l'Intelligence Artificielle, 2017.

Bibliographie

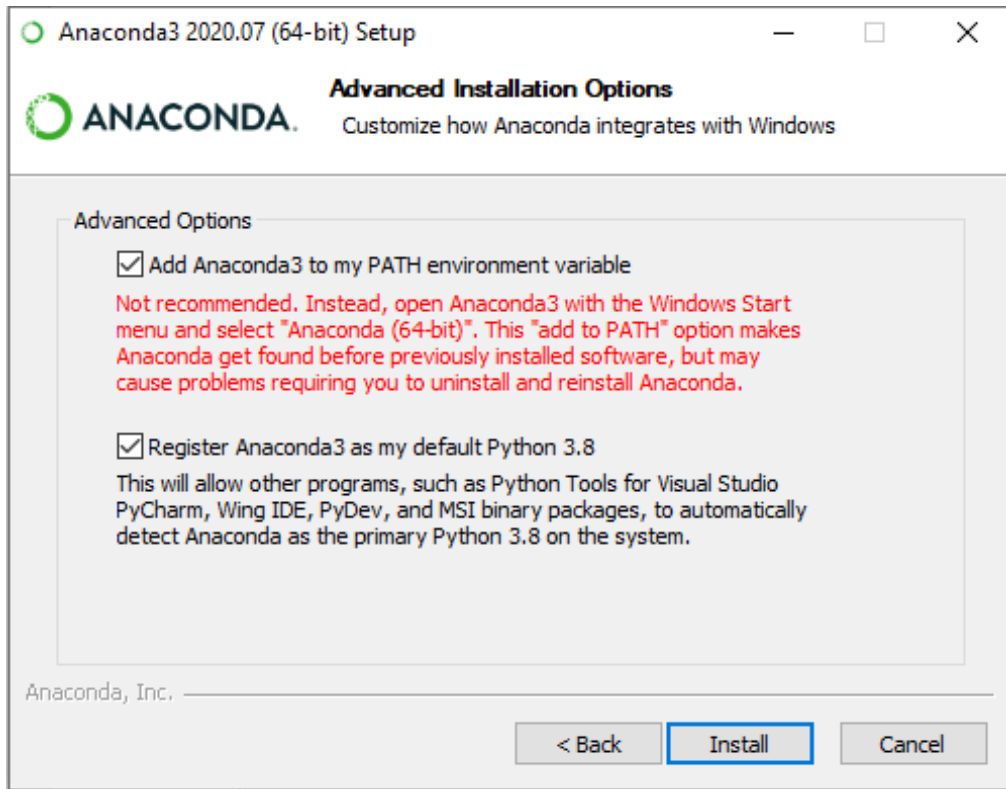
- [18] Dan Cireşan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. arXiv preprint arXiv :1202.2745, 2012.
- [19] Sparsh Mittal. A survey of fpga-based accelerators for convolutional neural networks. Neural computing and applications, pages 1–31, 2018.
- [20] Francois Chollet. Deep Learning mit Python und Keras : Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek. MITP-Verlags GmbH & Co. KG, 2018
- [21] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout : a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1) :1929–1958, 2014.
- [22] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In D. Touretzky, editor, Neural Information Processing Systems, volume 2. Morgan Kaufman, 1990.

Installation de l'Environnement de développement logiciel sous Windows 10 (64 bits)

Installation de l'Environnement

A.1 Python Environment Setup with Anaconda Python

1. Rendez-vous sur la page de téléchargements Anaconda
<https://www.anaconda.com/downloadset> obtenez la version Python 3.8
2. Exécuter l'installateur "Anaconda3-2020.07-Windows-x86_64.exe"



3. Vérifiez votre installation
 - (a) Sous windows, dans un cmd, taper la commande suivante :

```
c:\User\hani> python --version Python
3.7
```

4. Mettre à jour Anaconda de base
 - (a) Sous windows, ouvrir Anaconda Prompt
 - (b) Dans un Anaconda Prompt, taper les commandes suivante :

```
(base) C:\Users\hani>conda update conda
(base) C:\Users\hani>conda update anaconda
(base) C:\Users\hani>conda update python
(base) C:\Users\hani>conda update --all
```

Installation de l'Environnement

Cela devrait mettre à jour toute votre base Anaconda jusqu'aux derniers paquets. 5.
Créer un Python "environnement virtuel0"

A.2 Install Opencv, Numpy et Matplotlib

Dans un Anaconda Prompt, taper les commandes suivantes :

```
(app_final) C : \Users\hani>conda install -c michael_wildopencv-contrib  
(app_final) C : \Users\hani>conda install -c conda-forge matplotlib  
(app_final) C : \Users\hani>conda install -c anaconda numpy
```

A.3 Install Keras et Tensorflow

1. Ouvrir Anaconda Prompt avec les droits d'administration.
2. Dans un Anaconda Prompt, taper les commandes suivante :

```
(base) C : \Users\hani>conda activate app_final  
(app_final) C : \Users\hani>conda install keras-gpu
```

- (a) Install Tensorflow official :

```
(app_final) C : \Users\hani>conda install tensorflow-gpu
```

- (b) Install Tensorflow pour Windows 10:

```
(app_final) C : \Users\hani>conda install -c aaronzstensorflow-gpu  
(app_final) C : \Users\hani>conda install -c anaconda cudatoolkit  
(app_final) C : \Users\hani>conda install -c anaconda cudnn
```

A.4 Install CUDA et cuDNN

Ces étapes d'installation CUDA et cuDNN sont vaguement basées sur le guide d'installation Nvidia CUDA.

1. Télécharger CUDA sur le site Web de Nvidia <https://developer.nvidia.com/cuda-downloads>.

Installation de l'Environnement

Select Target Platform

Click on the green buttons that describe your target platform. Only supported platforms will be shown. E the terms and conditions of the [CUDA EULA](#).

Operating System

Linux

Windows

Architecture

x86_64

Version

10

Server 2019

Server 2016

Installer Type


exe (local)

exe (network)

Download Installer for Windows 10 x86_64

The base installer is available for download below.

> Base Installer

Download (3.1 GB) 

Installation Instructions:

1. Double click cuda_11.1.0_456.43_win10.exe
2. Follow on-screen prompts

The checksums for the installer and patches can be found in [Installer Checksums](#).
For further information, see the [Installation Guide for Microsoft Windows](#) and the [CUDA Quick Start Guide](#).

2. Installer CUDA.
3. TelechargercuDNN sur le site Web de Nvidia Programme des développeurs <https://developer.nvidia.com/cudnn>.

Installation de l'Environnement

cuDNN Download

NVIDIA cuDNN is a GPU-accelerated library of primitives for deep neural networks.

☒ I Agree To the Terms of the [cuDNN Software License Agreement](#)

Note: Please refer to the [Installation Guide](#) for release prerequisites, including supported GPU architectures and compute capabilities, before downloading.

For more information, refer to the cuDNN Developer Guide, Installation Guide and Release Notes on the [Deep Learning SDK Documentation](#) web page.

[Download cuDNN v8.0.4 \(September 28th, 2020\), for CUDA 11.1](#)

[Download cuDNN v8.0.4 \(September 28th, 2020\), for CUDA 11.0](#)

[Download cuDNN v8.0.4 \(September 28th, 2020\), for CUDA 10.2](#)

[Download cuDNN v8.0.4 \(September 28th, 2020\), for CUDA 10.1](#)

[Archived cuDNN Releases](#)

cuDNN Download

NVIDIA cuDNN is a GPU-accelerated library of primitives for deep neural networks.

☒ I Agree To the Terms of the [cuDNN Software License Agreement](#)

Note: Please refer to the [Installation Guide](#) for release prerequisites, including supported GPU architectures and compute capabilities, before downloading.

For more information, refer to the cuDNN Developer Guide, Installation Guide and Release Notes on the [Deep Learning SDK Documentation](#) web page.

[Download cuDNN v7.4.2 \(Dec 14, 2018\), for CUDA 10.0](#)

[Download cuDNN v7.4.2 \(Dec 14, 2018\), for CUDA 9.2](#)

[Download cuDNN v7.4.2 \(Dec 14, 2018\), for CUDA 9.0](#)

[Archived cuDNN Releases](#)



4. Dézipper les fichiers cuDNN et les copier dans les dossiers CUDA.

RÉSUMÉ

Les réseaux de neurones convolutionnels sont des réseaux de neurones multicouches qui sont spécialisés dans des tâches de reconnaissance de forme. Dans notre travail on a utilisé ce type de réseaux pour la classification des images, alors on a proposé trois modèles avec différentes architectures (le nombre des couches de convolutions, des couches de pooling, des couches entièrement connecté et le nombre d'époque). Les résultats obtenus ont montré que le choix du nombre d'époque et la taille de la base d'images ainsi que la profondeur du réseau ont une grande influence pour avoir des meilleurs résultats

ABSTRACT

Convolutional neural networks are multi-layered neural networks that are specialized in pattern recognition tasks. In our work we used this type of networks for images classification, then we proposed three models with different architectures (the number of convolutions layer, pooling layer, fully connected layers and the number of epochs). The results obtained showed that the choice of the epoch number, the size of the image database and the depth of the network have a great influence to have better results

الملخص

الشبكات العصبونية الالتفافية هي شبكات عصبية متعددة الطبقات تتمثل مهامها في التعرف على الأنماط. في عملنا قمنا باستخدام هذه الشبكات لتصنيف الصور حيث اقترحنا ثلاث نماذج مختلفة (عدد الطبقات الالتفافية, الطبقات الانتقائية, طبقات الاتصال الكامل و عدد التكرارات). النتائج المتحصل عليها أظهرت أن اختيار عدد التكرارات, حجم قاعدة الصور و عمق الشبكة ليها تأثير كبير للحصول على نتائج أفضل.