TABLE DES MATIERES

NOTATIONS ET ABREVIATIONS

1. Minuscules latines

b	Bias d'une connexion d'un réseau de neurone
c	Pondération d'un terme dans un document
d	Document à traiter
f	Fonction d'activation
g	Transformation affine
l	Seuil d'une fonction linéaire à seuil
m	Seuil d'une fonction linéaire à seuil
n	Vitesse d'apprentissage
q	requête
u	Valeur de sommation de la sortie d'un neurone
w	Poids d'une connexion d'un réseau de neurone
x	Entrée d'un neurone
y	Sortie du neurone

2. Majuscules latines

A	Nombre d'images dans l'ensemble de réponse
B	Bias d'une connexion d'un réseau de neurone
E	Erreur de prédiction d'un réseau de neurone
F	Taille d'un filtre d'une couche de convolution
I	Entrée d'une convolution
J	Fonction coût d'un réseau
K	Nombre de filtre d'une couche de convolution

M	Matrice
N	Document pertinent
P	Probabilité
R	Sous ensemble de documents
R	Nombre d'images pertinentes dans une base
S	Sortie du neurone
U	Noyau de convolution
W	Vecteur de poids augmenté
Z	Résultat d'une convolution
X	Vecteur d'entrée augmenté

3. Minuscules grecques

α	Paramètre d'une function d'activation exponentielle
β	Paramètre d'une normalisation de lot
γ	Paramètre d'une normalisation de lot
δ	Gradient à une couche définie
θ	Les paramètres d'un réseau de neurone : poids et bias
φ	Fonction Sigmoïde
∂	Dérivé

4. Abréviations

Adaline	Adaptive Linear Element
BN	Batch Normalization
CBIR	Content Based Image Retrieval
CIE	Commission International d'Eclairage
Cl	chlore
CNN	Convolutional Neural Network

CPU	Compute Processor Unit
CUDA	Compute Unified Device Architechture
DBM	Deep Boltzman Machine
df	document frequency
ELU	Exponential Linear Unit
FMP	Fractional max pooling
GCH	Global Color Histogram
GLCM	Gray-level Co-occurrence Matrix
GPU	Graphical Processor Unit)
HTML	HyperText Markup Language
ILSVRC	ImageNet Large-Scale Visual Recognition Challenge
K +	potassium
LCH	Local Color Histogram
LSTM	Long Short Term Memory
MLP	Multi Layer Perceptrons
MNIST	Mixed National Institute of Standards and Technology)
Na+	sodium
OpenCV	Open Computer Vision
PK	Permeabilité au potassium
PNa	Permeablité au sodium
ReLU	Rectified Linear Unit
RNA	Reseau de Neurone Artificiel
RVB	Rouge Vert Bleu
SGD	Stochastic Gradient Descent
SIFT	Scale-invariant feature transform

SOM	Self Organizing Map
SVM	Support Vector Machine
tf	term frequency
TSV	Teinte Saturation Valeur
URL	Uniform Resource Locator
VGG	Visual Geometry Group

INTRODUCTION GENERALE

Ces dernières années, des larges collections d'image et de vidéos ont rapidement émergé. La production d'images numériques a connu un essor considérable, tant chez les professionnels de l'image que chez les amateurs avec l'utilisation croissante mondiale des appareils photos numériques et des téléphones portables. Les capacités de stockage sur les divers supports repoussent de plus en plus loin les limites de quantité possible de prises de vues. Devant un tel contexte d'explosion de contenu numérique, se pose le problème de l'organisation et de la recherche.

Avant les bases d'images était majoritairement indexées manuellement, via l'association de métadonnées sous la forme de mots clés et autres informations circonstancielles sur la prise de vue. Ces annotations, quoique fiables, ne sont pas exemptes de défaut. Le principal défaut étant précisément le volume de données lui-même, qui rend la tâche particulièrement longue et fastidieuse. Ce problème décuple l'impact de tous les autres : les informations ne sont à priori annotées qu'en un nombre limité de langues, elles doivent aussi se conformer à un format dont toute modification entraîne une nouvelle tâche d'annotation. Il y a aussi la notion de subjectivité qui rend la tâche d'annotation pas forcément reproductible car deux annotateurs différents ne produiront pas systématiquement la même annotation pour une même image.

C'est donc tout naturellement qu'a émergé la recherche pour une solution informatique au problème. C'est ainsi qu'ont rapidement fleuri, d'un côté des systèmes de recherche par le contenu (dits systèmes Content Based Image Retrieval, ou CBIR) et d'un autre des systèmes d'indexation automatique fin d'accéder à ces informations visuelles. Si ces deux mécanismes se diffèrent par leurs approches respectives.

L'objectif général de ce mémoire est de développer une solution d'indexation et de recherche performante et qui réduit le « fossé sémantique » bien présent dans les solutions appliquées aujourd'hui. On entend par fossé sémantique le fossé qui sépare la "sémantique" de l'acquisition de données à partir des pixels de l'image. Intuitivement, il s'agit d'étiqueter les images automatiquement par des concepts que l'on perçoit dans celles-ci : personne, train, montagne, paysage, etc. Notre approche permettra donc à l'utilisateur de rechercher des informations par des mots qu'il utilise au quotidien.

La solution qu'on a retenue est l'amélioration de l'indexation textuelle d'image par l'utilisation de réseau de neurones à convolution d'où le titre du mémoire : « Indexation et recherche d′image par réseau de neurone profond à convolution ».

Ce mémoire est organisé de la manière suivante. Le premier chapitre décrit les systèmes d'indexation et de recherche d'image. Le choix du sujet y sera référé ainsi que les technologies à utiliser.

Le second chapitre aborde le réseau de neurones et ses différents aspects. On y parle de l'origine biologique de ces réseaux mais également de son fonctionnement et de l'apprentissage.

Le troisième chapitre est consacré aux réseaux de neurones à convolution qu'on a utilisés dans ce mémoire. Différents aspects seront évoqués comme les caractéristiques de ces réseaux, la technique de visualisation et d'apprentissage. Nous verrons aussi les architectures de référence dans ce domaine ainsi que des technologies innovantes.

Dans le quatrième chapitre, nous parlerons de la conception d'un réseau de neurone à convolution pour l'indexation textuelle. Nous y verrons le dimensionnement d'un réseau de neurone à convolution, les diverses contraintes ainsi que le choix des couches.

Le dernier chapitre sera pour l'implémentation de notre modèle avec le choix du simulateur, et les étapes nécessaires. En dernier lieu nous discuterons les résultats obtenus et proposerons quelques suggestions.

CHAPITRE 1
GENERALITE SUR LA RECHERCHE D'IMAGE

1.1 Introduction

La numérisation croissante et l'omniprésence de l'outil informatique a rendu l'information devenue facile à fabriquer et à manipuler. De plus, pouvant être dupliquée et stockée à faible coût, l'information a quantitativement explosé. Cette disponibilité de l'information, associée à sa masse, provoque des changements dans le comportement des individus mais entraîne également de nouveaux besoins. Un de ces besoins, parmi les plus évidents, est à l'origine de la branche de l'informatique, la recherche d'information, traitée ici.

L'indexation et la recherche d'images constitue des solutions les plus prometteuses pour la gestion des bases de données d'images numériques, qui ne cesse d'augmenter de jour en jour. Elle vise à résoudre ce problème en se basant sur une représentation de bas niveau du contenu de l'image : par la couleur, la texture, la forme, etc., ou par des représentations de haut niveau : tag, label, description. Dans ce domaine, plusieurs méthodes d'indexation peuvent être recensées. Elles convergent toutes vers une description ou la caractérisation de l'image dans un domaine descriptif mais complexe et adaptatif [1] [2].

Dans ce chapitre nous allons étudier les différentes méthodes, populaires et performantes, dans le domaine de la recherche et indexation d'images. En étudiant les forces et les faiblesses des techniques utilisées dans la recherche de l'indexation d'images, nous pourrons présenter une solution plus rapide et plus performante.

Pour cela nous verrons en premier lieu l'architecture classique d'un système d'indexation et de recherche d'images, ensuite nous verrons les points forts de ces systèmes et les points qui limitent leurs utilisations. En dernier lieu, la solution proposée est présentée avec les motivations de son utilisation.

1.2 Architecture d'un système d'indexation et de recherche d'image

Le problème d'indexation et recherche d'image se présente en deux aspects indissociables. Le premier consiste le mode de représentation des images et le second concerne l'utilisation de cette représentation dans le but de la recherche. [1] [2]

Les deux techniques possèdent chacun une phase d'indexation et une phase de recherche. L'indexation a pour but de représenter l'information de manière à faciliter son extraction. L'architecture classique d'un système d'indexation et de recherche d'images se décompose en deux phases de traitement : une phase d'indexation dit, hors ligne, extraction d'attribut à partir de l'image, stocké dans un vecteur numérique appelé descripteur visuel. Ensuite, ces caractéristiques sont stockées dans une base de données. L'autre phase de recherche, dit on-line, consiste à comparer le descripteur émis par l'utilisateur et le comparer avec les descripteurs de la base de données. [2][3][4]

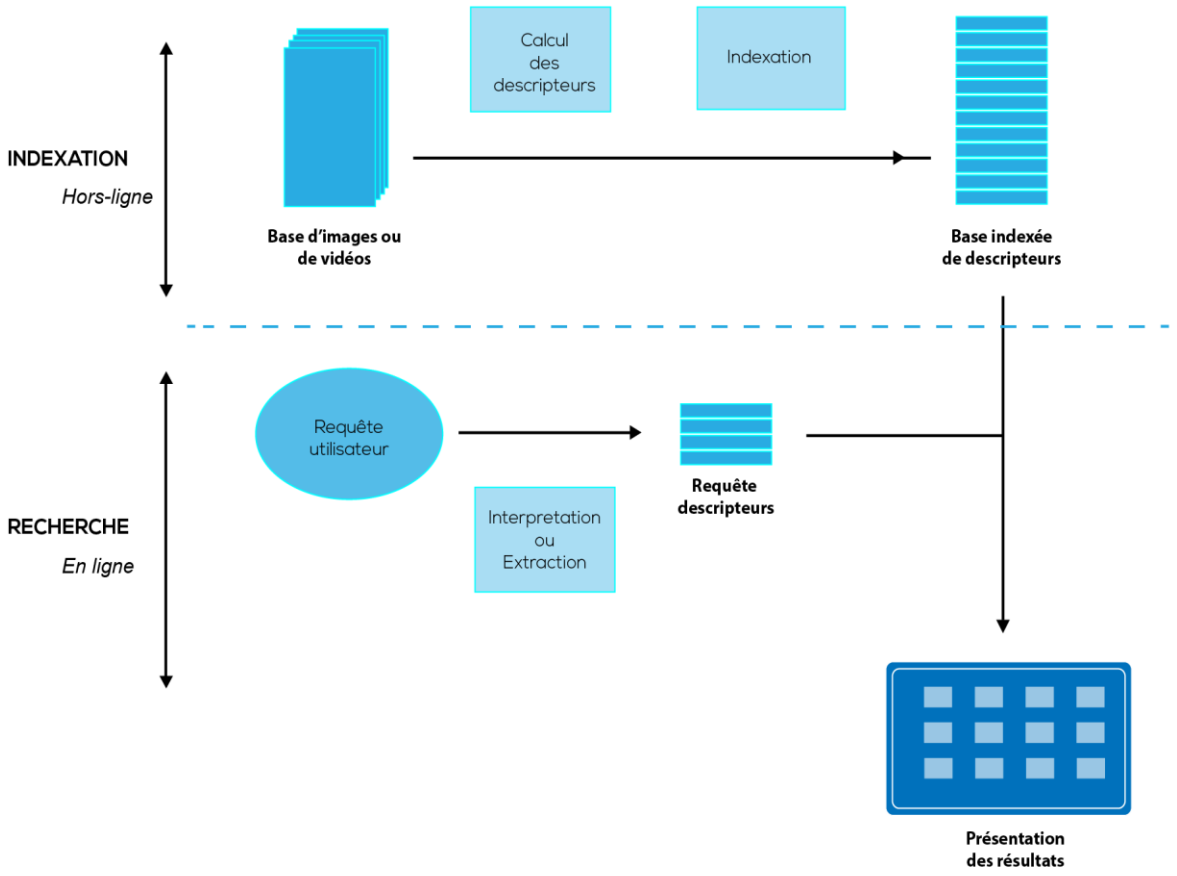

Figure 1.01 : *Les phases d'indexation et de recherche d'images*

1.3 Méthode Indexation et la recherche d'image

Les systèmes classiques de recherches d'image peuvent être classés en deux grandes classes :

- Les systèmes textuels de recherches d'images
- Les systèmes de recherches d'image par le contenue visuel (CBIR)

1.3.1 Indexation et recherche textuelle d'image

La méthode d'indexation textuelle peut se faire de deux manières distinctes, par une indexation manuelle et de façon automatique. [6]

1.3.1.1 Indexation textuelle manuelle

Lors de l'indexation textuelle manuelle, les images sont classées et indexées par un documentaliste appelé iconographe. Il a pour rôle d'associer les images à des groupes de mots. Il peut être aussi effectué par les utilisateurs qui souhaitent décrire leurs images personnelles.

Plusieurs problèmes peuvent être rencontrés pendant la phase d'indexation manuelle d'image. Le principal problème est le choix des termes. En effet, la personne qui indexe l'image et la personne qui recherche l'image ne choisira pas les mêmes termes. Une même image peuvent avoir plusieurs ses, contenir plusieurs sens, contenir plusieurs thèmes. On dit que l'indexation textuelle est subjective.

Il existe deux types d'indexation d'images. La première, le hard indexing, correspond à ce que l'indexeur voit dans l'image, par exemple le portrait d'une femme. La seconde est le soft indexing et porte sur la signification de l'image. [2][7]

Figure 1.02 : *Exemple d'images*

Malgré sa subjectivité, l'indexation manuelle reste une méthode efficace pour associer un sens à des images. Cependant, lorsque l'on a un grand volume d'images à indexer, ce travail devient fastidieux, voire impossible, ce qui n'est pas le cas pour l'indexation automatique.

1.3.1.2 Indexation textuelle automatique

L'indexation textuelle automatique d'image consiste à associer à une image au moyen d'un système informatique sans aucune intervention humaine. Il existe deux approches : l'indexation textuelle à partir du texte associé à l'image, et l'indexation textuelle automatique à partir du contenu visuel de l'image.

La première approche n'est possible que lorsque les images sont associées à du texte. C'est le cas des images des encyclopédies, des catalogues de vente, des manuels techniques... et aussi du web. L'indexation textuelle des images sur le web peut s'effectuer à partir des mots présents dans le titre de la page ou des mots les plus fréquents ou pertinents de cette page. Cependant, toutes les images présentes sur une même page web ne devraient pas être indexées avec les mêmes mots. Beaucoup de moteurs de recherche utilisent aussi l'Uniform Resource Locator (URL) et le nom de l'image, mais la plupart des images ne sont pas nommées de façon pertinente, mais bien souvent par des noms génériques comme img001.jpg qui ne portent pas de sens. D'autres techniques considèrent les mots associés à l'attribut « ALT » de la balise HyperText Markup Language (HTML) « IMG » d'une image ou bien le texte proche de l'image, ou bien une fusion de toutes ces informations. Mais, dans la pratique, peu d'images sur le web sont indexées de cette façon, car cela nécessite une indexation manuelle que l'on sait être très coûteuse en temps. De plus, le texte proche de l'image n'est pas forcément celui que l'on associerait à l'image. [8]

La deuxième approche est souvent appelée auto-annotation par le contenu. Annoter une image avec des mots seulement à partir du contenu visuel est impossible. C'est pourquoi la plupart du temps les méthodes d'auto-annotation sont en fait des méthodes de classification supervisée multi-classes (une classe par mot). Elles utilisent un ensemble d'apprentissage où les images sont associées aux classes de mots pour apprendre à prédire des mots sur de nouvelles images. C'est cette approche qui est utilisée dans ce mémoire. [7] [9]

1.3.1.3 Recherche d'image textuelle

Une fois indexées textuellement, les images peuvent être recherchées avec les modèles classiques de recherche dans les documents textuels. Pour cela il existe plusieurs méthodes : modèle booléen, modèle vectoriel, modèle probabiliste et modèle logique. Ces différentes techniques sont expliquées dans les paragraphes qui suivent. [7]

a. Modèle booléen

Dans le modèle booléen, un document di est représenté par une conjonction de termes indépendants que l'on représente sous la forme d'un ensemble : $d_i = \{c_1, c_2, \ldots\ldots, c_{nw_d}\}$ sans pondération. La requête est une expression logique composée de termes connectés par les opérateurs logiques ET, OU et NON. Une image sera jugé pertinente par le système si l'expression logique de la requête est

satisfaite par cette image. Ce modèle ne permet pas de retrouver des images qui ne correspondent que partiellement à la requête.

b. Modèle vectoriel

Dans le modèle vectoriel, un document est représenté sous la forme d'un vecteur à n_w dimensions : $d_i = \{w_{i,1}, w_{i,2}, \ldots\ldots, w_{in_w}\}$ où chaque $w_{i,j}$est la pondération associée au terme $w_i$dans le document : d_i. Ce modèle suppose que les vecteurs sont des points dans un espace où les termes forment une base orthogonale. Les termes sont supposés indépendants. La requête est exprimée selon le même formalisme : $d_i = \{w_{q,1}, w_{q,2}, \ldots\ldots, w_{q,j}\}$. Pour évaluer la pertinence d'un document par rapport à une requête, le système calcule une valeur de similarité entre les deux vecteurs d_i et q. Les mesures de similarité classiques sont le cosinus, la formule de Dice et la formule de Jaccard. Les pondérations tiennent compte de la fréquence du terme dans le document (term frequency, tf), du nombre de documents dans lesquels apparaît le terme (document frequency, df), de la longueur du document, et de l'apparition des termes d'indexation dans les parties logiques du document, comme le titre, le résumé. Etc.

c. Modèle probabiliste

Le modèle probabiliste essaye d'estimer la probabilité qu'un utilisateur a de trouver un document pertinent. Ce modèle suppose qu'il y a un sous-ensemble R de documents que l'utilisateur veut retrouver parmi ceux disponibles, les autres documents $\bar{R}$ étant considérés non pertinents. Un document d et une requête q sont représentés par un vecteur comme dans le modèle vectoriel, mais les poids sont binaires (le mot apparaît ou non dans le document). Si P(R|d) est la probabilité que le document d soit pertinent pour la requête q et si P($\bar{\text{R}}$|d) est la probabilité que le document ne soit pas pertinent pour la requête.

La similarité entre le document et la requête est :

$$sim(d,q) = \frac{P(R|d)}{P(\bar{\text{R}}|\text{d})} \qquad (1.01)$$

Ce modèle est difficile à mettre en œuvre en raison du calcul de probabilité initiale.

1.3.1.4 Avantages et limites

L'annotation textuelle d'images peut être effectuée manuellement, mais cela est très coûteux et subjectif. Elle peut également être effectuée automatiquement à partir du texte associé à l'image ou du contenu visuel, mais ces types de systèmes sont peu performants et les images sont finalement mal annotées. Cependant, l'avantage des systèmes textuels est qu'ils donnent la possibilité à l'utilisateur de poser des requêtes dans un langage de haut niveau lui permettant ainsi d'exprimer son besoin d'information facilement. De plus, les modèles, tels que le modèle vectoriel, permettent de retrouver rapidement et efficacement les documents répondant à une requête.

En ce qui concerne le problème de la subjectivité des termes, un thésaurus donnant les liens de synonymie et de hiérarchie entre les termes, tels que la strucutre WordNet, permet de réduire dans une certaine mesure les distances entre deux termes voisins . Cependant, il n'existe pas à notre connaissance de thésaurus hiérarchique conséquent spécialement adapté pour décrire les images.

Les modèles «sac de mots» ont l'avantage d'être très rapides, mais ils ne permettent pas de poser des requêtes plus complexes. Par exemple, rechercher des images avec une voiture rouge près d'une personne sur un fond en vert. L'approche conceptuelle permet de faire ce type de requête. Elle nécessite que les images soient décrites par des graphes représentant les relations entre les objets dans les images proposées et d'étendre le modèle vectoriel pour représenter les graphes conceptuels dans le cas de bases d'images.

1.3.2 Indexation et recherche d'images par le contenu visuel

Les systèmes de recherche d'images par le contenu visuel peuvent être construits avec différents objectifs en vue. Certains systèmes cherchent, à reconnaître un objet en particulier (un visage, un type d'objet...). On parle alors de reconnaissances de formes (pattern matching) ou d'objets (object recognition). Parmi ceux-ci, certains systèmes sont spécialisés pour travailler sur des images médicales ou des photos satellites.

Figure 1.03 : *Ensemble d'image sans thème*

D'autres cherchent à classer les images en fonction du type de scènes qu'elles représentent (naturel/artificiel, intérieur/extérieur...). On parle alors de reconnaissance de scènes ou de classification de scènes.

Une troisième catégorie de systèmes cherche à reconnaître les images similaires à une image requête. Les deux premiers types de systèmes nécessitent l'emploi de techniques et de descripteurs spécifiques à l'objet ou à la scène recherchée. Par contre, la troisième catégorie de systèmes traite des images généralistes et cherche à obtenir de bonnes performances en moyenne sur toutes les images. Dans le cadre de ce mémoire, nous nous intéressons surtout au troisième type de systèmes. [2] [3] [8]

Un cas particulier de systèmes généralistes est les bases de photographies personnelles. Ces systèmes sont généralistes dans le sens où ils doivent être capables de traiter tous types d'images, mais ce type de systèmes doit être simple et fournir des informations sur les images, comme la date de prise de vue, qui soit adaptée à l'utilisateur.

1.3.2.2 Extraction de descripteur

L'extraction de descripteur signifie l'extraction d'information compacte mais qui a de la valeur sémantique. Cette information est utilisée comme signature de l'image, son index. Les images similaires doivent avoir des signatures similaires. Si on regarde la figure, la couleur blanche et la texture du bâtiment sont des caractéristiques. On peut aussi prendre en compte la taille des objets dans l'image.

Figure 1.04 : *Exemple d'images à indexer*

La représentation des images doit considérer quel descripteur est le plus utile pour relater le contenu de l'image. L'extraction de descripteur de l'image est une opération effectué en off-line alors la complexité du traitement n'est pas pénalisante. Nous allons voir trois descripteurs communément utilisé : la couleur, la texture et la forme. [8][9][10][11]

a. La couleur

Une des plus importants descripteurs reconnus visuellement par l'homme dans une image est la couleur. La distinction des images par l'homme se base souvent sur la couleur. A cause de cela, le descripteur couleur est fortement utilisé dans les systèmes d'extraction d'image basé sur le contenu.

Le couleur est un descripteur puissant qui simplifie l'identification des objets. Afin d'extraire ce descripteur de l'image, une bonne représentation de l'espace de couleur est nécessaire.

L'objectif d'un espace couleur est de faciliter la spécification des couleurs. Chaque couleur dans l'espace de couleur est un point unique représentée dans un système de coordonnées. Plusieurs espaces de couleur, tel que RVB (Rouge Vert Bleu), TSV (Teinte Saturation Valeur), CIE (Commission International d'Eclairage) L*a*b, et CIE L*U*v, ont été élaborés à différentes fins. Bien qu'il n'y a pas d'accord sur l'espace de couleur qui est le meilleur pour CBIR, un système de couleur appropriée est nécessaire pour assurer l'uniformité perceptuelle. Par conséquent, l'espace couleur RVB, un système largement utilisé pour représenter les images couleur, n'est pas convenable pour CBIR parce que c'est un point de vue théorique non uniforme et dépendant des périphériques système.

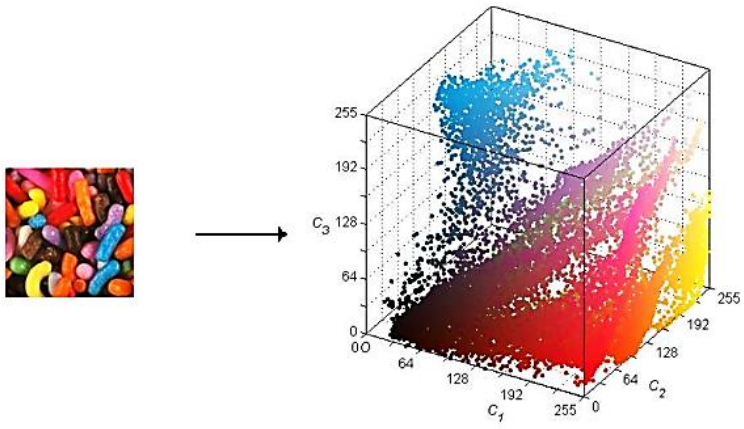

Figure 1.05 : *Représentation d'une image dans une espace de couleur (c1, c2, c3)*

La technique fréquemment utilisée consiste à convertir les représentations en couleurs à partir de la couleur RVB en l'espace pour le HSV, CIE L*U*v, ou CIE L*a*b espaces couleur avec uniformité perceptuelle. L'espace de couleur HSV est un système intuitif, qui décrit une couleur spécifique par ses valeurs de teinte, les valeurs de saturation et de luminosité. Ce système couleur est très utile en couleur interactive, la sélection et la manipulation. La CIE L*u*v ET CIE L*a*b espaces couleur sont à la fois sur le plan perceptuel, qui fournissent des systèmes uniformes faciles à utiliser pour la comparaison des mesures de similarité.

Après la sélection d'un espace couleur, un descripteur de couleur efficaces devraient être mis en place pour représenter la couleur de la zones en générale ou régionales. Plusieurs descripteurs de couleur ont été élaboré à partir de divers régimes de représentation, tels que la couleur des histogrammes, les moments de la couleur, contour de la couleur, texture de la couleur, et la corrélogrammes.

b. Histogramme de couleur

La méthode la plus couramment utilisée pour représenter la couleur caractéristique d'une image est l'histogramme de couleur. Un histogramme de couleur est un type de graphique à barres, où la hauteur de chaque barre représente un montant de couleur particulière de l'espace de couleur utilisé dans l'image. Les barres d'un histogramme de couleur sont nommées comme bins et ils représentent l'axe des x. Le nombre de bacs dépend du nombre de couleurs il y a dans une image. Le nombre de pixels dans chaque bac, indiqué sur l'axe des y, montre combien de pixels dans une image sont d'une couleur particulière. L'histogramme des couleurs peut non seulement facilement caractériser le global et la distribution régionale des couleurs d'une image, mais aussi être invariante à la rotation autour de l'axe de vue.

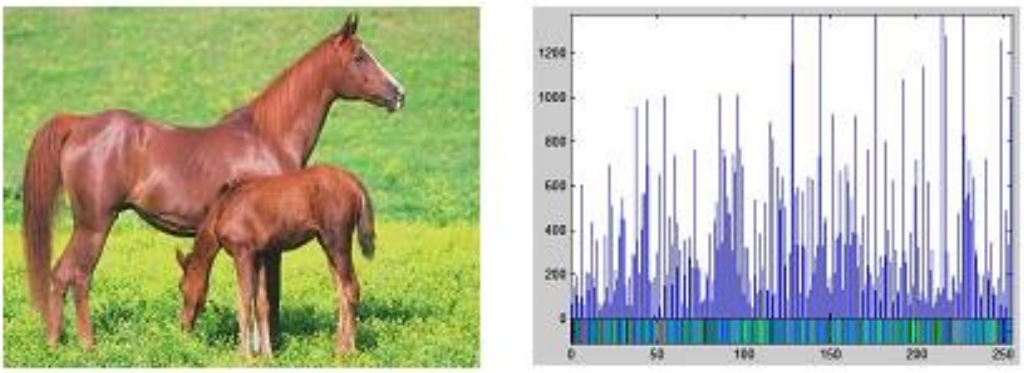

Figure 1.06 : *Une image et l'histogramme de couleur correspondante*

Dans les histogrammes en couleur, la quantification est un processus où le nombre de bacs est réduit en prenant les couleurs qui sont semblables les uns aux autres et les placer dans le même bac. La quantification réduit l'espace requis pour stocker l'information de l'histogramme et le temps pour comparer les histogrammes. De toute évidence, la quantification réduit les informations concernant le contenu d'images; c'est le compromis entre l'espace, le temps de traitement et d'exactitude des résultats.

Les histogrammes couleur sont classés en deux types, histogramme des couleurs global (GCH : Global Color Histogram) et histogramme de couleur local (LCH : Local Color Histogram). Un GCH prend l'histogramme des couleurs d'image entière et représente donc les informations concernant l'ensemble de l'image, sans prendre en compte la répartition des couleurs des régions dans l'image. Au contraire, une LCH divise une image en des blocs fixes ou régions, et prend l'histogramme de couleur de chacun de ces blocs. LCH contient plus d'informations sur une image, mais en comparant les images, il est plus cher du point de vue de calculs. GCH est connu comme une méthode traditionnelle pour la récupération d'images basées sur couleur. Puisqu'il ne comprend pas la distribution des couleurs des régions, lorsque deux GCHs sont comparés, on ne peut pas toujours obtenir un résultat approprié dans la perspective de similitude des images.

c. La texture

Dans le domaine de la vision par ordinateur il n'y a pas de définition claire du terme texture. La définition de texture est basée sur la méthode d'analyse de la texture et des descripteurs extrait de l'image. Toutefois, la texture peut être définie comme la répétition d'un motif dans le domaine spatiale. Des exemples de différentes textures perceptibles à l'œil humaines sont montrés sur la figure suivante.

Figure 1.07 : *Exemple de texture*

Dans le monde réel, la perception de texture peut être beaucoup plus complexe. Les diverses intensités de luminosité donnent lieu à un mélange différents perception humaine de la texture. La texture des images ont des applications utiles dans le traitement de l'image et la vision informatique.

Ils comprennent : la reconnaissance de l'image à l'aide de propriétés de texture de régions connues comme la classification de texture, reconnaissance de la texture à l'aide de propriétés de texture de bordures appelées segmentation de texture, synthèse de texture, et la génération de la texture d'une image à partir de modèles de texture connus.

Puisqu'il n'y a pas de définition mathématique pour la texture, de nombreuses méthodes différentes pour le calcul de la texture caractéristiques ont été proposées au fil des ans. Malheureusement, il n'y a toujours pas de méthode unique qui fonctionne le mieux avec tous les types de textures. Selon Manjunath Et ma, les méthodes communément utilisées pour la texture caractéristique description sont, statistiques basé sur un modèle, et des méthodes basées sur la transformation. Les catégories de descripteur par texture sont expliquées ci-dessous.

• Méthodes statistiques [5][10]

Les méthodes statistiques analysent la distribution spatiale des valeurs de gris par le calcul de descripteur local à chaque point dans l'image. Ils utilisent la matrice de Co-occurrence, les moments statistiques, la représentation les différences de niveau de gris, fonction d'autocorrélation, niveau de gris et des tirages.

La méthode statistique la plus utilisée est le GLCM (Gray-level Co-occurrence Matrix). Il s'agit d'une matrice à deux dimensions de probabilités conjointes P d,r(i, j) entre paires de pixels, séparés par une distance d, dans une direction r donnée. Il est populaire dans la description de textures et est basée sur la répétition de l'occurrence de quelque niveau de gris dans la configuration de la texture; cette configuration varie rapidement avec la distance dans le cas de texture fine et lentement dans les grosses textures. Haralick définit 14 caractéristiques statistiques de la matrice de cooccurrence en niveau de gris pour une texture de niveau de classification, tels que l'énergie, de l'entropie, contraste, probabilité maximale, l'autocorrélation, et différence inverse moment.

La matrice de cooccurrence en niveau de gris pour la représentation de la caractéristique des textures est très utilisée dans des applications en reconnaissant des défauts de tissu.

d. Approches basées sur le modèle

La méthode de texture basée sur modèle essaye de capturer le processus qui a généré la texture. En utilisant cette approche, une modèle d'image est utilisé et un algorithme défini les paramètres du modèle pour donner le meilleur ajustement.

e. Descripteur dans un domaine transformé

Le mot transformer renvoie à une représentation mathématique d'une image. Il y a plusieurs classifications de texture à l'aide de fonctionnalités de domaine de transformation dans le passé, tels que la transformée de Fourier discrète, discret, ondelettes et ondelettes de Gabor.

Figure 1.08 : *Décomposition d'une image en ondelette*

La méthode avec transformation analyse le contenu fréquentielle de l'image pour déterminer le descripteur de texture. L'analyse de Fourier consiste à décomposer un signal en ondes sinusoïdales de fréquences différentes. D'autre part, de l'analyse ondelettes découpe un signal en décalé et en versions redimensionnées de d'ondelette l'original.

Les moments des coefficients d'ondelettes dans différentes bandes de fréquences ont été prouvé très efficace pour la représentation de la texture. Manjunath et Ma ont montré que la récupération d'image à l'aide de Gabor dépasse les autres fonctions de description.

f. Forme

L'une des fonctionnalités couramment utilisées dans les systèmes CBIR est la forme. La forme d'un objet est la configuration de surface caractéristique représentée par le contour. La perception humaine de l'environnement se fait par la reconnaissance de forme. Cela est important pour le CBIR parce que ça correspond à la région d'intérêt dans les images (ROI). Dans les techniques basées sur l'approche région, tous les pixels qui correspond à une ROI est prise en compte pour obtenir la représentation de la forme

En comparaisons avec la technique basée sur région, la technique basée sur contours plus populaire. Un descripteur de basé sur le contour inclus la surface, le périmètre, l'orientation, l'élongation. Les descripteurs plus complexes peuvent contenir des descripteurs de Fourier, des chaines de code. [8][9][10][11]

1.3.2.3 Mesure de similitude

La similitude entre deux images est définie par une mesure de similarité. Le choix de la métrique de de similarité a un impact direct sur les performances de la récupération d'image basée sur le contenu. Le type de descripteur sélectionné détermine la forme des vecteurs de mesure qui sera utilisée pour comparer leur similitude. Si les caractéristiques extraites à partir des images sont présentées comme points multidimensionnelle, les distances entre points multidimensionnel correspondant peuvent être calculées. La distance euclidienne est la mesure la plus couramment utilisée pour mesurer la distance entre deux points dans l'espace multidimensionnel.

Pour d'autres sortes de caractéristiques telles que la couleur histogramme, la distance euclidienne n'est pas un métrique idéal ou peuvent ne pas correspondre à la perception l'homme. L'histogramme intersection a été proposé par Swain et Bllard trouve des objets connus dans les images à l'aide d'histogrammes de couleur. Un certain nombre d'autres paramètres, tels que la distance Mahalanobis, la distance de Minkowski, la distance Earth Mover, ont été proposées à des fins spécifiques. [2][4][10][11]

1.3.2.4 Structures de l'indexation

Lors de la manipulation des bases de données massive, un bon référencement est une nécessité. Le traitement toutes les articles dans une base de données lors de l'exécution de requêtes est extrêmement lente et inefficace. Lorsqu'on travaille avec des documents, la création d'un bon index n'est pas très difficile. On utilise tout simplement une liste de tous les mots dans la base de données, et l'information contenue dans le document. Lors de la recherche d'une expression, le système vérifie d'abord l'index pour les documents qui contiennent tous les mots de la recherche. Ensuite, le traitement en profondeur ne doit être effectué avec ces documents.

Toutefois, lorsque vous recherchez des images, cette approche est beaucoup plus difficile. Les données sont non index able comme tel, alors des vecteurs de descripteurs doivent être utilisés comme base de l'Index. La méthode populaires d'indexation multidimensionnelle incluent des algorithmes R-tree et la R*-tree.

Le Self Organizing Map (SOM) ou carte auto adaptative est également une structure d'indexation. Le SOM est entrainé pour faire correspondre la forme des données à l'espace descripteur. Après l'apprentissage, le plus proche nœud dans la SOM est calculé pour chaque image dans la base de données. Cette information à propos des nœuds les plus proches sont stockées. Lors d'une recherche,

la première chose à faire est de calculer le plus proche de nœud SOM, également connu comme la meilleure unité de correspondance au vecteur descripteur de l'image requête. Après cela, nous savons quelles images dans la base de données sont les plus proches à l'image de la requête : ceux qui correspondent au même nœud comme image de la requête. [12]

1.4 Evaluation des méthodes existante

Les méthodes d'indexation textuelles ont le grand avantage de représenter le contenue d'une image par des descripteurs de haut niveau contenue dans le langage humain. Le tag des images par des mots facilite la recherche et permet d'utilisation de concept sémantique. Le problème de cette méthode réside dans l'automatisation de la tache parce que la description d'une base de données entière est impossible pour des opérateurs humains. Cette technique a été abandonnée dans la littérature en faveur des descripteurs visuels. [13] [14] [15]

L'indexation et la recherche d'image basé sur le contenue est très efficace lorsqu'il est question de rechercher des images contenant des caractéristiques particulière. Cependant leur limite se situe au niveau des descripteurs même. L'utilisation d'histogramme par exemple pose des inconvénients : le manque d'information spatiale, la sensibilité à des changements de luminosité. L'utilisation de texture n'est pas aussi très efficace et sont très lent à calculer.

1.5 Approche proposée

L'indexation textuelle présente des avantages très utiles pour la recherche d'image. L'abandon de cette technique est causé par le manque de technologie nécessaire pour annoter une image par son contenue. Le problème d'indexation peut aujourd'hui être résolu par les systèmes à apprentissage automatique. Les avancés atteint aujourd'hui dans le domaine de la vision par ordinateur permet même de dépasser la capacité visuel humain. [9]

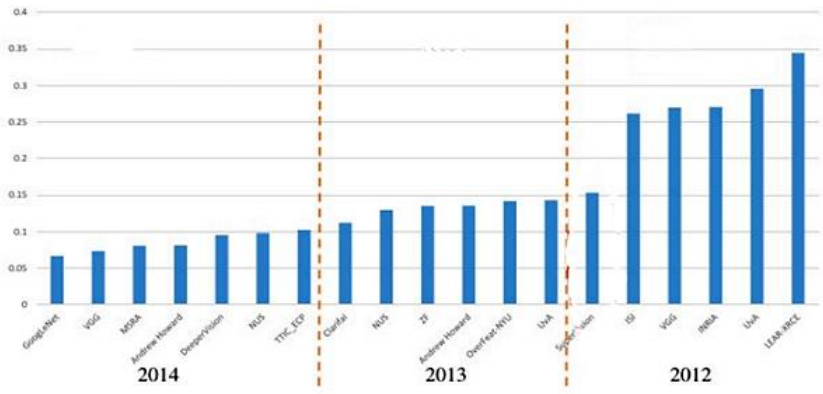

Figure 1.09 : *Réduction de l'erreur de classement ces dernières années*

En terme de classification, plusieurs méthodes sont disponible, on peut en citer : La mixture gaussien, le réseau de neurone, le SVM (Support Vector Machine). Toutefois en termes d'implémentation et de précision, les réseaux de neurone artificiel sont inégalés. Il a des nombres élevés de Framework complet et très performant sur ces réseaux, favorisant son implémentation. Comme le montre la figure suivante, les premières places des classifications sur la prédiction de la classe d'une image sont tous détenues par des réseaux de neurone artificiel. [9][13]

Concernant le choix du réseau de neurone artificiel à utiliser, on peut utiliser :

- Un réseau DBM ou Deep Boltzman Machine : il a pour avantage d'offrir des niveaux d'abstraction très élevés. Cependant son architecture n'est pas adaptée pour traiter des images.
- Un réseau de neurone à convolution ou CNN (Convolutional Neural Network) : présente des grandes performances lorsqu'il est question de traitement d'image.

Team	Year	Place	Error (top-5)	Uses external data
SuperVision	2012	1st	16.4%	no
SuperVision	2012	1st	15.3%	Imagenet 22k
Clarifai	2013	1st	11.7%	no
Clarifai	2013	1st	11.2%	Imagenet 22k
MSRA	2014	3rd	7.35%	no
VGG	2014	2nd	7.32%	no
GoogLeNet	2014	1st	6.67%	no

Tableau 1.01: *Classification **ILSVRC 2014***

La solution retenue est l'utilisation des CNN, à cause de sa popularité et son efficacité, par exemple sur le tableau 1.01, le VGG (Visual Geometry Group) et le GoogLeNet sont des réseaux de neurones à convolution. L'approche que nous proposons consiste donc à utiliser un réseau de neurone à convolution pour l'indexation textuelle d'image. Cela permettra de combler le problème d'indexation automatique de cette technique. De ce fait il nous sera possible d'appliquer l'indexation textuelle à grande échelle ainsi que favoriser l'utilisation de la recherche sémantique d'image [16]

1.6 Conclusion

L'indexation et la recherche d'image se présente sous deux formes : la première est l'utilisation de l'indexation textuelle d'image, la seconde est l'indexation et la recherche par des descripteurs visuel. La première méthode a été abandonnée par les chercheurs du fait qu'il n'y avait pas encore de technique assez efficace pour la description textuelle automatique de l'image. La deuxième méthode présente des inconvénients sur les descripteurs utilisés, cela limite leurs utilisations.

L'approche que proposer est d'améliorer l'indexation textuelle en utilisant un réseau de neurone à convolution dans le processus de description de l'image. Cela permettra de faire des recherches sémantiques sur une base de données d'image.

CHAPITRE 2
LES RESEAUX DE NEURONES

2.1 Introduction

Depuis son invention, les ordinateurs n'ont cessé de monter en puissance et en capacité de calcul. Toutefois, cette augmentation de puissance ne permet pas toujours de résoudre les problèmes d'une application informatique. L'idée est lancé que cette lacune n'était pas matériel mais logiciel.

La construction de logiciels s'appuie sur plusieurs approches. Deux parmi les plus utilisées sont l'approche algorithmique et l'approche basée sur la connaissance.

Une approche algorithmique nécessite l'écriture du processus à suivre pour résoudre le problème. Lorsque le problème est complexe, cela peut être une étape coûteuse ou impossible. D'autre part, les ordinateurs sont des machines complètement logiques qui suivent à la lettre chacune des instructions du programme. C'est un avantage lorsque tous les cas ont été prévus à l'avance par le programmeur. Ce n'est hélas pas toujours possible.

La seconde approche possible est celle de l'intelligence artificielle avec pour applications les plus connues les systèmes experts. Ici, la résolution du problème est confiée à un ensemble de règles données par l'expert humain du domaine. Il n'en demeure pas moins que toutes les règles doivent avoir été exprimées préalablement au traitement, et que le programme demeure binaire dans son exécution. Les cas qui n'ont pas été prévus par l'expert ne seront pas correctement traités. L'introduction de la logique floue ne change pas la nature des limitations d'emploi du programme : l'exécution reste totalement déterministe. En fait, l'approche basée sur la connaissance se limite à des domaines d'application où la modélisation de la connaissance, par exemple sous forme de règles, est possible. Ces domaines sont souvent ceux des sciences dites "exactes" comme l'électronique, la mécanique ou la physique, par opposition aux sciences dites "humaines" comme la médecine, la psychologie, la philosophie où la connaissance est plus empirique.

Ces deux approches ne suffisent pas à répondre à tous les problèmes existants tels que dans les domaines de la reconnaissance de forme, du contrôle moteur, de la traduction automatique, de la compréhension du langage, depuis longtemps explorés à l'aide des approches algorithmiques et à base de connaissances, qui n'ont pas rencontré le succès escompté. Pourtant, des êtres vivants relativement simples sont capables de réaliser certaines de ces opérations apparemment sans difficulté. Il suffit pour s'en rendre compte de lever les yeux, suivre le vol de la mouche et essayer

de la capturer. Une troisième approche au traitement automatique de l'information semble donc s'offrir à nous, où l'on cherche à s'inspirer du traitement de l'information effectué par le cerveau. De là, l'intérêt pour les réseaux de neurones s'est accrue.

Les réseaux de neurones sont des constructions abstraites s l'activité d'un réseau de neurones biologique simplifié. Le domaine des réseaux de neurones n'est pas nouveau car il a son origine dans des travaux conduits durant les années 40 (modèle de Hebb pour l'évolution des connexions synaptiques). Ces travaux conduisirent au modèle du perceptron dans les années 60 (modèle qui a principalement été appliqué à la reconnaissance de caractères). Mais ce n'est qu'à partir de 1986 que la recherche dans ce domaine a connu une expansion importante du fait de la publication de modèles de réseaux et d'algorithmes d'apprentissage suffisamment efficaces pour résoudre des problèmes réalistes et complexes.

Dans ce chapitre on va introduire le concept de réseau de neurone. On commence par un historique sur des dates symboliques dans ce domaine. Ensuite on va étudier les réseaux de neurone biologique et on termine l'agencement des neurones pour former un réseau de neurone artificiel.

2.2 Historique

W. James (1890, psychologue américain) : Introduit le concept de mémoire associative et propose une loi de fonctionnement pour l'apprentissage des réseaux de neurones connue plus tard sous le nom de "loi de Hebb".

J. Mc Culloch et W. Pitts (1943) : Laissent leurs noms à une modélisation du neurone biologique (un neurone au comportement binaire). Ils sont les premiers à montrer que des réseaux de neurones simples peuvent réaliser des fonctions logiques, arithmétiques et symboliques.

D. Hebb (1949, physiologiste américain) : Explique le conditionnement chez l'animal par les propriétés des neurones eux-mêmes (ex. : le fameux comportement pavlovien).Il propose aussi une loi de modification des propriétés des connexions entre neurones.

F. Rosenblatt (1957) : Développe le modèle du Perceptron. Il construit le premier neuro-ordinateur basé sur ce modèle et l'applique au domaine de la reconnaissance des formes (prouesse technologique pour l'époque). Il propose aussi le premier algorithme d'apprentissage, qui permet d'ajuster les paramètres d'un neurone.

B. Widrow (1960, automaticien) : Développe le modèle Adaline (Adaptive Linear Element), proche du modèle du Perceptron mais dont la loi d'apprentissage est différente. Celle-ci est à l'origine de

l’algorithme de rétro-propagation du gradient très utilisé pour l’apprentissage des Perceptrons multicouches.

M. Minsky et S. Papert (1969) : Publient un livre qui met en évidence les limitations du Perceptron, notamment l’impossibilité de résoudre des problèmes non linéaires. Après leur ouvrage, les financements des recherches sur les réseaux de neurones artificiels s’arrêtent et les chercheurs se tournent vers l’intelligence artificielle et les systèmes à base de règles.

1967-1982 : Période creuse pour les réseaux de neurones artificiels. Les recherches s’orientent sur des domaines connexes tels que le traitement adaptatif du signal, la neurobiologie, la reconnaissance de formes, etc.

J. J. Hopfield (1982, physicien) : Profite d’une certaine désillusion de l’intelligence artificielle (heurtée à de sérieuses limitations) et présente une théorie du fonctionnement et des possibilités des réseaux de neurones. Il explique notamment dans un ouvrage la structure et loi d’apprentissage d’un réseau de neurones correspondant à un résultat escompté. Ce modèle est encore très utilisé aujourd’hui pour des problèmes d’optimisation.

1983 : La machine de Boltzmann est le premier modèle connu capable de traiter de manière satisfaisante les limitations recensées du Perceptron. Mais l’utilisation pratique s’avère difficile, les temps de convergence étant considérables.

En 1986, « Rumelhart, Hinton et Williams » publient l’algorithme de la ‘rétro propagation de l’erreur’ qui permet d’optimiser les paramètres d’un réseau de neurones à plusieurs couches. À partir de ce moment, la recherche sur les réseaux de neurones connaît un essor fulgurant et, au cours des années 90, les applications commerciales de ce succès académique suivent. [17]

2.3 Neurone biologique

En biologie, un neurone est une cellule nerveuse dont la fonction est de transmettre un signal électrique dans certaines conditions. Il agit comme un relai entre une couche de neurones et celle qui la suit. Les caractéristiques des neurones sont encore mal connues (et font l'objet de recherches) mais on connait leur principe d'action.

2.3.1 Structure

Le neurone est une cellule composée d’un corps cellulaire et d’un noyau. Le corps cellulaire se ramifie pour former ce que l’on nomme les dendrites. Celles-ci sont parfois si nombreuses que l’on

parle alors d'arborisation dendritique. C'est par les dendrites que l'information est acheminée de l'extérieur vers le soma, corps du neurone. L'information traitée par le neurone chemine ensuite le long de l'axone (unique) pour être transmise aux autres neurones. La transmission entre deux neurones n'est pas directe. En fait, il existe un espace intercellulaire de quelques dizaines d'Angstroms (10-9 m) entre l'axone du neurone afférent et les dendrites (on dit *une* dendrite) du neurone efférent. La jonction entre deux neurones est appelée la synapse. [17] [18]

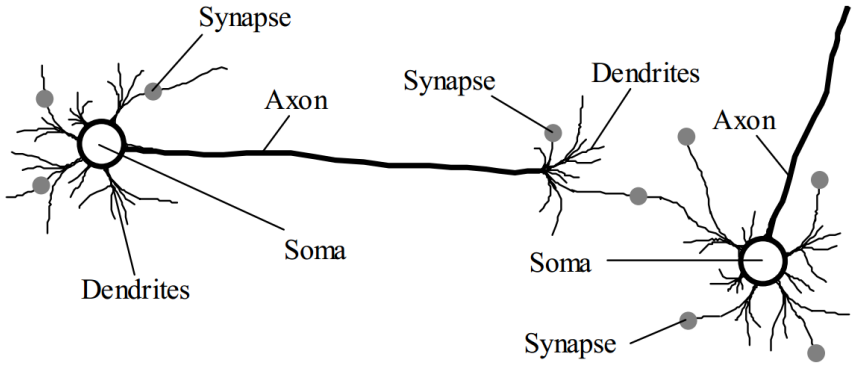

Figure 2.01 : *Schéma d'un neurone biologique*

Selon le type du neurone, la longueur de l'axone peut varier de quelques microns à 1,50 mètre pour un motoneurone. De même les dendrites mesurent de quelques microns à 1,50 mètre pour un neurone sensoriel de la moelle épinière. Le nombre de synapses par neurone varie aussi considérablement de plusieurs centaines à une dizaine de milliers.

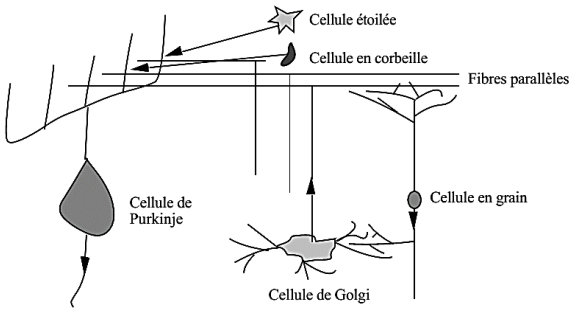

Figure 2.02 : *Différents type de neurone*

2.3.2 Potentiel au repos

La membrane du neurone est une membrane dialysant d'une épaisseur de l'ordre de 0 : 01 ¹m Au repos, il existe une différence de potentiel de -70mV entre l'intérieur et l'extérieur du neurone (potentiel de repos). Son existence est liée aux différences de concentrations ioniques de part et

d’autre de la membrane. A l’extérieur de la cellule, on trouve une forte concentration en sodium (N a +) et en chlore (Cl). A l’intérieur, on trouve une importante concentration en potassium (K +) et en anions organiques. Les principaux mouvements ioniques mis en jeu dans le fonctionnement du neurone sont ceux du sodium et du potassium.

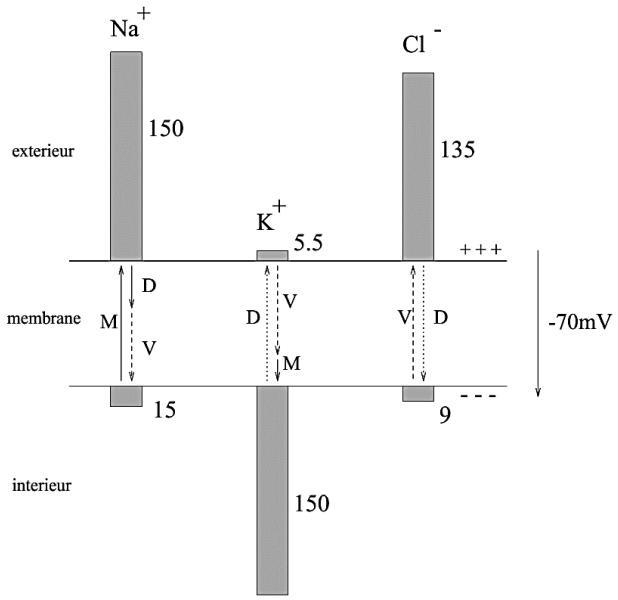

Figure 2.03 : *Différents concentration au sein d’un neurone*

PNa = Permeablité au sodium

PK = Permeabilité au potassium

La perméabilité de la membrane à ces ions dépend du potentiel. Du point de vue macroscopique, on peut distinguer trois flux correspondant à trois causes différentes:

- La diffusion, qui tend à égaliser les concentrations (elle dépend du gradient de concentration et de la perméabilité de la membrane).
- La force électrostatique, due à la différence de potentiel
- La pompe métabolique.

Au repos, ces trois flux s’équilibrent. La pompe métabolique est un mécanisme actif, consommateur d’énergie. En son absence, les concentrations tendraient à s’égaliser par le jeu de la diffusion [17] [18]

2.3.3 Potentiel d'action

Lorsque, à l'aide d'électro des, on augmente artificiellement le potentiel de la membrane, il ne se passe rien jusqu'à une valeur critique de l'ordre de -40mV. Au-dessus de cette valeur, on observe une inversion brutale du potentiel, puis un rapide retour à l'état de repos, dans un temps de l'ordre de 3.5ms.

Cette inversion temporaire du potentiel, appelée potentiel d'action, a pour origine une avalanche de phénomènes électro chimiques En effet, la perméabilité de la membrane dépend fortement du potentiel.

Lorsque le potentiel dépasse -40mV, la perméabilité aux ions Na^+ augmente rapidement, d'où une entrée massive de ces ions dans la cellule. Ceci entraîne une augmentation du potentiel, d'où une perméabilité encore plus importante au sodium, etc.

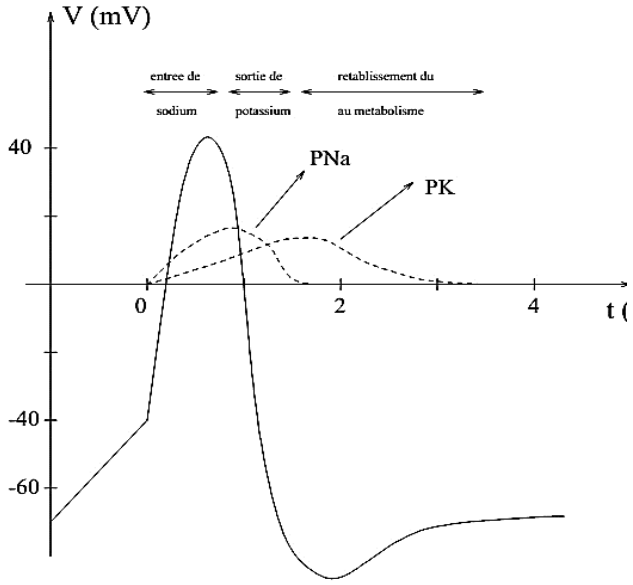

Figure 2.04 : *Potentiel lors de l'activation d'un neurone*

PNa = Permeablité au sodium

PK = Permeabilité au potassium

L'inversion du potentiel résultant de cette réaction auto catalytique provoque une augmentation de la perméabilité aux ions K^+. Il semblerait en effet que la perméabilité aux ions K^+ augmente plus lentement, et avec un certain retard, par rapport à la perméabilité aux ions Na^+. Ceci induit une

sortie massive d'ions potassium, d'où une chute du potentiel. Le potentiel de repos (-70mV) est ensuite rapidement rétabli par la pompe métabolique. [17] [18]

2.4 Du neurone biologique vers le neurone artificiel : le neurone formel

Les neurones formels sont dotés de caractéristique inspirée de celle des neurones biologiques vue précédemment :

- Le potentiel d'action : il s'agit ici d'une valeur numérique qui peut être transmise à des neurones en aval. Un neurone formel, ne peut transmette qu'une valeur unique qui correspond à son état d'activation.
- Les dendrites des neurones biologiques leur permettent de recevoir différents signaux de l'extérieur. De la même manière, un neurone formel des signaux de plusieurs neurones.
- Les synapses : les poids pondèrent les signaux émis par les différents neurones situé en amont ; On retrouve ici l'analogie des synapses qui peuvent être inhibitrice ou excitatrices.
- Seuil d'activation : une fonction d'activation gère l'état du neurone formel. [17] [18]

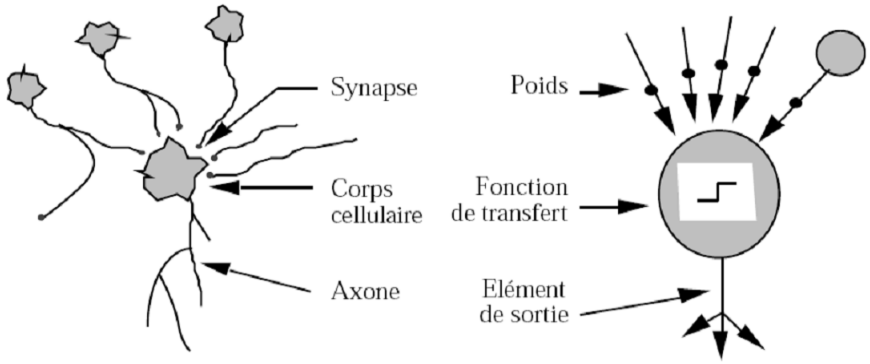

Figure 2.05 : *Mise en correspondance d'un neurone biologique et artificiel*

2.4.2 Formulation mathématique

Ce neurone formel est un processeur élémentaire qui réalise une somme pondérée des signaux qui lui parviennent. La valeur de cette sommation est comparée à un seuil et la sortie du neurone est une fonction non linéaire du résultat:

$$u = \sum_{j=1}^{d} w_j \, x_j \qquad (2.01)$$

$$y = \varphi(u) \tag{2.02}$$

En notant par $X = [-1\, x_1\, x_2 \ldots\ldots\ldots\ldots x_d]^T$ le vecteur d'entrée augmenté, et par $W = [w_1\, w_2 \ldots\ldots\ldots\ldots w_d]^T$ le vecteur de poids augmenté :

$$y = \varphi(W^T X) \tag{2.03}$$

2.4.3 Fonction d'activation

L'activation d'un neurone est donnée par la somme des potentiels de sortie de ses prédécesseurs, pondérée par les poids synaptiques. Ce potentiel d'activation est ensuite transformé par une fonction afin de déterminer le potentiel de sortie.

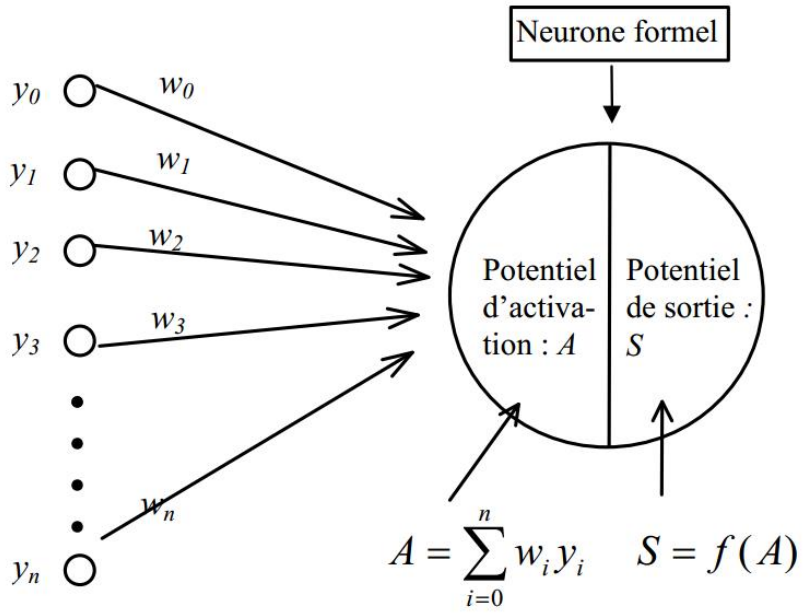

Figure 2.06 : *Modélisation de la fonction d'activation*

En principe, toute fonction choisie croissante et paire peut être utilisée, mais le plus souvent on fait appel à des fonctions ramenant le résultat à l'intérieur de bornes prédéfinies. Dans sa première version, le neurone formel était donc implémenté avec une fonction à seuil. Mais plusieurs implémentations ont été étudiées par la suite :

2.4.3.2 Fonction binaire à seuil

$$H(x) = \begin{cases} 1 \; si \; x \geq 0 \\ 0 \; sinon \end{cases} \& \tag{2.04}$$

$$Sgn(x) = \begin{cases} 1 \; si \; x \geq 0 \\ -1 \; sinon \end{cases} \tag{2.05}$$

Le seuil introduit une non-linéarité dans le comportement du neurone, cependant il limite la gamme des réponses possibles à deux valeurs. [17] [18][19]

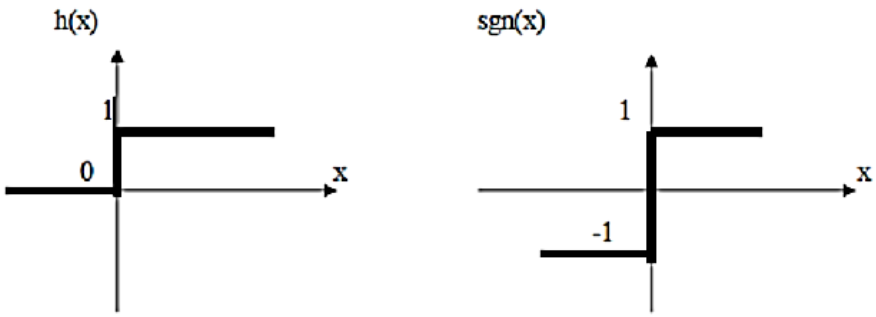

Figure 2.07 : *Fonction binaire à seuil*

2.4.3.3 Fonction linéaire

C'est l'une des fonctions d'activations les plus simples, sa fonction est définie par : F(x)=x

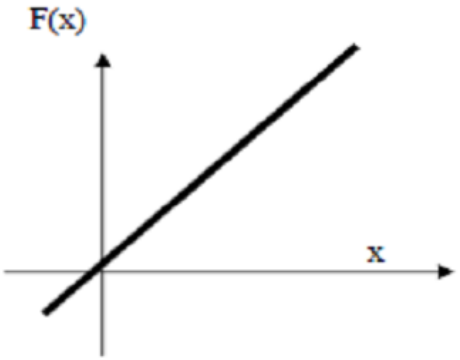

Figure 2.08 : *Fonction linéaire*

Fonction linéaire à seuil ou multi seuil Cette fonction représente un compromis entre la fonction linéaire et la fonction seuil, entre ses deux barres de saturation, elle confère au neurone une gamme

de réponses possibles. En modulant la pente de la linéarité, on affecte la plage de réponse du neurone.

$$F(x) = \begin{cases} x & x \in [u, v] \\ l & si\ x \geq v \\ m & si\ x \leq u \end{cases} \tag{2.06}$$

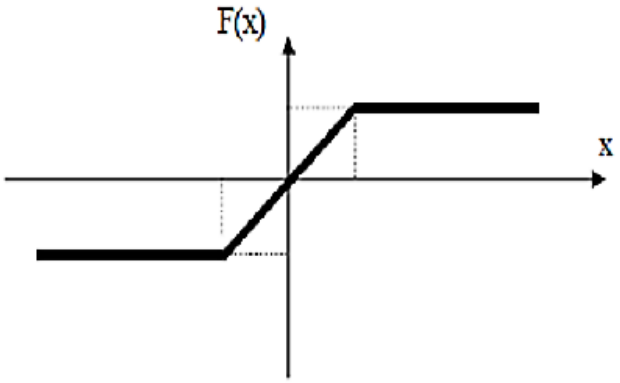

Figure 2.09 : *Fonction linéaire à seuil*

2.4.3.4 La fonction sigmoïde

Elle est l'équivalent continu de la fonction linéaire. Etant continu, elle est dérivable, d'autant plus que sa dérivée est simple à calculer, elle est définie par :

$$\varphi(x) = \frac{1}{1 + e^{-x}} \tag{2.07}$$

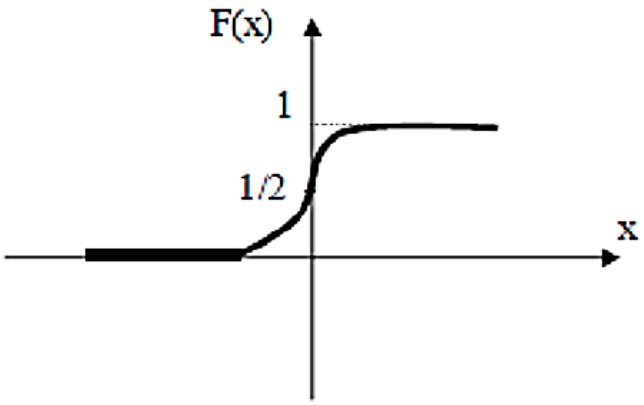

Figure 2.10 : *Fonction sigmoïde*

2.5 Réseau de neurone artificiel

Un réseau de neurone artificiel est un agencement de neurone suivant une architecture particulière. Les neurones qui le composent peuvent être de simple neurone formel comme le cas du perceptron multicouche, peuvent être aussi complexe et disposer de plusieurs entrée et sortie comme celui d'une cellule de LSTM (Long Short Term Memory) et peuvent même être composé de plusieurs neurones. L'architecture du réseau est arbitraire et est plus dicté par l'utilisation de ce dernier. [18] [19] [20]

2.5.1 Structure d'interconnexion

Les connexions entre les neurones qui composent le réseau décrivent la topologie du modèle. Elle peut être quelconque, mais le plus souvent il est possible de distinguer une certaine régularité.

Réseau multicouche : les neurones sont arrangés par couche. Il n'y a pas de connexion entre neurones d'une même couche et les connexions ne se font qu'avec les neurones des couches en aval. Habituellement, chaque neurone d'une couche est connecté à tous les neurones de la couche suivante et celle-ci seulement. Ceci nous permet d'introduire la notion de sens de parcours de l'information (de l'activation) au sein d'un réseau et donc définir les concepts de neurone d'entrée, neurone de sortie. Par extension, on appelle couche d'entrée l'ensemble des neurones d'entrée, couche de sortie l'ensemble des neurones de sortie. Les couches intermédiaires n'ayant aucun contact avec l'extérieur sont appelés couches cachées.

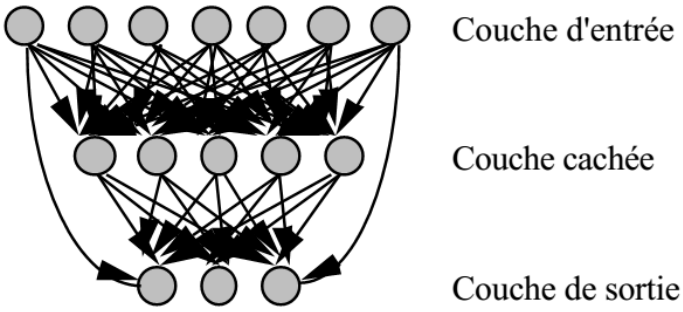

Figure 2.11 : *Représentation d'un réseau multicouche*

Réseau à connexions locales : Il s'agit d'une structure multicouche, mais qui à l'image de la rétine, conserve une certaine topologie. Chaque neurone entretien des relations avec un nombre réduit et localisé de neurones de la couche en aval. Les connexions sont donc moins nombreuses que dans le cas d'un réseau multicouche classique.

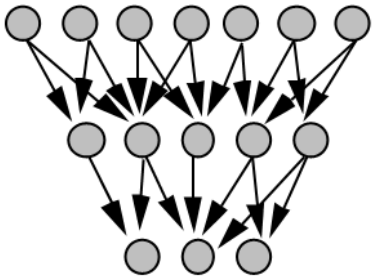

Figure 2.12 : *Réseau de neurone artificiel à connexions locales*

Réseau à connexions récurrentes : les connexions récurrentes ramènent l'information en arrière par rapport au sens de propagation défini dans un réseau multicouche. Ces connexions sont le plus souvent locales

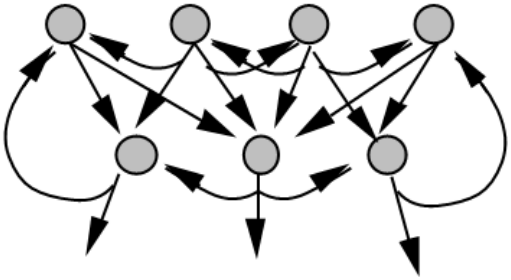

Figure 2.13 : *Représentation d'un réseau récurent*

Réseau à connexion complète : c'est la structure d'interconnexion la plus générale. Chaque neurone est connecté à tous les neurones du réseau

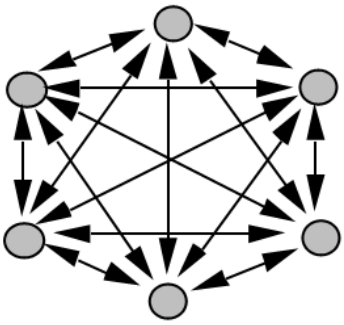

Figure 2.14 : *Réseau à connexion complète*

2.5.2 Formulation mathématique

Un réseau de neurones agit sur des données. Ces donnée s sont définies formellement comme étant un ensemble de couples (x, v) avec x ∈ Rn et y ∈ Rm, n, m ∈ N ∗. En outre, un réseau de neurones

peut être défini comme étant une fonction f : Rn → Rm qui prend en entrée les informations du vecteur x et qui effectue une prédiction y ∈ Rm sur le vecteur c.

Le réseau de neurone est disposé en r couches. La couche 0 représente l'entrée et possède n neurones, soit la dimension du vecteur d'entrée x. De même, la r-ième couche, la couche de sortie, possède m neurones, soit la dimension du vecteur de sortie y. Les couches intermédiaires, nommées couches cachées, possèdent un nombre de neurones arbitraires, souvent déterminé par la complexité du problème à résoudre.

Les neurones d'un réseau de neurones représentent les unités de calcul du modèle. Chaque neurone possède une entrée (un scalaire), une fonction d'activation et une sortie. Par exemple, le k-ième neurone de la couche i reçoit une valeur d'entrée

$$e_k^{(i)} = \sum_j w_{j,k}^{(i-1)} s_j^{(i-1)} + b_k^{(i)} \tag{2.08}$$

et retourne la valeur $s_k^{(i)} = a_k^{(i)}(e_k^{(i)})$ qui est la fonction d'activation appliquée à l'entrée du neurone.

La valeur $b_k^{(i-1)}$ est un biais ajouté à l'entrée du k-ième neurone de la couche i servant, en quelque sorte, de seuil déterminant à partir de quelle valeur le neurone est significativement activé. En d'autres termes, le biais dicte à partir de quelle valeur la s om me des produits des poids et des sorties de la couche précédente passe le seuil où un neurone est considéré actif.

Finalement, il y a la fonction d'erreur E qui dénote l'erreur du réseau de neurones dans son approximation de la sortie y par rapport à la cible c. C'est cette fonction qui détermine la qualité du modèle de comportement des données.

2.5.3 Modélisation matricielle du réseau en entier

L'effet d'un neurone à e_1 entrées peut être modélisé par une forme linéaire de R_{e_1} dans R (qui multiplie les composantes de l'entrée par des poids *w* bien choisis), à laquelle on soustrait un scalaire biais et l'on applique enfin une fonction d'activation *f* au résultat.

L'action d'une couche de s_1 neurones est donc assimilable à une application linéaire de R_{e_1} dans R_{s_1}, que l'on représentera ici par une matrice à e_1 colonnes et s_1 lignes : une entrée à e_1 composantes aura pour image un vecteur à s_1 composantes. On soustrait à cette première sortie un biais B_1 qui est cette fois-ci vectoriel, et l'on applique une fonction d'activation f_1 à chaque composante du vecteur obtenu

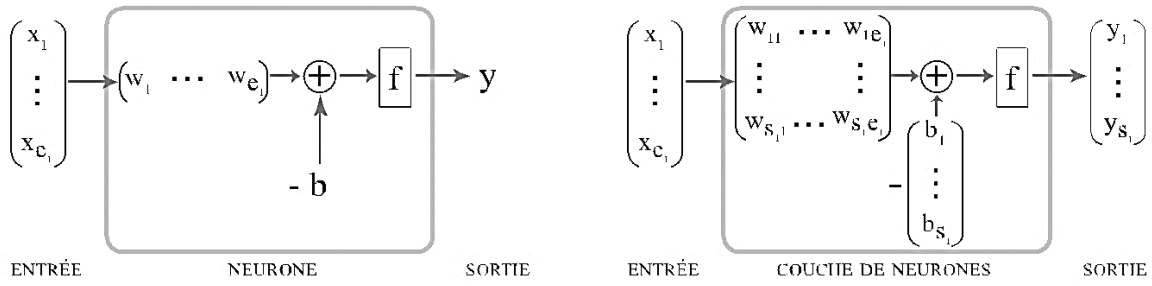

Figure 2.15 : *Modélisation mathématique d'une couche de neurones*

Un réseau de neurones tout entier (*N* couches empilées les unes à la suite des autres) est donc une succession de *N* matrices de tailles $e_i \times s_i (1 \leq i \leq N)$, auxquelles on associe, à chaque fois, un vecteur biais b_i et une fonction d'activation f_i. Par souci de simplicité, on prendra la même fonction d'activation f_i pour tous les neurones d'une même couche. En revanche, utiliser des fonctions différentes pour les différentes couches peut s'avérer assez intéressant. [17]

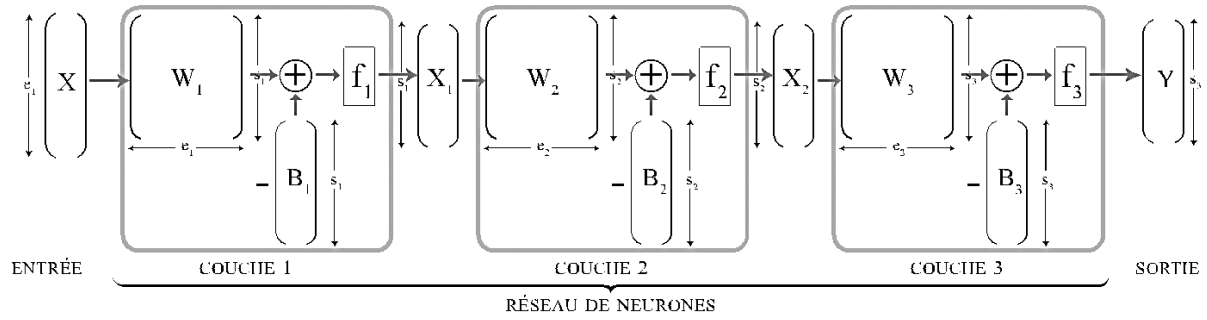

Figure 2.16 : *Modélisation d'un réseau en entier*

2.5.4 Le perceptron Multicouche

Le perceptron de Rosenblatt est l'un des premiers RNA (Reseau de Neurone Artificiel) opérationnels. Pour traiter plusieurs sorties, on utilise plusieurs Perceptrons en parallèle, on parle de réseau monocouche.

L'utilisation d'une fonction d'activation du type linéaire à seuil permet de réaliser pour chaque neurone i une partition des vecteurs d'entrées en deux classes dont la frontière est définie par l'hyperplan de dimension n-1 (où n est le nombre d'entrées) et d'équation :

$$\sum_{j=1}^{d} w_{ij}\, x_j + w_{i0} = 0 \qquad (2.09)$$

Le perceptron multicouche ou MLP (Multi Layer Perceptrons) est un type de réseau de neurones parmi les plus utilisés. Son fonctionnement est le suivant : l'information se propage de couche en couche, suivant le modèle décrit plus haut, toujours dans le même sens jusqu'à la sortie. On donne au réseau l'information sous forme d'un vecteur de R_{e_1} et on récupère l'information traitée sous forme d'un vecteur de R_{s_v}.

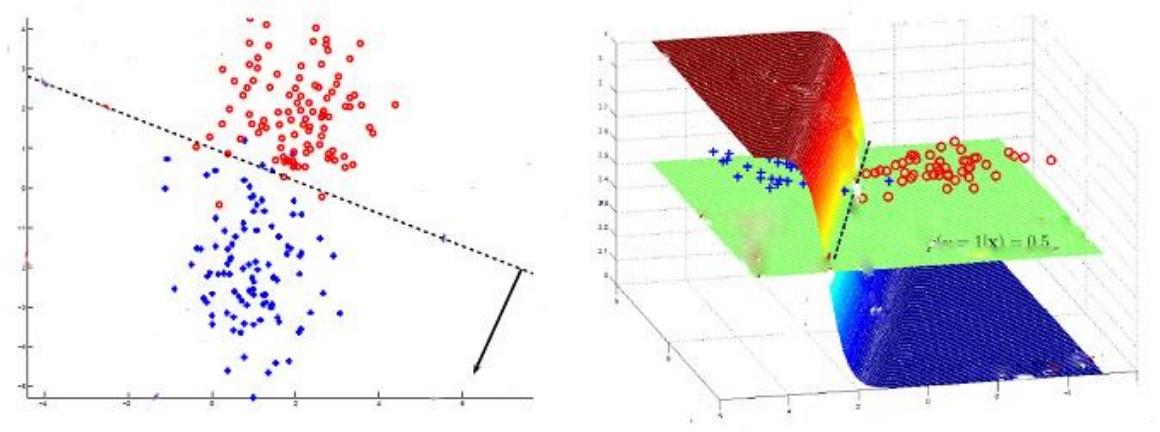

Figure 2.17 : *Séparation Hyperplan effectué par un perceptron multicouche*

2.5.5 Apprentissage

Dans le cas des réseaux de neurones artificiels, on ajoute souvent à la description du modèle l'algorithme d'apprentissage. Le modèle sans apprentissage présente en effet peu d'intérêt. Dans la majorité des algorithmes actuels, les variables modifiées pendant l'apprentissage sont les poids des connexions. L'apprentissage est la modification des poids du réseau dans l'optique d'accorder la réponse du réseau aux exemples et à l'expérience. Il est souvent impossible de décider à priori des valeurs des poids des connexions d'un réseau pour une application donnée. A l'issu de l'apprentissage, les poids sont fixés : c'est alors la phase d'utilisation. Certains modèles de réseaux sont improprement dénommés à apprentissage permanent. Dans ce cas il est vrai que l'apprentissage ne s'arrête jamais, cependant on peut toujours distinguer une phase d'apprentissage (en fait de remise à jour du comportement) et une phase d'utilisation. Cette technique permet de conserver au réseau un comportement adapté malgré les fluctuations dans les données d'entrées. [20]

2.5.5.1 Type d'apprentissage

Au niveau des algorithmes d'apprentissage, il a été défini généralement deux grandes classes selon que l'apprentissage est dit supervisé ou non supervisé, mais il y a aussi l'apprentissage par renforcement qui diffère un peu des deux autres. Cette distinction repose sur la forme des exemples d'apprentissage. [20]

a. Apprentissage non-supervisé

Il n'y pas de connaissances à priori des sorties désirés pour des entrées données. En fait, c'est de l'apprentissage par exploration où l'algorithme d'apprentissage ajuste les poids des liens entre neurones de façon à maximiser la qualité de classification des entrées. Le réseau apprend par lui-même à classer des entrées similaires en trouvant des points communs aux stimuli appliqués

b. Apprentissage supervisé

Cet algorithme d'apprentissage ne peut être utilisé que lorsque les combinaisons d'entrées-sorties désirés sont connues. L'apprentissage est alors facilité et par là, beaucoup plus rapide que pour les deux autres algorithmes puisque l'ajustement des poids est fait directement à partir de l'erreur, soit la différence entre la sortie obtenue par le RNA et la sortie désirée. Si l'apprentissage est efficace, la norme de l'erreur diminue globalement. On l'appelle aussi apprentissage par des exemples.

c. Apprentissage par renforcement :

Dans ce cas, bien que les sorties idéales ne soient pas connues directement, il y a un moyen quelconque de connaître si les sorties du RNA s'approchent ou s'éloignent du but visé. Ainsi, les poids sont ajustés de façons plus ou moins aléatoire et la modification est conservée si l'impact est positif ou rejetée sinon.

2.5.5.2 Optimisation

Au départ, un réseau de neurones avec des poids initialisés aléatoirement donne de pauvres résultats. C'est pourquoi il y a une étape d'apprentissage, c'est-à-dire d'ajustement des poids du réseau afin que ceux-ci puissent donner de meilleures prédictions.

Une façon de voir le problème est d'essayer de minimiser l'erreur de prédiction du modèle. Cette formulation transforme le problème d'apprentissage en un problème d'optimisation pour lequel

plusieurs algorithmes existent. L'optimisation de la fonction d'erreur s'effectue par l'a justement des paramètres de la fonction, c'est-à-dire les poids et les biais du réseau de neurones.

Etudions le cas d'un apprentissage supervisé pour un réseau de neurones typique est le réseau feed-forward ; Ce réseau propage l'entrée du réseau vers les couches suivantes sans jamais revenir en arrière. La fonction d'activation utilisé est le logistique définie par

$$\varphi_j^{(i)}(x) = \frac{1}{1 + e^{(-x)}} \tag{2.10}$$

, sauf pour la dernière couche qui prendra une identité comme fonction ; Cela facilite l'expression du réseau. La valeur de sortie est donc :

$$s_j^{(i)} = \varphi_j^{(i)}\left(e_j^{(i)}\right) = \frac{1}{1 + e^{\left(-e_j^{(i)}\right)}} \text{ pour } i \neq r \tag{2.11}$$

La fonction d'erreur qu'on utilise est la moitié de la distance Euclidienne entre la sortie et la cible.

$$E = \frac{1}{2}\|y - v\|^2 = \frac{1}{2}\sum_{i=1}^{m}(y_i - v_i)^2 \tag{2.12}$$

Pour effectuer la propagation avec un couple (x, c), les sorties de la couche d'entrée sont assignées aux valeurs du vecteur x, c'est-à-dire que s(1)j = xj. Les valeurs sont ensuite propagées au travers du réseau de neurones grâce aux formules (2.08) et (2.09) pour donner les sorties yj = s(jr) qui sont comparées aux cibles cj par la fonction d'erreur E (2.10).

Il faut maintenant minimiser la moyenne des erreurs données par la fonction E sur l'ensemble des données fournies en entrée :

$$E_{moyenne} = \frac{1}{N}\sum_{t=1}^{N} E_t \tag{2.13}$$

Où N est le nombre de couples donné en entrainement au réseau de neurones et Et représente la t-ième erreur d'apprentissage.

Soit la fonction dim(i) qui donnne le nombre de neurones de la couche i. Soit

$$\theta = \begin{bmatrix} w_{j,k}^{(i)} & b_j^{(i)} \end{bmatrix} \tag{2.14}$$

où $0 \leq i < r$, $0 \leq j < \dim(i)$ et $0 \leq k < \dim(i+1)$ qui représentent le vecteur rassemblant chacun des paramètres du réseau de neurones. Le problème d'optimisation associé est de trouver les paramètres du réseau de neurones permettant de minimiser l'erreur $E(\theta)$. Formellement, il faut trouver

$$\theta^* = \arg\min_{\theta} \frac{1}{N} \sum_{t=1}^{N} E_{t(\theta)} \tag{2.15}$$

2.5.5.3 Rétro propagation

L'idée de la rétro propagation est de faire circuler l'information sur la dérivée de la fonction d'erreur à partir de la couche de sortie, où l'erreur de prédiction est connue (la différence entre y et c), jusqu'à la couche d'entrée. Pour y arriver, il faut arriver à exprimer la dérivée de la fonction d'erreur d'un nœud en fonction de l'information donnée par les couches suivantes. De cette façon, les valeurs attendues de la couche de sortie pourront être rétro propagées vers l'entrée. La dérivée, à la sortie, de la fonction d'erreur (3) est donnée par

$$\frac{\partial E}{\partial y_i} = y_i - c_i \tag{2.16}$$

Il est possible d'arriver à la dérivée de la fonction d'erreur par rapport à chacun des poids du réseau de neurones, en appliquant la règle de dérivée en chaine : [20]

$$\frac{\partial E}{\partial w_{j,k}^{(i)}} = \frac{\partial E}{\partial e_k^{(i+1)}} \frac{\partial e_k^{(i+1)}}{\partial w_{j,k}^{(i)}} = \frac{\partial E}{\partial e_k^{(i+1)}} s_j^{(i)} \tag{2.17}$$

Finalement, comme les biais sont également des paramètres impliqués dans l'apprentissage d'un réseau de neurones, il faut aussi pouvoir leur trouver une valeur optimale. Il est donc nécessaire de trouver la dérivée de l'erreur par rapport aux biais. De la même façon qu'avec les poids, un biais n'affecte l'erreur de prédiction qu'au travers de l'entrée du nœud auquel il est rattaché.

$$\frac{\partial E}{\partial b_j^{(i)}} = \frac{\partial E}{\partial e_j^{(i+1)}} \frac{\partial e_j^{(i+1)}}{\partial b_j^{(i)}} = \frac{\partial E}{\partial e_j^{(i+1)}} \tag{2.18}$$

2.5.5.4 Calcul de la dérivée seconde

Pour pouvoir minimiser la fonction d'erreur de manière efficace, plusieurs algorithmes d'optimisation nécessitent des informations sur la dérivée seconde qui représente le comportement

de la fonction autour d'un point. Dans le cas de la fonction d'erreur, une fonction avec un domaine dans Rm, sa dérivée seconde est en fait la matrice hessienne

$$M = \begin{bmatrix} \frac{\partial^2 E}{\partial \theta_1{}^2} & \frac{\partial^2 E}{\partial \theta_1{}^2 \partial \theta_2} & \cdots & \frac{\partial^2 E}{\partial \theta_1{}^2 \partial \theta_n} \\ \frac{\partial^2 E}{\partial \theta_1{}^2 \partial \theta_1} & \frac{\partial^2 E}{\partial \theta_2{}^2} & \ddots & \frac{\partial^2 E}{\partial \theta_2{}^2 \partial \theta_n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial^2 E}{\partial \theta_n{}^2 \partial \theta_1} & \frac{\partial^2 E}{\partial \theta_n \partial \theta_2} & & \frac{\partial^2 E}{\partial \theta_n{}^2} \end{bmatrix} \quad (2.19)$$

Malheureusement, cette matrice, pour de grandes dimensions, devient difficile à stocker et à calculer, même pour un ordinateur. Puisque la plupart des algorithmes utilisent plutôt le produit de la matrice hessienne avec un vecteur arbitraire v (résultant en un vecteur de dimension m), il est plus facile de calculer directement ce vecteur plutôt que la matrice hessienne entière.

En se basant sur le principe de la propagation de valeurs exposé plus haut, il est possible de trouver la valeur de Hv en apportant quelques changements aux formules déjà données. Cette technique est attribuée à Pearlmutter.

$$Hv = R\{\nabla_\theta E(\theta)\} \quad (2.20)$$

2.5.5.5 Algorithme d'apprentissage

Comme il a été mentionné plus tôt, le but de la phase d'apprentissage est de trouver le vecteur de paramètres w qui minimise la fonction d'erreur. La descente de gradient est la plus utilisé dans ce domaine. Descente de gradient

L'apprentissage par descente de gradient s'apparente à un algorithme d'optimisation. Ayant obtenue la fonction de coût qui montre l'erreur entre la sortie voulue (cible) et la sortie du réseau qui est en fonction des poids des réseaux, l'apprentissage est de minimiser cette erreur. [19] [20]

a. Concept

La descente de gradient le calcul des poids qui minimise l'erreur en recherchant le minima de la fonction coût qui est donnée par l'inverse de la direction du gradient. La méthode de Descente de gradient consiste à appliquer un algorithme de descente de gradient sur l'ensemble des couples

(x,c). Cette façon de procéder consiste à trouver le gradient de l'erreur de chacun des couples et d'en faire la moyenne. Comme le gradient est un vecteur pointant vers la direction de plus forte croissance de la fonction d'erreur, déplacer les paramètres dans la direction opposée au gradient fait diminuer l'erreur.

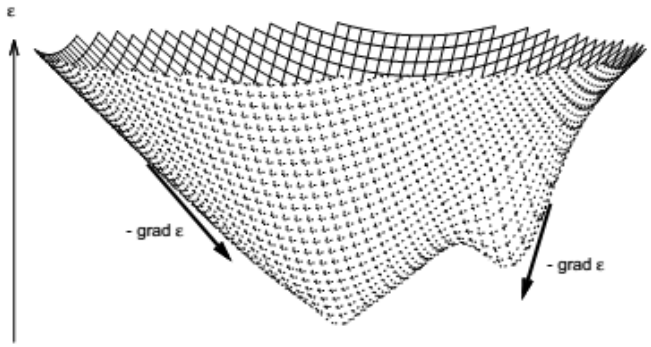

Figure 2.18 : *Apprentissage par descente du gradient*

La fonction du coût est donnée par :

$$J(w) = \frac{1}{2}\sum_{i}(cible^{(i)} - sortie^{(i)})^2 \tag{2.21}$$

La magnitude et la direction du changement à affecter au poids est donnée par la direction opposé du gradient de la fonction de coût:

$$\Delta w_j = -n\frac{\partial J}{\partial w_j} \tag{2.22}$$

Ici n est la vitesse d'apprentissage. Un taux trop élevé pourrait nuire à l'algorithme qui sauterait par-dessus du déplacement idéal. Un taux trop petit va tendre le temps de convergence vers l'infini. [21]

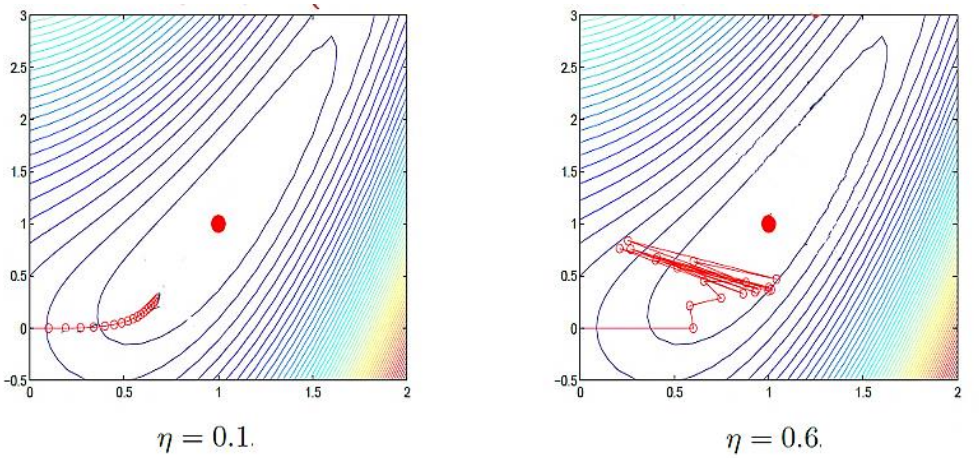

Figure 2.19 : *Exemple de taux d'apprentissage mal défini*

Les poids du réseau sont mises à jour après le parcours des données de la manière suivante :

$$w \coloneqq w + \Delta w \tag{2.23}$$

La valeur de Δw est calculée comme suit :

$$\Delta w_j = -n \frac{\partial J}{\partial w_j} \tag{2.24}$$

$$\Delta w_j = -n \sum_i \left(cible^{(i)} - sortie^{(i)}\right)\left(-x_j^{(i)}\right) \tag{2.25}$$

$$\Delta w_j = n \sum_i \left(cible^{(i)} - sortie^{(i)}\right)\left(x_j^{(i)}\right) \tag{2.26}$$

b. Descente de gradient stochastique

L'optimisation par descente de gradient le calcul du gradient du coût se base sur tout l'ensemble de la donnée d'apprentissage, il est aussi appelé batch descente de gradient. Dans le cas de large donnée d'apprentissage, qui est souvent le cas avec les réseaux à convolution, il faut énormément de temps et de calcul pour converger vers le minima global. Et en pratique on n'aura pas assez de mémoire pour l'ensemble de données. [20][21]

Algorithme de descente de gradient par lot :

For one or more epochs:

For each weight j

$w_j \coloneqq w + \Delta w_j, where: \Delta w_j = n \sum_i \left(cible^{(i)} - sortie^{(i)}\right)\left(x_j^{(i)}\right)$

Dans la descente de gradient stochastique, on met à jour les poids après chaque unité de données. Le terme stochastique vient du fait que le gradient basé d'une seule donnée d'apprentissage est une approximation stochastique du gradient réel. L'algorithme utilisé est la suivante:

For one or more epochs, or until approx. cost minimum is reached:

For training sample i:

For each weight j:

$$w_j \coloneqq w + \Delta w_{j,} where: \ \Delta w_j = \ n\left(cible^{(i)} - sortie^{(i)}\right)\left(x_j^{(i)}\right)$$

A cause de cette nature stochastique le chemin vers le minimum n'est pas direct comme pour la Descente de gradient mais en zigzag. Toutefois il est certain que le SGD va converger vers un minimum. De plus il y a des méthodes qui permettent accélérer l'apprentissage :

- Une vitesse d'apprentissage adaptatif :

On choisit une constante d qui adapte la vitesse d'apprentissage en fonction de temps :

$$n(t+1) \coloneqq \frac{n(t)}{(1 + t * d)} \tag{2.27}$$

- L'utilisation du Momentum qui ajoute un facteur du précédent gradient pour accélérer la mise à jour :

$$\Delta w_j \coloneqq n\nabla J(w_{t+1}) + \propto \Delta w_t \tag{2.28}$$

2.6 Conclusion

On a vu dans ce chapitre que les réseaux de neurones artificiels ont beaucoup de point commun aux neurones biologiques qui ont inspiré leur création. Les poids synaptiques du neurone biologique est illustré par des poids dont les valeurs sont apprises lors de l'apprentissage. Et si l'activation d'un neurone biologique est géré par le potentiel de plusieurs composant, celui d'un neurone artificiel est régit par une fonction d'activation. Avant d'affecter le neurone à sa tâche, il est nécessaire de l'entrainer par des exemples de données de cette tâche. La technique la plus populaire et efficace pour apprendre le réseau est la propagation en arrière avec la descente de gradient.

CHAPITRE 3
RESEAU DE NEURONE A CONVOLUTION

3.1 Introduction

Découragé à cause du maigre résultat obtenu avec les réseaux de neurone standard, cette branche de l'apprentissage automatique a failli s'éteindre. A l'époque, le vrai problème résidait sur l'apprentissage du réseau. La technique nécessaire pour calculer et formuler les différents poids du réseau manquait. Il était dès lors impossible d'entrainer un réseau dès que celui-ci est composé de plusieurs couches cachées. Une publication sur l'apprentissage des réseaux profond combiné à la possibilité de parallélisassions des calculs a ravivé les recherche jusqu'à l'effervescence que connait ce domaine aujourd'hui. De là est aussi né le concept d'apprentissage profond ou « deep learning » à laquelle on appelle un réseau comprenant plusieurs couches empilées les uns sur les autres. Le deep learning stipule aussi qu'à chaque couche on obtient une représentation de l'information de plus en plus abstrait, on part par exemple d'un ensemble de pixel jusqu'à arriver à des concepts de haut niveau comme un chien ou un chat. Le réseau de neurone à convolution est le premier réseau profond à montrer des résultats concluant avec l'architecture LeNet qui fut utilisé pour la lecture des chèques.

Le choix de l'architecture d'un réseau de neurone se fait toujours en fonction des données à traiter, l'utilisation du réseau elle-même. Lorsqu'il y a des caractéristiques spatiales dans les données, il est recommandé d'utiliser les réseaux à convolution. On peut utiliser d'autre architecture comme le Deep Boltzman Machine pour ce genre de taches, toutefois la performance obtenue n'atteint pas celles des CNN. En termes de classification d'images, les CNN sont les plus performants sur l'état de l'art. La précision de certain réseau comme le GoogLeNet dépasse même la capacité visuelle humain avec seulement 6.67% d'erreur.

Les réseaux de neurone à convolution sont des variant des perceptrons multicouches inspiré de la biologie. A partir des travaux de Hubble et Wiesel sur le cortex visuel des chats, on sait que les cortex visuels contiennent de complexe arrangement de cellules qui sont sensibles à des petites régions du champ visuel. C'est ce concept qui a été ramené au réseau de neurone sous la forme de convolution.

Dans ce chapitre nous allons étudier le CNN et son fonctionnement. Dans un premier lieu, on parle des différentes motivations qui encouragent l'utilisation de ce réseau. Ensuite, nous abordons la

connectivité et l'architecture sera vue en détail. Une partie sera consacrée aux techniques utilisées pour l'apprentissage de ce genre de réseau. En dernier lieu nous verrons l'état de l'art sur les réseaux de neurone à convolution.

3.2 Motivation

Les réseaux de neurones à convolution révèlent 3 principaux avantages à son utilisation : une connectivité sparte, le partage de paramètres. Ces termes sont approfondis dans les paragraphes qui suivent. [23] [24]

3.2.1 Connectivité sparte

Le réseau de neurones traditionnels utilise une multiplication de matrice pour représenter les interactions entre chaque entrée et chaque sortie. Cela signifie que tous les neurones de sortie communiquent avec ceux des entrées. Prenons un réseau qui traite des images, pour une image de taille 32x32x3 (32 pixel de hauteur, 32 pixel de largeur, 3 canal de couleur), un simple neurone complètement connecté possède dans son première couche caché 32*32*3 = 3072 poids, ce qui semble encore traitable, mais ce genre de structure ne convient pas à des taille d'image plus élevé ; par exemple une image de 600x400x3 nous amène à un réseau qui aura 600*400*3 = 720 000 poids .

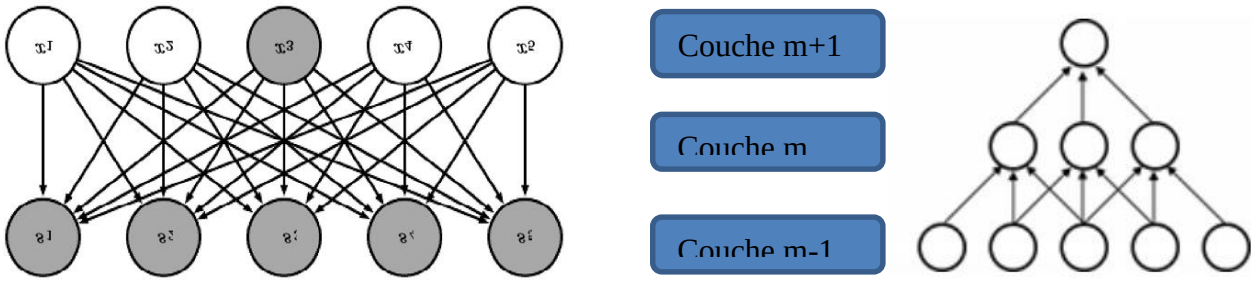

Figure 3.01 : *Un réseau de neurone complètement connecté à gauche et à connectivité sparte à droite*

Les réseaux de neurone à convolution ou convnet exploitent la corrélation spatiale locale en renforçant les connections entre les neurones de couche adjacent. En d'autre terme, les entrées des unités cachées dans une couche m sont issues d'un ensemble d'unité de la couche m-1. La figure 3.01 illustre ce concept. Cela est obtenu en utilisant un noyau plus petit que l'entrée. Par exemple, pour le traitement d'image, l'entrée peut avoir des centaines de millier de pixels, on peut détecter des petites contenues comme des bords avec un noyau de convolution qui n'occupe qu'une dizaine

ou centaine de pixels. On a alors besoin de stocker qu'un nombre réduit de paramètre, ce qui réduit l'occupation en mémoire du model ainsi que les calculs à effectuer.

3.2.2 Partage de paramètre

Dans les ConvNet, chaque filtre ***hi*** est répliqué sur le champ de vision en entier (toute l'image). Les unités répliqué partage les mêmes paramètres (vecteur de poids et biais) et forme ce qu'on appelle un « feature map ».

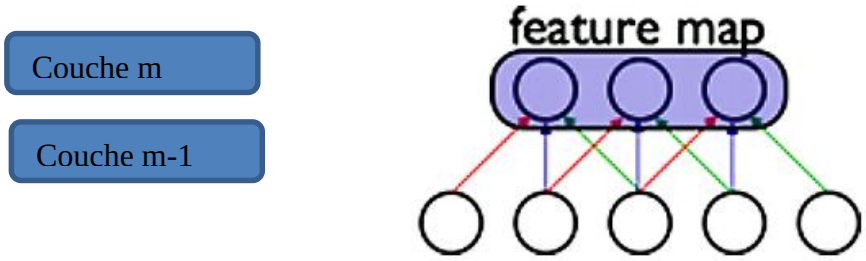

Figure 3.02 : *Partage de valeur de paramètre*

Dans la figure 3.02 on voit 3 unités cachés appartenant à la même feature map. Les poids ayant les mêmes couleurs sont partagés et contraint à être identique. Cela réduit considérablement le nombre de paramètre libre à apprendre. On peut utiliser la descente de gradient pour apprendre à ces paramètres partagés, en effectuant un changement à l'algorithme original. Le gradient des poids partagé est la somme de chacun des paramètres à partager.

3.2.3 **Connectivité**

Du fait que l'entré des CNN sont des images, l'architecture de ce type de réseau est ainsi contraint à prendre une forme particulière. Contrairement au réseau de neurone classique qu'on a vu au chapitre précédent, les couches des réseaux de neurone à convolution sont arrangées en 3 dimensions : longueur, hauteur et profondeur. Par exemple la couche d'entrée d'un réseau qui traite des images couleur de taille 200x200 pixels aura pour volume de dimensions 200x200x 3.

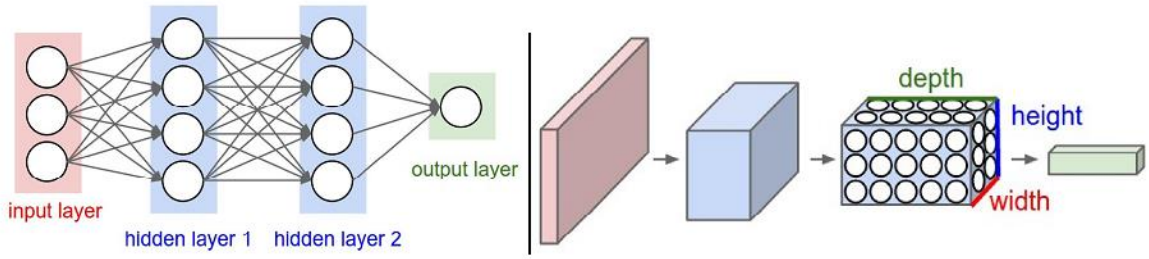

Figure 3.03 : *A gauche : réseaux de neurone classiquer, à droite : Architecture en 3D d'un CNN*

3.3 Architecture

On distingue principalement 3 types de couche dans un réseau de neurone à convolution : la couche de convolution, la couche de pooling ou de sous échantillonnage, et une couche classification. Ces différentes couches sont empilées pour former l'architecture d'un réseau de neurone à convolution. Nous verrons en détail la particularité de ces couches dans les paragraphes qui suivent. [25] [26]

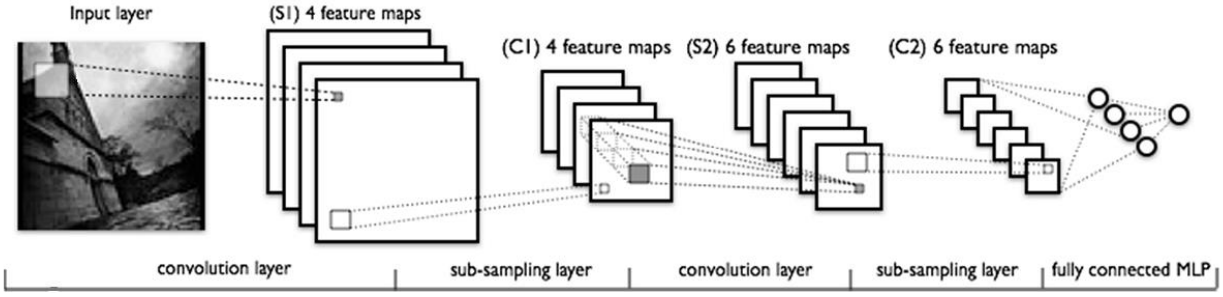

Figure 3.04 : *Le réseau LeNet et les couches qui le composent*

3.3.1 La couche de convolution

La couche de convolution est le cœur d'un réseau de neurone à convolution, sa sortie peut être interprétée comme un arrangement de neurone en 3 dimensions. Chacun calcul l'entrée et ses poids par un produit différentiel. [27]

3.3.1.1 L'opération de convolution

De manière générale, la convolution est une opération de 2 fonctions d'argument réel. Prenons l'exemple d'un capteur qui délivre à sa sortie un signal x(t) à un temps t, x et t sont réels. On suppose que le signal obtenue est bruité. De cela, pour réduire le bruit, on souhaite obtenir la moyenne de certaines mesures. Les plus récent mesure sont souvent les plus révélant, on utilise pour cela une moyenne pondéré qui met plus de poids au mesure récent. Soit alors w(a) la fonction de pondération, où est l'âge de la mesure. Si on applique la moyenne pondéré à tous moment, on obtient une nouvelle fonction s qui donne une estimation plus lisse : [22]

$$s(t) = \int x(a)w(t-a)da \tag{3.01}$$

Cette opération est appelée convolution, elle est notée par un astérisque :

$$s(t) = (x * w)(t) \tag{3.02}$$

Dans les réseaux de neurone à convolution, le premier argument (ici x) est l'entrée et le second argument (w) est le noyau. La sortie est souvent appelée « feature map ». On travaille souvent avec des données avec les ordinateurs, le temps est alors discrétisé, et à intervalle régulier. Ainsi l'index de temps t ne prend que des valeurs entières. On peut alors définir la convolution discrète par :

$$s(t) = (x * w)(t) \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{3.03}$$

La convolution s'utilise le plus souvent sur plusieurs axes en même temps. Par exemple si on a une image à deux dimensions en entrée, on utilise un noyau U à deux dimensions :

$$S(i,j) = (I * K)(i,j) \sum_{m} \sum_{n} I(m,n)U(i-m, j-n) \tag{3.04}$$

Certain « Framework » d'apprentissage automatique utilise une autre forme d'écriture de la convolution qui profite de sa propriété commutative :

$$S(i,j) = (I * K)(i,j) \sum_{m} \sum_{n} I(i-m, j-n)U(m,n) \tag{3.05}$$

a. Implémentation dans le CNN

La convolution discrète peut être vue en une multiplication de matrice. Toutefois le noyau doit avoir une forme particulière : les entrées de la matrice sont contraintes à être égales à d'autre entrée. Par exemple pour la matrice dite de Toeplitz, chaque ligne est contrainte à être égale à la ligne du dessus décalé d'un élément. En deux dimensions, une matrice à double bloc circulant correspond à une convolution. En plus de ces contraintes, une convolution correspond souvent à une matrice sparte où nombreux de ses entrées sont égales à zéro. Prenons l'exemple d'une image X à 3 pixels X1 X2 X3, et soit W le filtre de convolution de taille Mf = 2 [21]

$$W = [W_1, W_2]$$

$$X = [X_1, X_2, X_3]$$

avec

$$Z = [Z_1, Z_2]$$

On a

$$Z_1 = W_1 X_1 + W_2 X_2$$

$$Z_2 = W_1 X_2 + W_2 X_1$$

En généralisant,

$$Z_{i'} = \sum_{i=1}^{\infty} w_i X_{i'+i-1} \tag{3.06}$$

Cette opération est appelée corrélation, elle recherche les similarités entre la matrice X et le noyau W. On obtient la convolution en inversant (flip) le noyau W en $\bar{W}$:

$$\bar{W} = [W_2, W_1]$$

La formule devient alors :

$$Z_{i'} = \sum_{i=1}^{\infty} X_{i'+i-1} \bar{W}_{Mf-i+1} \tag{3.07}$$

b. Application sur une image

Dans un réseau de neurone à convolution, la convolution s'illustre par le parcours de l'image avec un filtre afin d'en extraire les composants qui nous intéressent. La figure 3.06 illustre ce concept. La taille des filtres utilisés est toujours plus petite que celle de l'image. [28] [29]

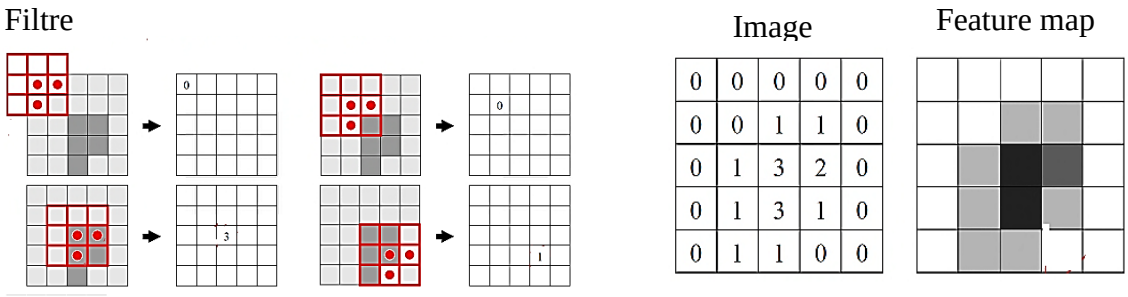

Figure 3.05 : *Extraction d'un feature map par la convolution d'une image avec un filtre*

A la sortie d'une couche de convolution on obtient un ensemble de feature map qui peut être visualisé comme une image. La figure 3.06 montre les résultats de convolutions effectuées sur une image qui s'apparente à une détection de contours.

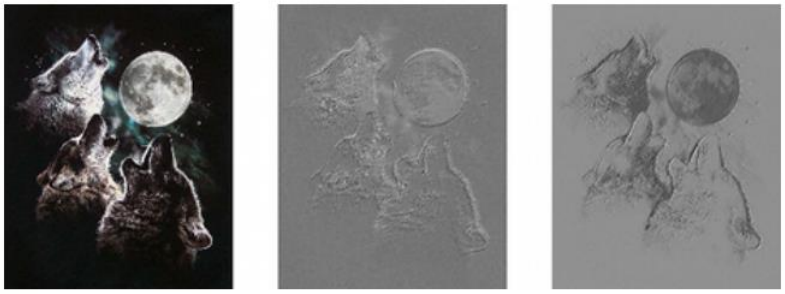

Figure 3.06 : *Résultats de convolution avec des filtres initialisé de façon aléatoire*

La couche de convolution comporte plusieurs filtres. Il y a ainsi autant de feature map que de filtre. Chaque filtre est appliqué à l'image et partage certain paramètres.

c. Méthode de calcul

Dans les applications réelles, les images à traiter sont des images couleur. L'expression de la sortie de la couche est exprimée suivant 3 dimensions : largeur, hauteur et profondeur.

Soit X le tensor à l'entrée du réseau de neurone, qui est composé de trois dimensions i pour la largeur, j pour la hauteur et f pour la profondeur qui est le différent canal si l'entrée est une image RGB et les empilements de feature map s'il y une couche de convolution en amont. Y est l'empilement de feature map résultant du calcul, avec i' j' et f' comme composante. Et θ désigne le paramètre à apprendre composé de Wf, Hf et F. La formule suivante donne la fonction d'une couche de convolution en entier. [21]

$$Y_{i',j',f'} = b_{f'} + \sum_{i=1}^{H_f} \sum_{j=1}^{W_f} \sum_{f=1}^{F} x_{i'+i-1,j'+j-1,f} \theta_{ijff'} \tag{3.08}$$

Ici $b_{f'}$ désigne le « Bias » de la couche exprimé.

L'apprentissage des réseaux de neurone nécessite souvent le calcul du gradient de l'erreur par rapport à l'entrée et le gradient par rapport aux paramètres. La formule donne cette valeur, qui nous sera utile plus tard.

$$\frac{\partial E}{\partial \theta_{ijff'}} = \sum_{i'j'f'} \delta^{l+1}_{i'j'f'} \frac{\partial f_{i'j'f'}(x;\theta_{f'})}{\partial \theta_{i'j'f'}} \tag{3.09}$$

$$= \sum_{i'j'f'} \delta^{l+1}_{i'j'f'} x_{i'+i-1,j'+j-1,f} \tag{3.10}$$

Le gradient par rapport aux paramètres s'exprime par :

$$\frac{\partial E}{\partial \theta_{ijff'}} = \sum_{i'j'f'} \delta^{l+1}_{i'j'f'} \frac{\partial f_{i'j'f'}(x;\theta_{f'})}{\partial \theta_{i'j'f'}} \tag{3.11}$$

$$\frac{\partial E}{\partial \theta_{ijff'}} = \sum_{i'j'f'} \delta^{l+1}_{i'j'f'} \theta_{i\ -i'+1,-j'+j+1,f,f'} \tag{3.12}$$

3.3.1.2 Fonction d'activation

Comme une unité de neurone standard, les neurones de la couche de convolution disposent aussi de fonction d'activation. Les régulateurs se trouvent après l'opération de convolution et leur sortie deviennent les sorties de la couche de convolution. On l'utilise pour induire une non-linéarité au réseau sans pour autant affecter les champs de réceptions. [21][23]

a. Rectified Linear Unit

Les fonctions, hyper-tangente et sigmoïde, qu'on a vue dans le chapitre 3 peuvent être utilisé mais le Rectified Linear Unit ou ReLU est plus apprécié. Le ReLU est aussi connu comme fonction rampe et est défini par :

$$f(x) = \max(0., x) \tag{3.13}$$

Des textes dénomment qu'il est plus plausible biologiquement que les plus utilisé fonction sigmoïde. Une approximation plus lisse de cette fonction est la fonction soft plus défini par :

$$f(x) = \ln(1 + e^{x}) \tag{3.14}$$

La figure 3.07 représente la vitesse auquel le réseau de neurone retrouve sa valeur d'erreur pendant l'apprentissage ; en utilisant la fonction d'activation hardmax et softplus(ou ReLU)

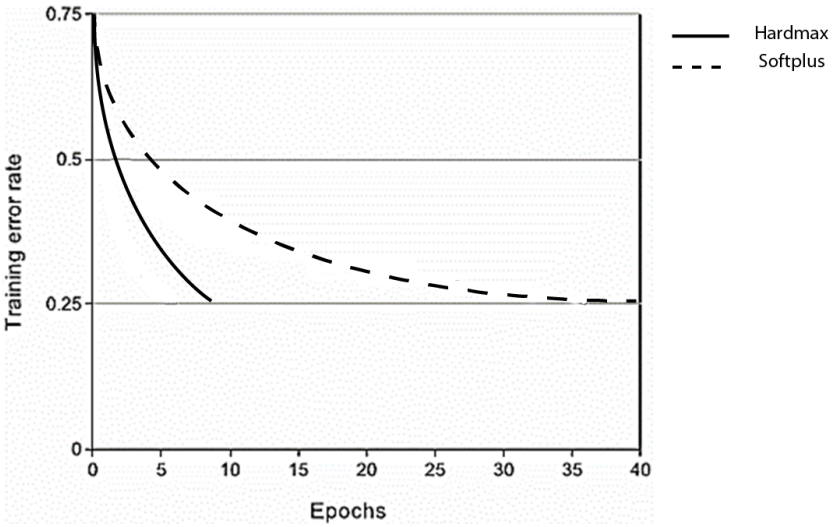

Figure 3.07 : *Comparaison entre la fonction sigmoïde, le hard max et le softplus*

b. Avantage du ReLU

Le Rectified Linear Unit ou ReLU est plus privilégié grâce à plusieurs de ses propriétés :

- La principale raison d'utiliser le ReLU est qu'il peut être efficacement calculé.

- Il a aussi été démontré que le ReLU converge facilement même sans préapprentissage.
- Le gradient de la fonction sigmoïde se volatilise en la croissance ou la décroissance de x, contrairement à celui de la fonction max.
- L'utilisation du ReLU en tant que fonction d'activation induit aussi de la spartiété dans les couches cachées.
- La fonction sigmoïde a un intervalle de [0 1], quand à celui du ReLU est de [0 $+\infty$]. Le sigmoïde est alors plus utilisé pour modéliser une probabilité mais le ReLU pour modéliser des un Réelle positif qui est le plus adapté à une couche de convolution. [22]

3.3.2 La couche de Pooling

Un autre concept très important des réseaux de neurone à convolution est le Pooling, qui est une forme de sous échantillonnage non linéaire. Cette couche reçoit en son entrée les valeurs issue de la couche de convolution qui le précède ou du régulateur s'il y en a. Le principe du Pooling est de remplacer la sortie du réseau à certaine location par une addition statistique de ses entourages. Cependant elle n'affecte que la dimension spatiale, seulement sur x et y ; Chaque feature maps doit être alors traité indépendamment. [22]

3.3.2.1 Rôles

La couche de pooling a été introduit grâces aux divers avantages qu'elle présente. On en distingue principalement 3 différents rôles et avantages :

- Permette au réseau d'être invariant à des légères transformations sur l'entrée : translation, rotation. Le pooling sur une région produit une invariance à de légères translation, mais si on pool sur des sorties de convolutions séparément paramétré, on peut apprendre à quel genre de transformation être invariant. Par exemple, les deux entrées suivantes vont activer une même unité après la couche de pooling.

Figure 3.08 : *Le chiffre 5 présenté avec une rotation*

- Offrir aux unités de couches suivantes un champ de vision plus large.

- Réduire la dimension des features map issue de la couche de convolution. Le pooling est aussi utilisé comme sous échantillonneur. La taille de la fenêtre F et le stride S conditionnent la taille de la sortie, qui peut être réduit jusqu'à la S fois de l'entrée.

3.3.2.2 Différents types de Poolling

Les divers types de pooling se distinguent sur l'opération utilisée pour extraite les valeurs dans la fenêtre de pooling. Dans la littérature, le max pooling est le plus apprécié, il y a aussi le pooling par moyenne qui présente des avantages intéressant. Récemment on parle même de réseau tout-convolution qui incube de retirer la couche de pooling.[21][25] [29]

a. Le pooling par moyen

Introduit par l'architecture original du réseau LeNet, le 'average pooling' ou pooling par moyen est la première méthode de pooling réussie appliquée au réseau de neurone à convolution. Le procédé se fait en calculant la moyenne des valeurs comprise dans la fenêtre de pooling.

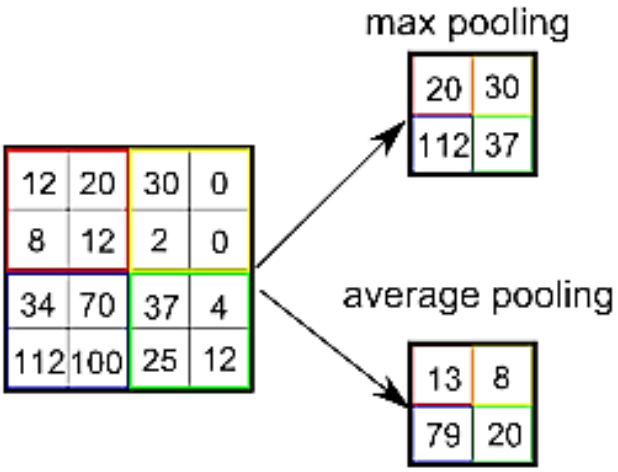

Figure 3.09 : *Exemple de pooling 2 x 2 appliqué sur une matrice*

b. Le max-pooling

Le max pooling extrait la sortie maximum sur une région délimité. D'après une publication sur l'analyse de l'effet de différents pooling, le max pooling est la mieux approprié au CNN. On peut définir cette opération par :

$$y_{i',j'} = \max_{ij\omega(i'j')} x_{ij} \tag{3.15}$$

D'autres études ont toutefois révélé que l'extraction des valeurs maximum est d'une manière trop destructive pour les features map et qui engendrerait trop de perte de d'information. D'autre système

vient pour pallier ce problème, comme le full-conv. La figure illustre les deux types de pooling avec une taille de 2 x 2 et sans écrasement.[29][30][31]

3.3.3 La couche de classification

En étant la dernière couche du réseau, la couche de classification génère la sortie. Il a pour fonction de designer la classe au quelle appartient l'entrée en fonction de la feature map qui lui est présenté. Un perceptron multicouche est souvent utilisé mais certaines publications ont montrées de bons résultats avec l'utilisation des Support Vector Machine (SVM). Le nombre de neurone pour la couche d'entrée du MLP est dicté par la sortie de couche de pooling et le nombre de couche caché est choisi en fonction du nombre de classes.

3.4 Visualisation et fonctionnement

La technologie du réseau de neurone à convolution fait partie de la branche de l'apprentissage automatique : l'apprentissage profond. Un réseau profond ne concerne pas seulement le niveau de couches (plusieurs couches cachées), mais concerne la représentation de la donnée même à l'intérieur du réseau. Le concept apporté par l'apprentissage profond est la présence de représentation hiérarchique de données allant de la forme la plus brute (pixel) vers un autre plus abstraite, un chat par exemple. [21]

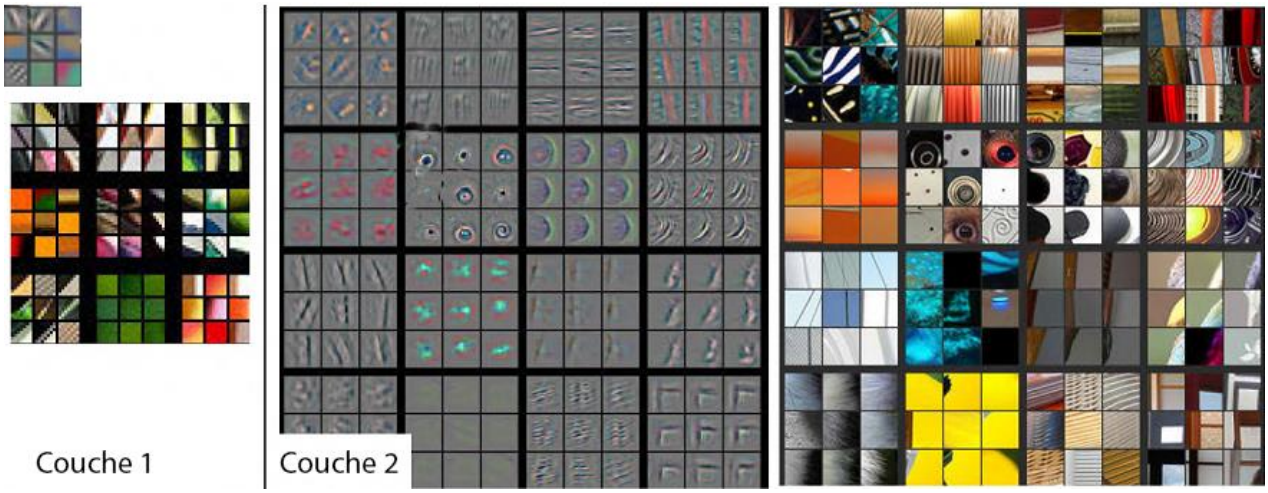

Figure 3.10 : *Visualisation des filtres appris par un CNN*

La hiérarchie de représentations est très visible dans les CNNs. Prenons l'exemple d'un réseau déjà entrainé et comprenant 5 couches de convolution. Les filtres de la première couche sont des simples détecteurs de contours, de lignes. Dans la deuxième couche on aperçoit des formes géométriques :

cercle, carré, ovale. Dans la troisième couche on voit des détecteurs de divers motifs et texture. A la 4-ème couche, il y a des morceaux d'objet, des pattes d'animal. Dans dernière couche on peut enfin apercevoir des formes complètes de chien, de voiture…etc.

3.5 Apprentissage des réseaux de neurone à convolution

Les réseaux de neurone à convolution sont généralement entrainés de manière supervisée : on présente un couple d'image et d'étiquette à l'entrée et à la sortie du réseau et ensuite on calcule les poids minimisant l'erreur. On a recourt à l'apprentissage non supervisé que si on manque de donnée étiqueté. Les CNN utilisent le même algorithme d'apprentissage que les réseaux de neurone standard : le back propagation avec la descente de gradient. Des récentes publications ont introduit de nouveaux concepts pour améliorer l'apprentissage: le dropout, la normalisation de batch. [32]

Figure 3.11 : *Exemple d'images pour l'apprentissage*

3.5.1 Back propagation et descente de gradient

Pour rappel de ce qu'on a vus au chapitre précèdent, le back propagation n'est que le fait de calcul du gradient des couches un à un depuis la sortie vers la couche d'entrée. Pour cela on applique la méthode de calcul dite de chainrule. Avant le back propagation il est d'abord nécessaire de propager une entrée dans le réseau en entier.

Les CNN utilisent cet algorithme pour l'apprentissage avec la descente de gradient. Pour une convolution d'une matrice à 1 dimension suivie d'une couche de Pooling:

Assumons qu'il y a y^{l-1}de taille N entrée de la couche $l-1$, m est la taille du noyau (ou filtre), on note chaque poids w_i, la sortie est x^l :[33] [34]

$$x_i^l = \sum_{a=0}^{m-1} w_a y_{a+i}^{l-1} \tag{3.16}$$

Où $y_i^l = f(x_i^l)$ et f est la fonction d'activation.

Considérons la fonction E la fonction d'erreur, et $\frac{\partial E}{\partial y_i^l}$ la fonction d'erreur au niveau de la couche de convolution. La dépendance de l'erreur par rapport au poids dans la couche précédente est

$$\frac{\partial E}{\partial wa} = \sum_{a=0}^{N-m} \frac{\partial E}{\partial x_i^l} \frac{\partial x_i^l}{\partial w_a} \tag{3.17}$$

$$\frac{\partial E}{\partial wa} = \sum_{a=0}^{N-m} \frac{\partial E}{\partial w_a} y_{i+a}^{l-1} \tag{3.18}$$

Où nous avons la somme sur toute l'expression contenant w_a, qui est$N - m$. Pour calculer le gradient on doit calculer le premier terme par :

$$\frac{\partial E}{\partial x_i^l} = \frac{\partial E}{\partial y_i^l} \frac{\partial y_i^l}{\partial x_i^l} \tag{3.19}$$

$$= \frac{\partial E}{\partial y_i^l} \frac{\partial}{\partial x_i^l} f(x_i^l) \tag{3.20}$$

De là, nous pouvons calculer l'erreur et la propager en arrière vers la couche précédente :

$$\delta_a^{l-1} = \frac{\partial E}{\partial y_i^{l-1}} \tag{3.21}$$

$$= \sum_{a=0}^{m-1} \frac{\partial E}{\partial x_{i-a}^l} \frac{\partial x_{i-a}^l}{\partial y_i^{l-1}} \tag{3.22}$$

$$= \sum_{a=0}^{m-1} \frac{\partial E}{\partial x_{i-a}^l} w_a^T \tag{3.23}$$

w^Tréfère à la matrice transposé de w .

L'erreur dans la couche suivante peut donc être calculé par :

$$\delta^l = (w^l)^T \delta^{l+1} f'(x^l) \tag{3.24}$$

Avec la couche de sous échantillonnage (pooling) on a

$$\delta^l = upsample((w^l)^T \delta^{l+1} f'(x^l)) \quad (3.25)$$

L'opération u$psample$ propage l'erreur à travers la couche de pooling. Cette opération est effectué pour chaque couche jusqu'à la couche d'entrée.

3.5.2 Le Dropout

L'idée du dropout vient de la théorie du rôle du sexe sur l'évolution. La reproduction sexuelle prend la moitié des gènes d'un parent et la moitié de l'autre, et avec de petite mutation aléatoire. Le rôle de la reproduction sexuelle n'est pas juste l'ajout de nouveau gène qui va se partager sur la population mais aussi de faciliter ce processus en réduisant les coadaptations complexe entre les gènes.

Ce concept a été rapporté au réseau de neurone par le dropout qui s'effectue en enlevant aléatoirement certains neurones pendant la phase d'apprentissage. Ces neurones sont remis à la phase de test en gardant leurs valeurs d'initialisation.[35]

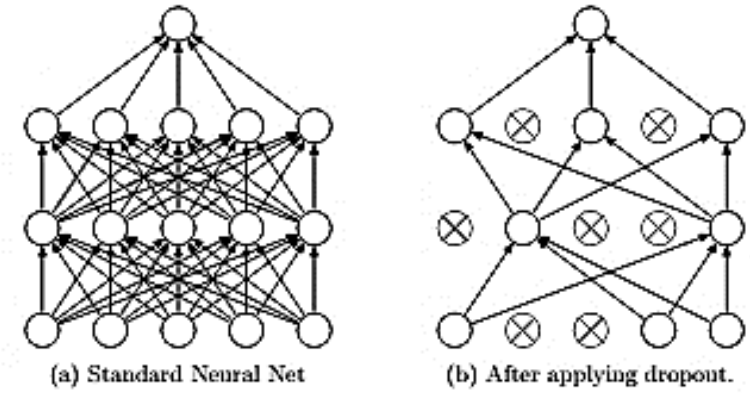

Figure 3.12 : *Un réseau de neurone standard auquel en a appliqué le dropout*

Le principal rôle du dropout est de réduire le sur-apprentissage ou l'overfitting, qui fait en sorte que les unités de neurone sont trop interdépendantes. Le choix de l'unité qui est enlevé est aléatoire. De la façon la plus simple, chaque unité est retenue avec une probabilité p fixe indépendante des autres unités, où p peut être choisi en utilisant un ensemble de validation ou fixé à 0,5 et qui semble être adapté à plusieurs variétés d'architecture. Les réseaux auquel on a appliqué le dropout ont montré un gain conséquent sur le test d'erreur, comme le montre la figure suivante qui illustre le test effectué sur un réseau comprenant 2 à 4 couches cachées.

3.5.3 Augmentation de données

L'apprentissage de réseau large et profond nécessite l'acquisition de beaucoup d'exemples. La plus facile et la plus commune des méthodes d'élargissement du set de donnée est d'utiliser des transformations sur le set d'image elle-même en conservant les labels. On utilise le set d'image initial pour faire une copie auquel on ajoute des transformations aléatoires. Ensuite le réseau est entrainé avec l'ensemble de données initiale et la copie modifiée.[36]

Plusieurs sortes de transformation sont disponibles pour y parvenir :

- Translation d'image
- Réflexion horizontale
- Déformation élastique
- Distorsion photométrique
- Modification des intensités des canaux RGB

En plus d'augmenter le nombre d'images pour l'apprentissage, cette technique améliore la généralisation du réseau en réduisant la dépendance des poids au set d'images.

3.5.4 Normalisation de batch

Les réseaux de neurones profond sont atteints par un problème appelé « internal covariance shift ». Lors de l'apprentissage ce phénomène se présente pendant l'apprentissage, la distribution de l'entrée et l'activation de chaque couche change quand les paramètres de la couche qui le précède change. Cela réduit la performance du réseau et ralenti l'apprentissage. La solution est de fixer la distribution de l'entrée x pendant l'apprentissage. Le Normalisation de batch ou BN (Batch Normalization) peut être appliquée à toutes les activations du réseau. [37]

Dans le cas d'un réseau de neurone à convolution, la transformation BN est ajoutée avant le non linéarité. Prenons l'exemple d'une transformation affine suivie d'un non linéarité :

$$z = g(Wu + b) \tag{3.26}$$

Où W le poids et b le biais qui sont les paramètres à apprendre du réseau, et g(.) le non linéarité (sigmoïde ou ReLU). La transformation de normalisation de batch est ajouté avant g(.) pour normaliser x = Wu + b. On peut ignorer la normalisation du biais, z devient alors

$$z = g(BN(Wu + b)) \tag{3.27}$$

$$z = g(BN(Wu)) \tag{3.28}$$

L'ajout de la normalisation de batch dans un réseau permet d'utiliser un taux d'apprentissage élevé ce qui accélère l'apprentissage. Ces réseaux convergent plus vite qu'un réseau sans BN, la figure 3.13 montre l'apport de la normalisation à la phase d'apprentissage. Il a aussi été prouvé que la classification est même est améliorée, l'utilisation du dropout n'est même plus nécessaire.

L'apprentissage d'un réseau avec de la normalisation de batch suit l'algorithme suivant :

Entrée : Réseau de neurone N avec un paramètre θ

Sortie : Réseau batch normalisé avec inférence N_{BN}^{inf}

1 : N_{BN}^{tr} N

2 : **for** k = 1 … K **do**

3 : Ajouter transformation $y^{(k)} = BN_{\gamma^{(k)},\beta^{(k)}}(x^{(k)})$ à N_{BN}^{tr}

4 : Modifier chaque couche dans N_{BN}^{tr} avec l'entrée $x^{(k)}$ pour prendre $y^{(k)}$

5 : end for

6 : Entrainer N_{BN}^{tr} pour optimiser les paramètres $\theta \cup \{\gamma^{(k)}, \beta^{(k)}\}_{k-1}^{K}$

7 : $N_{BN}^{inf} \leftarrow N_{BN}^{tr}$

8 : **for** k = 1…K do

9 : Traiter chaque mini lot B de taille m, et calculer la moyenne :

$E[x] \leftarrow E_B[\mu_B]$

$Var[x] \leftarrow \frac{m}{m-1} E_B[\sigma_B^2]$

10 : Dans N_{BN}^{inf} remplacer la transformation $y = BN_{\gamma,\beta}(x)$ par

$$y = \frac{\gamma}{\sqrt{Var[x] + c}} . x + (\beta - \frac{\gamma E[x]}{\sqrt{Var[x] + c}}$$

11 : end for

La transformation BN contient les paramètres γ et β qui doivent être appris par entrainement. Pour le CNN, ces paramètres sont appris par feature map et non par unité d'activation.

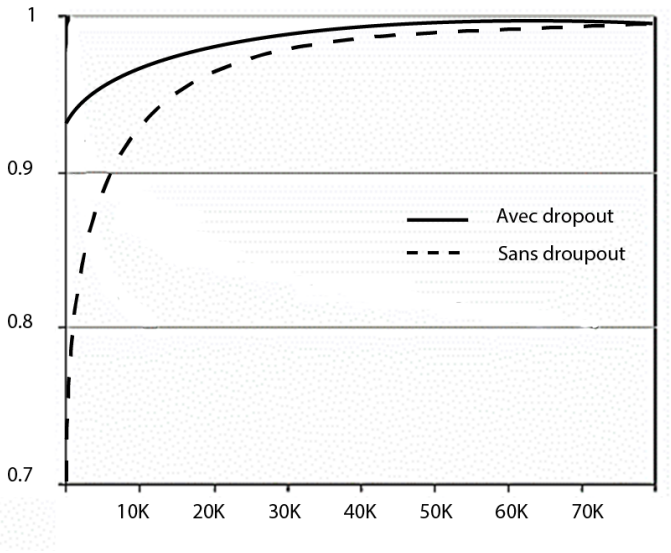

Figure 3.13 : *Effet du BN sur l'apprentissage*

3.6 Les architectures de références

Chacun des architectures que nous allons étudier sont les plus performantes à leur sortie ou ont gagné une compétition de classification d'image. Ils font partie intégrante de la littérature et sert ainsi de référence lors d'élaboration de nouvelle modèles ou algorithmes.

3.6.1 LeNet

Considéré comme pionnier dans la reconnaissance d'écriture manuscrite, Yann LeCun est le créateur de l'architecture LeNet. Le nouveau concept introduit par sa publication de l'époque est de de consacrer l'extraction de feature à un système d'apprentissage automatique. La méthode traditionnelle utilisée était l'extraction feature par un module manuellement défini, suivi d'un classificateur.

Le réseau à convolution pouvait remplacer ces modules avec une erreur de seulement 0,7 % sur le set d'image MNIST (Mixed National Institute of Standards and Technology) qui continent des écritures manuscrites de chiffres allant de 0 à 9. Dès lors, l'impact de cette technologie fut très grand que 90% partie des lecteurs de chèque de banque aux Etats Unis utilisait cette technique. [26]

3.6.1.1 Architecture

L'architecture du LeNet-5 contient 7 couches contenant des paramètres pouvant être appris. On a comme entrée une image de 32 x 32 pixels plus grands que la taille des images du MNIST : 20x20, pour recentrer l'image sur le champ de réceptions. La première convolution C1 est de 5x5 et contient 6 feature map aillant une taille de 28x28. Cela fait en somme 156 paramètres entrainables et 122 304 connections.

La couche est S2 est une sous échantillonnage de 2x2, réduisant la taille du feature map de moitié. Cette couche dispose de poids et de bias entrainable ainsi que d'une fonction d'activation : le sigmoïde.

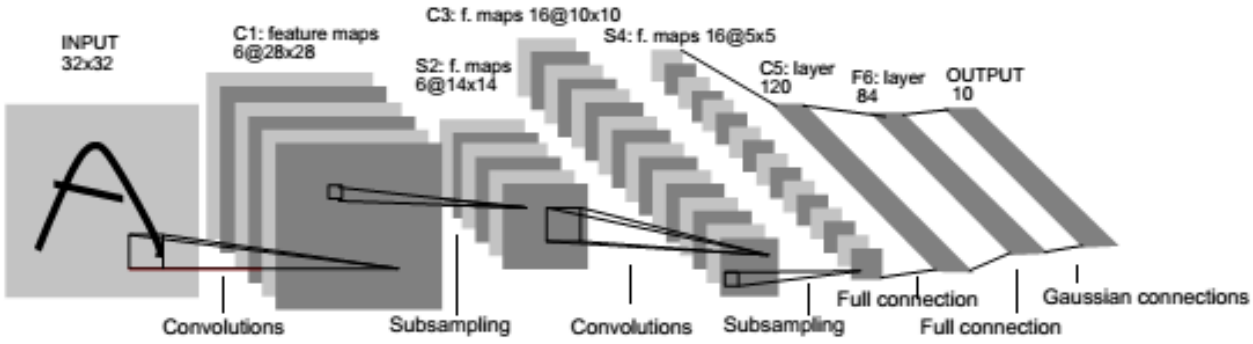

Figure 3.14 : *Architecture du LeNet-5*

La couche c3 est une convolution de 5x5 avec des parties non connectées à la couche S2. La couche S4 est une sous échantillonnage de 2x2 réduisant le feature map à une taille de 5x5. En C5 il y a aussi une couche de convolution de 5x5 avec une sortie de taille 1x1 qui est ensuite connecté à un réseau tout connecté avec une hyper tangente comme fonction d'activation. .

3.6.1.2 Caractéristiques

Le réseau LeNet est entrainé par la propagation en arrière de gradient. La fonction de coût est tirée d'une erreur quadratique. Avant l'apprentissage, les poids ont été initialisé aléatoirement avec une distribution uniforme compris dans -2.4/Fi et 2.4/Fi, où Fi désigne le nombre d'entrée. Cette approche favorise une convergence rapide lors de l'apprentissage.

Quoique le LeNet fût très performant sur le set d'image MNIST, l'architecture ne convenait pas à la classification d'image couleur ni sur d'autres sets d'image.

3.6.2 AlexNet.

Cette architecture est certainement la plus innovante dans l'histoire du réseau à convolution. Elle a été destinée à concourir à l'ImageNet Large-Scale Visual Recognition Challenge (ILSVRC10 et 12) une compétion annuelle de classification d'images sur un très grand set d'image labélisé : 1,2 million pour l'apprentissage, 50 000 pour la validation et 150 000 pour le test, la taille de l'image est fixe à 256x256. Ce réseau n'a pas seulement remporté la compétion mais aussi introduit de nouveau concept pour la classification : l'utilisation du ReLU comme activation, le DropOut, l'augmentation de données, et la plus importante est l'apprentissage en parallèle sur GPU (Graphical Processor Unit). [36]

3.6.2.1 Architecture

Le réseau AlexNet comprend 8 couches entrainables : 5 couches de convolution et 3 couches à connexion complète. On trouve ici la première utilisation des ReLU dans un réseau de neurone à convolution. Comme on l'a vu dans le chapitre précèdent, l'utilisation de ce non linéarité accélère l'apprentissage du réseau. Ce modèle ajoute un ReLU après chaque opération de convolution.

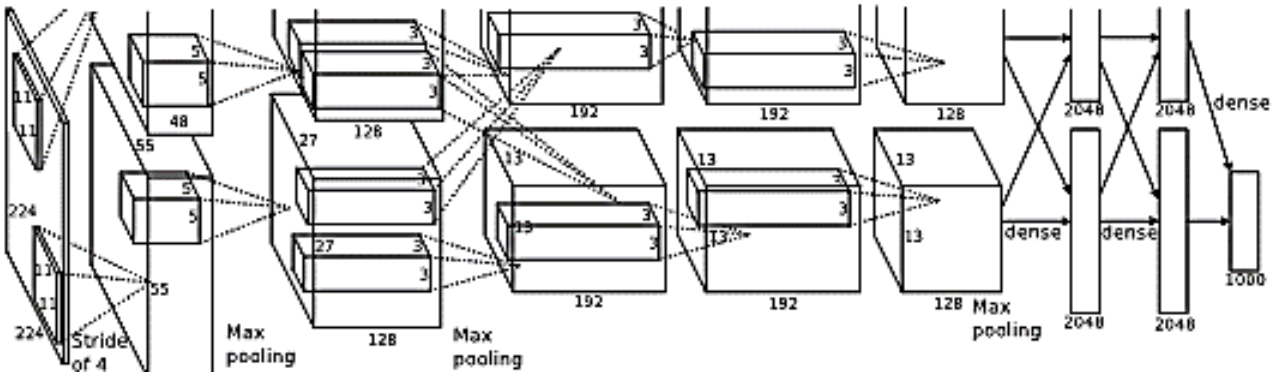

Figure 3.15 : *Architecture du AlexNet*

Le mode de pooling utilisé est le max pooling qui s'opère avec écrasement. Ils ont opté pour cette solution afin de réduire overfiting. On en compte 3 au total. Le Réseau est séparé en deux parties différentes pour faciliter l'apprentissage en parallèle sur 2 GPU. La première couche de convolution contient 96 noyaux de taille 11x11x3 avec un filtre de 224x224x3, avec un stride de 4 pixels. La deuxième convolution a 256 noyaux de taille 5x5x48. Les deux dernières couches de convolution sont connectées directement avec 384 noyaux de 3x3x192.

3.6.2.2 Optimisations

Ce réseau est le premier à implanter des concepts tels que le dropout, ou l'augmentation de donnée. Le modèle AlexNet est sujet à des overfiting parce que le l'architecture compte 60M de paramètres alors que le set d'apprentissage ne que contient que 1000 classes et 1.2 M d'images. Ils ont alors mis au point des différents algorithmes pour pallier ce problème. Delà vient les méthodes de dropout et d'augmentation de données qu'on a vues au chapitre précèdent. Ces spécificités de l'architecture ont permis au réseau de tenir la première place surpassant de loin ses concurrents.

Le réseau a été entrainé par descente de gradient stochastique avec un lot de 128 exemples. Cet algorithme a été couplé au momentum et au Weight decay. L'initialisation des poids s'est fait par une distribution gaussienne de déviation standard 0.01.

Une des grandes innovations apportées par AlexNet est l'apprentissage sur plusieurs GPU. Pour l'occasion, l'équipe d'Alex Krizhevsky ont développé une librairie pour l'apprentissage sur GPU en exploitant la technologie CUDA de Nvidia. L'architecture a été divisée sur 2 GTX 580 disposant de 3Gb chacun et entrainée en parallèle, toutefois il existe des partages de paramètres entre les deux parties du réseau. Cette technique est 20 fois plus rapide que l'apprentissage classique sur les CPU (Compute Processor Unit).

Model	**Top-1(Val)**	**Top-5(Val)**	**Top-5(Test)**
SIFT + FVs	-	-	26.2%
1 CNN	40.7%	18.2%	-
5 CNN	38.1%	16.4%	16.4%
1 CNN	39.0%	16.6%	-
7 CNN	36.7%	15.4%	15.3%

Tableau 3.01: *Comparaison de pourcentage de l'erreur de classification*

3.6.3 GoogLeNet

Szegedy, Christian, et al se sont basés sur l'idée que le moyen d'améliorer les performances des réseaux de neurone profond est d'augmenter leur taille, cela inclus d'augmenter sa profondeur, le nombre de niveau et aussi le nombre d'unité dans chaque niveau. Cependant, la grande taille des réseaux entraine deux grands problèmes : l'overfiting, causé par le nombre élevé de paramètres et l'augmentation exponentielle des calculs nécessaire. La solution à ces problèmes est d'introduire de la sparsité en remplacement des couches à connectivité complète. Les recherches d'Arora et al ont

montré que le réseau optimal qui pourrait représenter la distribution du set d'apprentissage peut être construite couche après couche en analysant les corrélations statistiques de la couche précédente et en parallélisant des neurones avec des sorties hautement corrélés. GoogLeNet implémente ces résultats de recherche avec une architecture dont le nom de code est Inception. [37]

3.6.3.1 Architecture

Le module Inception utilise 3 sortes convolution : 1x1, 3x3, et 5x5 afin de capturer une plus grande variété de structures et de formes. A cause des convolutions 3x3 et 5x5 qui sont très gourmandes en calcul, il a fallu introduire une convolution 1x1 réduire la dimension de l'entrée.

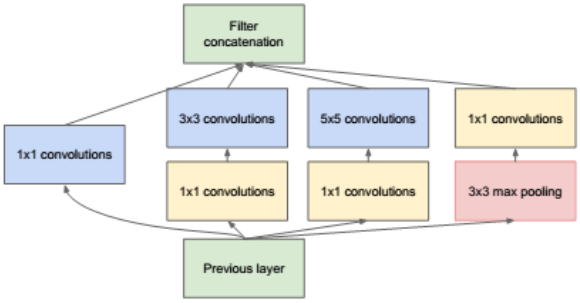

Figure 3.16 : *Le Module Inception*

Cette réduction ne concerne que la dimension et garde l'information nécessaire. Après chaque convolution de 1x1, il aussi y a une utilisation de ReLU ce qui ajoute plus de non-linéarité.

Le GoogLeNet consiste en un empilement de 9 modules d'Inception avec plusieurs couche en en amont. L'ensemble compte au total 22 couches entrainable et 27 si on compte les couches de pooling. Ce qui en fait que ce réseau possède 12 fois plus de paramètres que le réseau AlexNet ce qui nécessite 2x plus d'opération. L'équipe a suivi les recommandations de M.Lin sur l'utilisation de l'average pooling avant la couche de classification. La profondeur du réseau a entrainé des problèmes sur l'efficacité de la propagation en arrière de gradient. Leur solution est l'ajout de classificateur auxiliaire pour faciliter l'apprentissage.

3.6.3.2 Apprentissage

L'apprentissage a été effectué avec le framework DistBelief, en utilisant que l'implémentation sur CPU. L'équipe a utilisé la descente de gradient stochastique avec un momentum de 0.9, la vitesse d'apprentissage est rabaissée de 4% à chaque 8 epochs. A l'occasion de la compétition ils ont aussi utilisé l'augmentation de donnée en utilisant des patchs de taille allant de 8% à 100% de l'image. Avec un ratio contraint entre ¾ et 4/3.

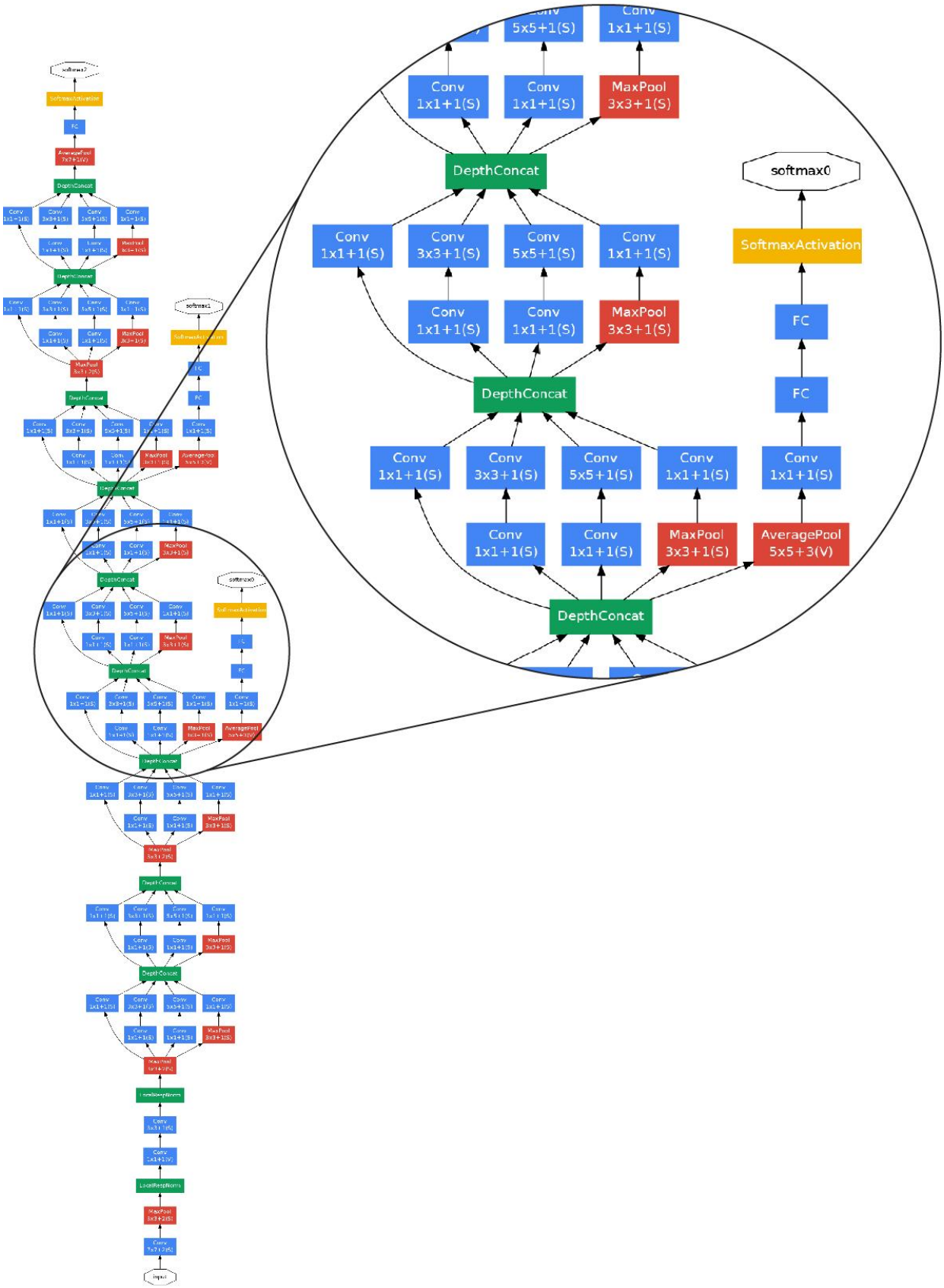

Figure 3.17 : *Architecture du GoogLeNet*

3.6.3.3 Résultats

L'ILSVRC 2014 impose le même set d'image, l'imagenet, que les précédentes compétitions. On en compte toujours 1,2 M image d'entrainement pour 1000 classes.

Team	Year	Place	Error (top-5)	Uses external data
SuperVision	2012	1st	16.4%	no
SuperVision	2012	1st	15.3%	Imagenet 22k
Clarifai	2013	1st	11.7%	no
Clarifai	2013	1st	11.2%	Imagenet 22k
MSRA	2014	3rd	7.35%	no
VGG	2014	2nd	7.32%	no
GoogLeNet	2014	1st	6.67%	no

Tableau 3.02: *Classification des performances*

La compétition utilise la mesure sur le top-5 de taux d'erreur, comparaison de la vraie prédiction contre les 5 premières classes prédites par le réseau, pour classer les participants. Le tableau 3.02 montre le classement entre les 3 premiers participants, les gagnants des précédentes années y sont aussi référés.

3.7 Techniques innovante

A la recherche de plus de précision, les études récentes se portent sur les problèmes qui limitent la généralisation des réseaux de neurone à convolution. Trois publications se distinguent par l'approche qu'ils portent : le fractionnal pooling et l'ELU (Exponential Linear Unit).

3.7.1 Fractional pooling

Les recherches se sont toujours concentrées à améliorer la couche de convolution avec des techniques comme l'utilisation de plusieurs filtres, un réseau très profond, le dropout. En comparaison, l'opération de pooling a été négligée. Il est de coutume d'utiliser un maxpooling de taille 2x2 appelé aussi MP2 (Max pooling de taille 2).Il a pour avantages d'être rapide, réduit rapidement la taille des dimensions, Toutefois, puisque la MP2 réduit rapidement la dimension dans

les couche cachée, il est nécessaire de mettre dos à dos 2 couches de convolution pour construire un réseau très profond.

La solution apporté par le FMP (Fractional max pooling) est de réduire la taille spatiale des images par un facteur α appartenant à] 1,2[. La région de pooling peut être choisie aléatoirement ou de façon pseudo aléatoire. [31]

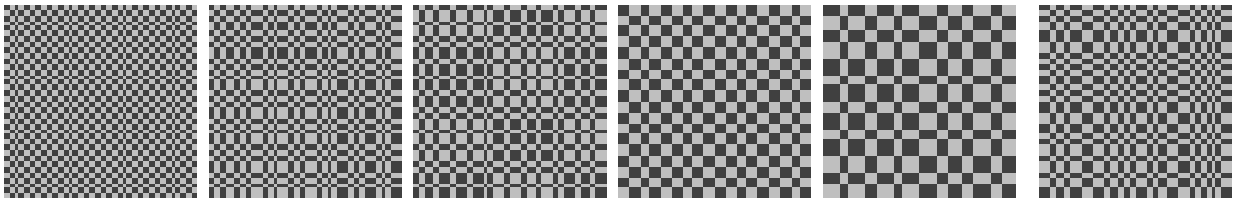

Figure 3.18 : *Exemple de région de pooling*

Les tests réalisés avec le FMP concluant sur l'apport sur la précision du réseau. En utilisant le set d'image CIFAR-10 ils ont obtenu un pourcentage d'erreur de 4.50%,

3.7.2 Exponential Linear Unit

Présenté lors de l'ICCV 2015 (International Conference on Computer Vision), ELU est une solution afin d'accélérer l'apprentissage. Comme le ReLU, ELU permet d'éviter la disparition de gradient. Au contraire du ReLU, ELU possède des valeurs négatives qui permet l'activation moyenne des unités de neurone proche de zéro. Cela permet d'accélérer l'apprentissage parce que le gradient est rapproché du graduent naturel. [38]

La fonction d'activation ELU est régit par la formule :

$$\mathrm{f(x)} = \begin{cases} x \; si \; x \geq 0 \\ \alpha(e^{(x)} - 1) \;\; si \; x \leq 0 \end{cases} \tag{3.29}$$

Sa dérivé s'obtient par :

$$\mathrm{f(x)} = \begin{cases} 1 \; si \;\; x \geq 0 \\ f(x) + \alpha \;\; si \; x \leq 0 \end{cases} \tag{3.30}$$

L'hyper paramètre α permet de contrôler la valeur auquel le ELU est saturé pou des entrées négatives. Les résultats expérimentaux montrent clairement que les réseaux utilisant l'ELU comme

fonction d'activation surpasse les réseaux avec ReLU. Ils ont aussi noté que cette technique est la plus active lorsque le nombre de couches cachés est supérieur à 5.

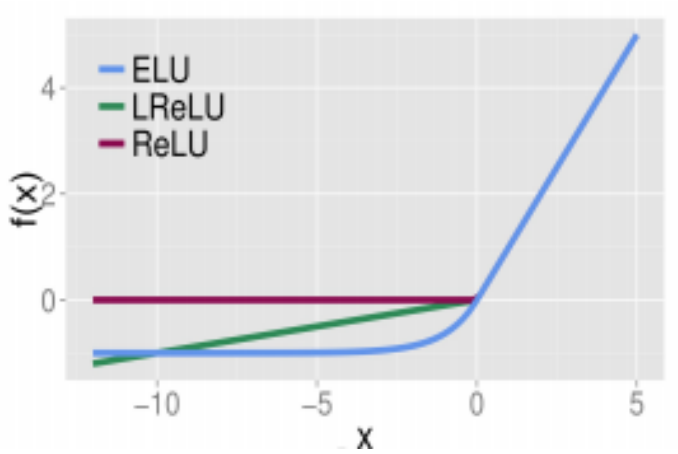

Figure 3.19 : *Comparaison entre les fonctions d'activation ELU, ReLU et LReLU*

3.8 Inconvénient des CNN

Même s'il est leader dans la classification d'images, le réseau de neurone à convolution présente certains inconvénients qui le paralysent. Les deux principaux facteurs limitant est la complexité et la lourdeur de calcul.

Le domaine de l'apprentissage automatique nécessite la maitrise de plusieurs domaines. Il est essentiel de détenir des connaissances avancées en probabilité et en algèbre linéaire. Diverses aptitudes comme la programmation en parallèle sur GPU (Graphical processor Unit) est aussi nécessaire, sans parler de l'intelligence artificielle, science cognitive,… etc., Cela limite le nombre de personnes pouvant porter leur contribution. [39]

Le plus grand problème est aussi la limitation matérielle. Les opérations de convolution sont très gourmandes en utilité de calcul. Plus le réseau est profond, plus le coût du calcul est élevé. En guise d'exemple, Le réseau AlexNet a été entrainé sur 2 GTX 580 disposant chacun 3GB de Ram ; l'apprentissage de l'architecture GoogLeNet nécessite de 2 GTX TITAN X avec 12GB de mémoire vive, un GTX TITAN X coutant aujourd'hui près de 2000$.

3.9 Conclusion

Inspiré du système visuel animal, le réseau de neurone à convolution dispose plusieurs couches de neurones à la fonction hiérarchisée. Ses caractéristiques comme le partage de paramètres et sa connectivité en fait un candidat idéal pour le traitement d'images. L'architecture d'un réseau à

convolution est composée d'empilement de couche de convolution et de pooling. Cette technique a permis au réseau de neurones de surpasser le système visuel humain, avec une précision de 96.6 %.

Toutefois cette technologie présente des inconvénients. Construire un réseau de neurone profond nécessite des connaissances dans plusieurs domaines et son entrainement requiert des matériels très chers.

CHAPITRE 4
CONCEPTION D'UN SYSTEME D'INDEXATION ET DE RECHERCHE TEXTUELLE D'IMAGES

4.1 Introduction

Nous avons déjà mentionné l'intérêt actuel porté sur les bases d'images, les techniques d'indexation et les techniques de recherche d'images dans de telles bases. Cependant, dans les applications pratiques de ces bases d'images que l'on retrouve aujourd'hui, que ce soit pour les chaînes de télévision, les journaux, les musées et mêmes pour les moteurs de recherche sur Internet qui proposent des solutions de recherche d'images, l'indexation et la recherche de ces images se font en se basant majoritairement sur des descripteurs visuel. Une méthode qui n'est pas souvent adaptée à l'exigence et à la nécessité de l'utilisateur.

Dans d'autre cas l'archivage des images et des séquences vidéo ne se fait qu'au prix d'une étape d'annotation manuelle à l'aide de mots-clés. Cette indexation représente une tâche longue et répétitive pour l'humain, surtout avec les bases d'images qui deviennent aujourd'hui de plus en plus grandes. De plus, cette tâche est très subjective à la culture, à la connaissance et aux sentiments de chaque personne. Cette technique offre des bons résultats, toutefois le facteur humain limite cette pratique.

L'approche proposée dans ce mémoire est de remplacer le travail effectué par l'homme dans l'indexation textuelle par un réseau de neurones à convolution. L'objectif de ce chapitre est donc d'étudier, de définir et de mettre en place un système d'indexation et de recherche textuelle d'image automatique par un réseau de neurones à convolution. Ce chapitre donne une vue plus détaillée sur les différentes techniques utilisées dans notre travail.

Le présent chapitre est divisé en deux parties qui sont les étapes nécessaires à la conception du système : En premier lieu, on va parler de la méthode d'indexation. L'architecture et les composants de notre réseau y seront décrits.

Ensuite on étudie la partie recherche du système telles que la récupération de la requête, la comparaison dans la base et la présentation.

4.2 Architecture du système

Nous utiliserons la même architecture que les techniques d'image. Reprenons la figure suivante :

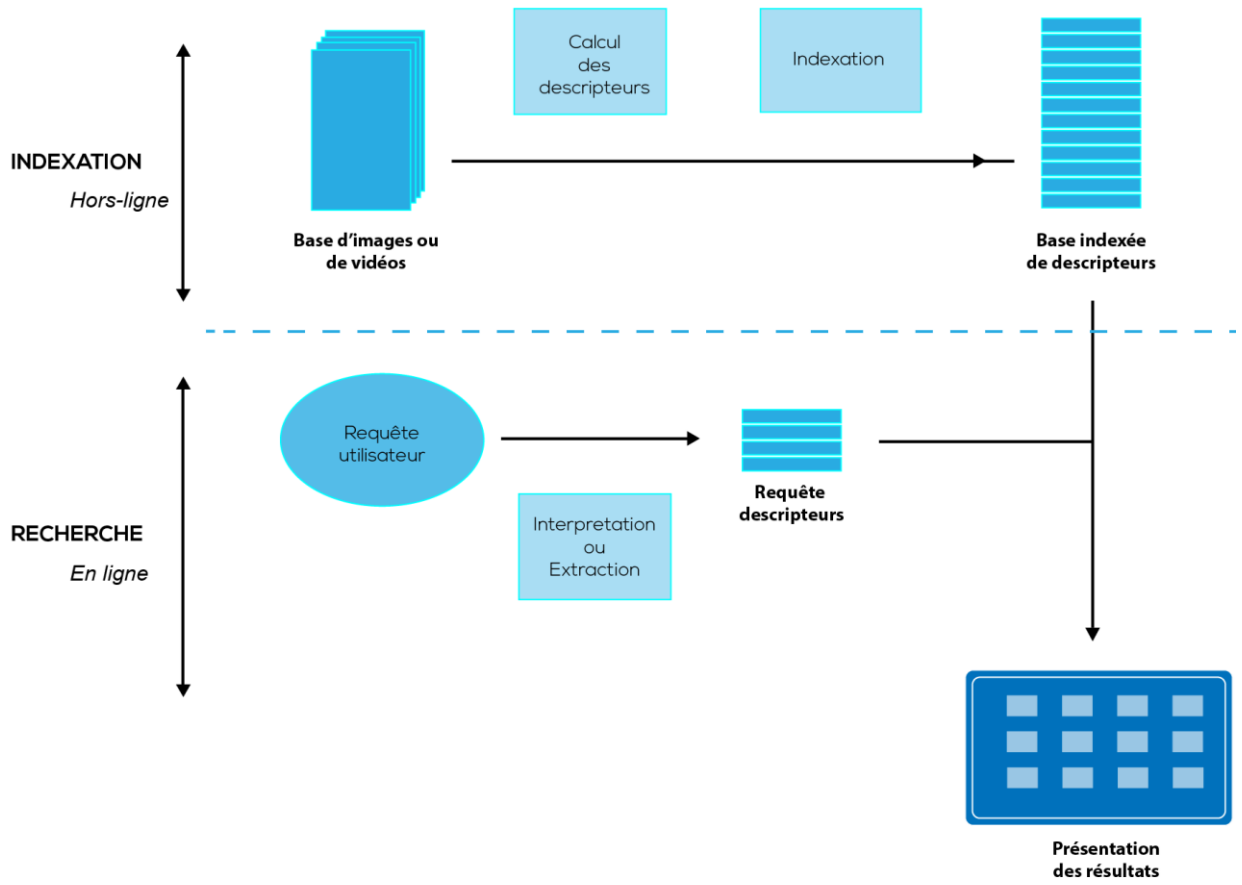

Figure 4.01 : *Architecture d'un système d'indexation et recherche d'image*

Le système se divise en deux parties distinctes, la partie indexation et la partie recherche. Dans la partie indexation, le logiciel va prendre chaque image dans la base et va inscrire l'index correspondant. Dans notre cas, ces indexes sont des ensemble de mots qui vont décrire le contenu de l'image. Ce travail est assuré par le réseau de neurones. Une base d'index contiendra alors l'ensemble de tous les index avec la référence de l'image correspondante. L'autre partie concerne la recherche d'image.[2]

Le problème consiste à extraite du requête de l'utilisateur les mots clés qui seront utilisés pour la comparaison avec la base d'index. Pour résoudre ce problème on utilise une méthode d'extraction de terme expliqué plus loin. Comme on a vu dans le premier chapitre il existe plusieurs méthodes

afin de comparer la requête et la base. Ne voulant pas s'attarder sur ce problème, on utilise la plus simple méthode qui consiste en une comparaison basée sur des opérations booléennes.

4.3 Conception de l'indexation textuelle

Cette partie est la plus importante du projet. Les paragraphes suivant décrivent le choix du système d'indexation et sa conception.

4.3.1 Contrainte du système

Afin de correspondre à notre attente, la méthode proposée doit satisfaire plusieurs critères vitaux à l'indexation

- Rapidité : nécessitant le traitement d'une base d'image en entier, qui peut contenir des milliers voir des millions d'images, le traitement d'une image doit se faire dans un temps acceptable, inférieur à 1s si possible.
- Précision : le système est conçu pour le remplacement de l'homme. La précision sur l'annotation des images doit avoisiner la capacité humaine, mesurée à près de 90% sur une base d'un million d'images.
- Légèreté en demande de ressources : afin de convenir à la majeure partie des ordinateurs, le système ne doit pas nécessiter trop de ressources.

4.3.2 Entrée/Sortie

Le système d'indexation prendra en entrée une image couleur, le format du couleur n'interférant pas au calcul d'un réseau de neurone à convolution, nous utiliserons le RGB. Le réseau de neurone nécessite avant son utilisation un apprentissage supervisé sur un set d'image, nous utiliserons le set d'images ILVCRC 2012 parce qu'il est libre à utiliser et contient 1.2 Million d'images classées en 1000 catégories. Les images originales de l'Imagenet sont des images en haute définition pesant au total 258Gb. Etant incapable de télécharger ce volume, nous utiliserons une version avec une taille de 256x256. Cela conviendra parfaitement à l'apprentissage du réseau. [40]

Figure 4.02 : *Fonctionnement de notre indexation textuelle*

Les 1000 catégories de l'ensemble du set d'image conviennent largement pour l'annotation d'une image. L'index sera donc un vecteur contenant 1000 valeurs qui correspond à la probabilité d'appartenances à chacune des classes. [40]

4.3.3 Le réseau de neurone à convolution

La conception d'un réseau de neurones à convolution ne suit pas une règle précise. Le choix des couches à utiliser et ses nombres n'est qu'arbitraire. C'est plus basé sur la recherche de performance.

Comme les initiateurs du « batch normalization », nous nous baserons sur cette architecture du réseau GoogLeNet pour la conception de notre système. Toutefois, plusieurs adaptations y seront apportées afin de correspondre à nos besoins.

4.3.3.1 L'importance de la profondeur

L'équipe du GoogLeNet ainsi que Simonyan, Karen, et Andrew Zisserman ont largement évoqué l'importance de la profondeur dans un réseau de neurones. Ces publications expliquent que plus il y a de couche de convolution, plus le réseau est performant. Cependant, l'empilement de plusieurs couches de convolution entraine une argumentation exponentielle du nombre de paramètres, ce qui entraine un besoin élevé en puissance de calcul lors de l'entrainement. [36][37]

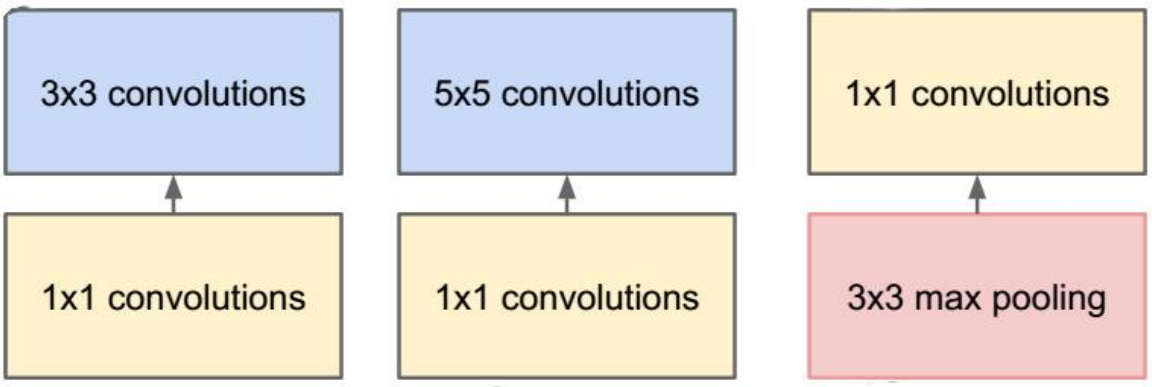

Figure 4.03 : *Couche de convolution utilisé dans le module inception*

La solution qu'ils ont trouvée est l'utilisation de convolution de petite taille : 5x5, 3x3, 1x1. Les convolutions de taille 5x5 et 3x3 sont excellentes pour l'extraction de descripteur sans pour autant affecter le calcul. Les convolutions de taille 1x1 interviennent encore plus la dimension avant l'entrée d'une autre couche de convolution. Elle est plus préférable aux couches de pooling qui sont jugées trop destructrices.

4.3.3.2 Les contraintes matérielles

Le plus gros inconvénient des réseaux de neurone à convolution est son besoin en puissance de calcul et mémoire. Actuellement, les bibliothèques d'apprentissage automatiques permet d'entrainer le réseau sur le GPU, ce qui rend cette phase 20x plus rapide que sur les CPU. Toutefois, entrainer un réseau tel le GoogLeNet nécessite près de 5Gb de mémoire vidéo, soit un GPU TITAN X, mais même avec ce type de GPU, l'apprentissage au complet en utilisant la base de données ILSVRC12 prend 2 semaines.

Il est difficile de formuler la quantité de mémoire et de calcul nécessaire pour un réseau de neurone à convolution parce que les architectures sont tous différents, et chaque Framework utilise le calcul de la convolution à sa façon. Toutefois, un utilisateur a montré qu'avec 4Gb de mémoire vidéo, on peut entrainer 16 couches. A l'occasion de ce mémoire, on dispose d'un GTX 660 qui possède 2Gb de ram. Avec une simple estimation, on peut entrainer un réseau à 8 couches sans problème.

4.3.3.3 Méthode de calcul de la dimension d'entrée et sortie

On peut calculer la taille du volume de sortie d'une couche de convolution par la formule : [25]

$$V = \frac{W - F + 2P}{s} + 1 \tag{4.01}$$

Si la sortie V calculée n'est pas un nombre entier, il y a incohérence entre les valeurs choisies. Cette erreur se traduit par le fait que le noyau de convolution ne s'ajustera à l'entrée. La simple façon d'y remédier est de modifier le zéros padding. Si on souhaite conserver la dimension de l'entrée, il faut calculer le zero-padding de la façon suivante :

$$P = \frac{F - 1}{2} \tag{4.02}$$

Ainsi, la couche de convolution :

- ❖ Accepte une entrée de volume :

$$W_1 \times H_1 \times D_1 \tag{4.03}$$

- ❖ Nécessite 4 hyper paramètre :

- Nombre de filtre K
- Taille d'un filtre F

- Le stride S
- La quantité de zero padding P

❖ Produit une sortie de volume :

$$W_3 \times H_2 \times D_2 \quad (4.04)$$

- $W_3 = \frac{W_1 - F + 2P}{S} + 1$
- $H_3 = \frac{H_1 - F + 2P}{S} + 1$
- $W_3 = K$

La couche dispose de $F.F.D_1$ poids par filtre, donc un total de $(F.F.D_1)K$ poids et K bias

Le volume de sortie de la couche de pooling se calcule de la même façon :

Accepte une entrée de volume :

$$W_1 \times H_1 \times D_1 \quad (4.05)$$

Nécessite 2 hyper paramètres:

- Taille d'un filtre F
- Le stride S

Produit une sortie de volume :

$$W_3 \times H_2 \times D_2 \quad (4.06)$$

- $W_2 = \frac{W_1 - F}{S} + 1$
- $H_2 = \frac{H_1 - F}{S} + 1$
- $D_2 = D_1$

On utilise souvent un pooling 3x3 avec overlapping soit F = 3, S=2, ou le pooling de 2x2 F = 2 S=2

4.3.3.4 Architecture

La conception d'un architecture de réseau de neurone à convolution se fait à partir de l'entrée et en calculant la sortie d'une couche qui prend en entrée la sortie de la couche précédente. Afin de faire correspondre l'entrée et la sortie des couches il a fallu réduire l'entrée du réseau à 229x229.

Couche	Entrée	F	S	K	P	Sortie
Convolution+ReLU	229x229x3	7	2	64		112x112x64
Maxpooling	112x112x64	3	2			56x56x64
Convolution+ReLU	56x56x64	3	1	128	1	56x56x128
Maxpooling	56x56x64	2	2			28x28x128
Convolution+ReLU	28x28x128	3	1	256	1	28x28x256
Maxpooling	28x28x256	2	2			14x14x256
Convolution+ReLU	14x14x256	3	1	512	1	14x14x512
Maxpooling	14x14x512	2	2			7x7x512
Convolution+ReLU	7x7x512	3	1	1024		7x7x1024
Average pooling		7	7			1x1x1024
FullConnect	1x1x1024			1000		1x1x1000
Softmax	1x1x1000			1000		1x1x1000

Tableau 4.01: *Architecture proposé*

W volume de l'entrée, **S** le stride, **F** taille du filtre, **P** nombre de zero-padding, **K** nombre de filtre

On s'est gardé d'utiliser le plus possible des couche de convolution de taille 3x3, avec de petite stride, tiré du GoogLeNet ainsi on pourra empiler beaucoup plus de filtre.

Comme on l'a vue dans le précèdent chapitre, l'utilisation du ELU est préférable au ReLU, cependant c'est encore un concept récent, on ne trouve pas encore son implémentation dans les Framework d'apprentissage automatique.

Noter l'utilisation de l'average pooling au lieu d'une couche à connexion complète, ceci est tiré de la publication de Szegedy selon laquelle l'utilisation de plusieurs couches de classification n'est utile qu'en manque de donnée d'apprentissage. L'enlèvement de ces couches réduit les paramètres à apprendre.

La couche Softmax ne fait que mapper les sorties du réseau vers une autre probabiliste. Le vecteur construit par le nôtre réseau contient ainsi les probabilités d'appartenance de l'image à chaque classe.

4.3.3.5 Apprentissage

En comparaison au réseau AlexNet, notre réseau est composé de moins de paramètre avec plus de profondeur. Toutefois on dispose de trop de paramètre par rapport aux données d'apprentissage, le réseau est sujet à l'overfitiing. Les mesures suivant ont été prose pour éviter ce désagrément :

- Utilisation du dropout : le processus se fera au niveau du classificateur, juste après l'average pooling. On utilise une quantité de 40%, recommandé dans la publication originale.
- La donnée d'apprentissage est augmenté de 50% en utilisant la méthode de data augmentation.

L'apprentissage se fera par SGD (Stochastic Gradient Descent) avec une batch de taille 128. On utilise un momentum de 0.9 recommandé dans la publication sur l'importance du momentum par I. Sutskever et son équipe.[41] [42]

4.4 Conception de la recherche

La partie recherche consiste à comparer la requête de l'utilisateur à la base de données. Les images correspondantes aux critères définis sont renvoyées en guise de résultat.

4.4.1 Méthode de comparaison

On a vu au premier chapitre qu'il y avait plusieurs méthodes pour comparer les images et la requête. Ne voulant pas trop s'attarder sur ce sujet, on s'est décidé à utiliser la comparaison booléenne. La requête prendra donc la forme : Chien ET Chat, Voiture OU Avion.

La nature probabiliste du vecteur de sortie nous permet d'améliorer la recherche en utilisant un paramètre de tolérance à la requête. [3][4]

4.4.2 Requête par phrase

L'indexation textuelle a le grand avantage de faciliter l'implémentation d'une recherche sémantique. Afin d'illustrer ce propos, on a utilisé une forme simple de recherche sémantique en utilisant une technique d'extraction de terme pour trouver les mots clé dans la requête. L'utilisateur pourra ainsi formuler ses requêtes en utilisant des phrases.

La recherche sémantique peut être plus performante en utilisant un réseau de neurone artificiel comme exécuteur de la requête. Il prendra ainsi en entrée une phrase et extrait une requete compréhensible à la méthode de comparaison. Un système plus avancé ajoutera directement cette

couche à la comparaison d'image. Un tel système pourra prendre une phrase en entrée et procurer une liste d'image. [6][7]

4.4.3 Recherche par l'exemple

La forme particulière de notre système nous offre l'avantage d'utiliser en plus une image comme requête. Dans ce cas-là, le système en entier est semblable au CBIR classique, et notre vecteur de classification fait guise d'index visuel.

La comparaison de l'index se fera comme dans l'indexation basé sur le contenue en comparant la distance entre l'index de l'image de requête et les images dans la base. [2][11][13]

4.5 Conclusion

Dans ce chapitre nous avons conçu un système d'indexation textuelle automatique et de la recherche correspondant. Nous avons discuté des différentes contraintes auquel ce genre de système doit se plier. Nous y avons été étudiés l'architecture d'un réseau de neurone à convolution qui va remplacer le travail manuel dans la méthode d'indexation textuelle. La précision d'un réseau croît en fonction de la profondeur de la profondeur cependant avec cela croit le nombre de calcul affaire et la quantité de mémoire nécessaire à l'apprentissage. Nous avons alors conçue l'architecture en fonction du matériel dont nous disposant. Le côté recherche de notre système s'est limité à l'utilisation d'opérateur logique. Toutefois, notre système offre de nouvelle perspective à des techniques plus bien plus performantes.

CHAPITRE 5
SIMULATION DU SYSTEME D'INDEXATION ET DE RECHERCHE D'IMAGE PAR UN CNN

5.1 Introduction

La simulation reste la meilleure façon de tester de nouvelle architecture et de nouveau système. Il y a quelques années les logiciels de simulation de réseau de neurone étaient orientés à des fins professionnelles et leurs accès étaient limités, comme par exemple le DistBelief de Google qu'on ne trouve que dans son laboratoire. Convaincu par les avancé que cette technologie peuvent offrir, les grandes firmes et les universités leader ont rendu ouvert l'accès au logiciel et à leurs enseignements.

Ce chapitre décrit donc la simulation du système d'indexation textuelle et recherche d'image par réseau de neurone à convolution ; Il consiste en premier lieu à l'apprentissage et la validation de l'architecture du réseau de neurone proposé. La suite aborde les différents tests de performance du système d'indexation et de recherche lui-même.

5.2 Outils de simulation

Dans le but de mener à bien l'expérience, nous avons besoin d'outil robuste et pratique. Les logiciels et Framework utilisés sont ceux des grands laboratoires de recherche.

5.2.1 Python

Python est un langage de programmation de haut niveau créée par Guido van Rossum, il est devenu très populaire grâce à sa simplicité et la facilité de lecture de ses codes. Il permet au programmeur d'exprimer ses idées en quelques lignes sans sacrifier sa facilité de lecture. Coté syntaxe Python a été conçu pour être lisible et clair. C'est un langage très structuré, l'indentation représente les blocs représentés par des accolades pour des langages comme le C. En Python, tout est objet, les fonctions, les types de base et même les classes sont des objets.

Un des forces de python est la possibilité d'utiliser un autre langage comme de C par l'intermédiaire d'un conteneur. Ceci permet d'utiliser des Framework écrit en d'autre langage et ainsi profiter des avantages propres à ces langages, par exemple la rapidité du C++.

Python est très utilisé dans le monde scientifique, il dispose d'outil de calcul numérique et de gestion de tableau multidimensionnel très performant (numpy) ainsi que des outils de création de courbe et

de visionnage. Il est même plus préférable au logiciel Matlab qui est payant et son module d'apprentissage automatique n'est pas très optimisé pour les réseaux de neurone à convolution.

Fonction factorielle en Python

```
def factorielle (x)
    if x>2:
        return 1
    else:
        return x*factorielle(x-1)
```

Figure 5.01 : *Syntaxe de python*

5.2.2 CUDA

Compute Unified Device Architecture ou CUDA est une plateforme de programmation parallèle développée par Nvidia. Il permet de décupler la puissance de calcul du système en exploitant la puissance des processeurs graphiques (GPU).

L'utilisation des processeurs graphique offre une vitesse de calcul jusqu'à 20 fois plus rapide que les CPU. Il est très apprécié pour le calcul numérique notamment pour l'apprentissage des réseaux de neurone. Le développement de la bibliothèque CuDNN par NVIDIA a boosté la vitesse d'entrainement des réseaux de neurone à convolution. L'inconvénient c'est que la dernière version de CuDNN n'est compatible qu'avec des GPU NVIDIA avec un 'compute capability' ≥ 3, soit les récent carte graphique à partir de la série de GTX 600. Nous allons profiter de cette technologie pour notre simulation.

5.2.3 Traitement d'image : OpenCV

L'Open Computer Vision(OpenCV) a débuté dans en 1999 dans les laboratoires d'Intel par Gary Bradsky. Plus tard son développement actif a continué avec le support de Willow Garage. Aujourd'hui, OpenCV intègre beaucoup d'algorithmes de vision par ordinateur ainsi que l'apprentissage automatique.

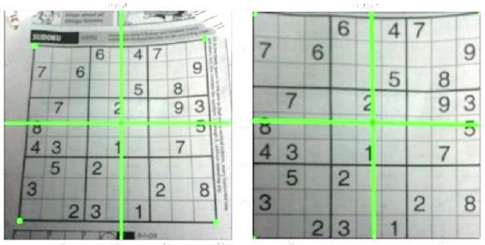

Figure 5.02 : *Exemple de traitement sur OpenCV : transformation de perspective*

Actuellement Open CV supporte une grande variété de langage de programmation tel que C++, JAVA, Python. Son implémentation sur python utilise un conteneur pour le code en C++. Le support de numpy rend les taches plus faciles. Dans notre simulation, nous l'utilisons pour le prétraitement des images : réduction, modifications diverse.

5.2.4 Framework d'apprentissage automatique

L'apprentissage automatique nécessite des calculs très complexes avec de grandes quantités de données. Les logiciels d'apprentissage automatique ne sont en fait que des logiciels optimisés pour le calcul numérique et la gestion de tableau multidimensionnel. Il existe plusieurs Framework très utilisés, on peut en citer Torch, Theano, Caffe et TensorFlow. Chacun possède sa force et sa faiblaisse, le choix se fait en étudiant la fonctionnalité qui nous est plus utile et lequel faire un compromis.

5.2.4.1 Torch

Torch est un Framework de calcul scientifique open source. Il procure plusieurs algorithmes d'apprentissage automatique facile à implémenter. Les codes sont exprimés en langage de script rapide LuaJIT, combiné à une implémentation en C. Il support aussi l'apprentissage sur GPU. Ce framework est très utilisé dans les laboratoires de Google et Facebook.

Par rapport à ses concurrent, Torch est le plus rapide, toutefois, tout n'est pas dans la rapidité, le critère de choix concerne aussi sur l'accessibilité. L'inconvénient de ce Framework est le script Lua, l'utilisateur devra apprendre ce langage avant de pouvoir utiliser l'outil. Le code écrit en Lua est très difficile à débuguer et les retours fournis par Torch sont souvent incompréhensible.

5.2.4.2 Theano

Theano est une bibliothèque de calcul mathématique sur python issue de l'université de Montréal. A proprement dit, Theano n'est pas une Framework de « machine learning » mais un outil mathématique qui rend la conception de modèle facile grâce à d'autre module.

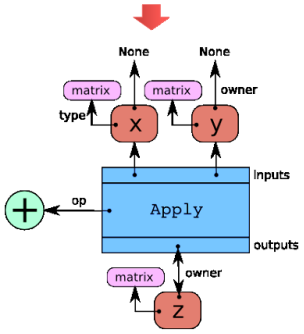

Figure 5.03 : *Graphe d'utilisation de Theano*

Pylearn2 est un module python pour la « machine learning » qui se base sur Theano. Il supporte le deep learning, les réseaux de neurone à convolution et aussi le calcul sur GPU. Le couple Theano/Pylearn2 plus adapté pour les recherches académiques. Son programmation sur Python permet de profiter de la clarté de ce dernier, on peut aussi associer plusieurs Framework sans grande difficulté.

5.2.4.3 Caffe

Caffe est développé par le BVLC (Berkeley Vision and Learning Center) et crée par Yangqing Jua comme projet lors de son PhD à l'université de Berkeley. Il offre les avantages suivant :

- Architecture expressive qui encourage l'application et l'innovation. La conception d'une architecture se fait par configuration et évite le codage bas niveau. Le passage entre l'utilisation d'un CPU et d'un GPU se fait par le changement d'un seul paramètre.
- Rapidité : Caffe est le plus rapide en termes de réseau de neurone à convolution.
- Accessible : Caffe est écrit en C++ et dispose d'un module python très pratique.

Nous utiliserons ce Framework dans sa version python. Ainsi nous pouvons le combiner avec d'autre outil : OpenCv, matplotlib pour le traçage de courbe, etc.

5.3 Implémentation

L'implémentation d'un modèle sur caffe se fait par un fichier de configuration .prototxt qui est passé en paramètre pour la création d'un réseau.Tous les hyper paramètres et la nature des couches sont décrit dans ce fichier.

L'apprentissage du réseau se fait par le biais d'un "solver" dans caffe. Le solver se charge d'effectuer la propagation en avant, en arrière et la mise à jour des poids. Il y a plusieurs types de solver dans python : SGD pour la descente de gradient Stochastique, AdaDelta, Adaptive Gradient, RMSprop. Les paramètres du Solver sont aussi décrits dans un fichier de configuration .prototxt .

Nous utilisons le SGD avec les paramètres suivants :

- base_lr : 0.01 #vitesse d'apprentissage initiale
- lr_policy : ''step '' # change la vitesse d'apprentissage par un facteur gamma à chaque étape
- gamma : 0.1
- stepsize : 100000 # nombre de l'itération par étape
- momentum : 0.9

Le début de notre fichier de configuration ressemble à :

```
name : ''IndexNet''
input :  ''data''
input_dim :10
input_dim:3
input_dim:224
input_dim:224
layers {
    bottom:  ''data"
    top:  ''conv1_1''
    name:  ''conv1_1''
     type: CONVOLUTION
    convolution_param  {
        num_output: 64
```

5.4 Méthode de mesure du système

Plusieurs critères permettent d'évaluer la performance de notre système. Nous évaluons indépendamment le réseau de neurone à convolution et le système de recherche.

5.4.1 Evaluation du réseau de neurone

Le réseau de neurone s'évalue en calculant sa précision de classification. Cela est exprimé par l'erreur de classification du top-1 : la mesure se fait sur le premier label prédit par le réseau et le top-5 : on prend les 5 premières classes prédites.

Un set d'image à part le set d'apprentissage est utilisé pour la phase d'évaluation. Ces images sont nouvelles au système car elles ne font pas partie du lot d'apprentissage.

$$E = \frac{Nombre\ de\ classe\ exact}{Nombre\ d'image\ testé} \qquad (5.01)$$

5.4.2 Evaluation du système de recherche

Les mesures les plus courantes pour évaluer un système sont le temps de réponse et l'espace utilisé. Plus le temps de réponse est court, plus l'espace utilisé est petit, et plus le système est considéré bon. Mais avec des systèmes qui ont été faits pour la recherche d'informations, en plus de ces deux mesures, on s'intéresse à d'autres mesures. Dans le système de recherche d'informations, l'utilisateur s'intéresse aux réponses pertinentes du système. Donc les systèmes de recherche d'informations exigent l'évaluation de la précision de la réponse. Ce type d'évaluation est considéré comme l'évaluation des performances de recherche. Le système d'indexation et de recherche d'images est un système de recherche d'informations. Dans les systèmes de recherche d'images, les auteurs ont souvent utilisé les mesures d'évaluation que l'on a utilisées pour évaluer des systèmes de recherche d'informations.

5.4.2.1 Mesure de l'invariance

Ce test consiste à mesurer la tolérance du système face à plusieurs transformations de l'image. Une image est d'abord utilisée en requête, on note les résultats fournis par la recherche. Ensuite on applique une déformation à l'image de requête et on observe le résultat de recherche. Le processus est répété autant de fois que le nombre de transformation à étudier, dont les principaux sont :[3][10]

- Recadrage
- Rotation
- Ecriture
- Changement de luminosité

Rappel et précision

a. Le rappel:

Le rappel est le rapport entre le nombre d'images pertinentes dans l'ensemble des images trouvées et le nombre d'images pertinentes dans la base d'images. [2][8][10]

$$Rappel = \frac{|R_a|}{|R|} \tag{5.02}$$

$|R_a|$ est le nombre d'images pertinent dans l'ensemble réponse

$|R|$ est le nombre d'images pertinentes dans la base d'images

b. La précision :

La précision est le rapport entre le nombre d'images pertinentes dans l'ensemble des images trouvées et le nombre d'images trouvées. [2][8][10]

$$Precision = \frac{|R_a|}{|A|} \tag{5.03}$$

$|A|$est le nombre d'images dans l'ensemble de réponses

c. La courbe de rappel et précision

Le rappel et la précision sont les mesures importantes, mais si on voit seulement une paire de valeurs de rappel et précision, cette paire de valeurs ne peut pas indiquer la performance du système. C'est pourquoi on donne souvent une distribution de rappel et précision sous en forme de courbe.

La figure 5.04 donne un exemple de courbe de rappel et précision. Pour dessiner cette courbe, on doit calculer plusieurs paires de rappel et précision et les interpoler [2][8][10]

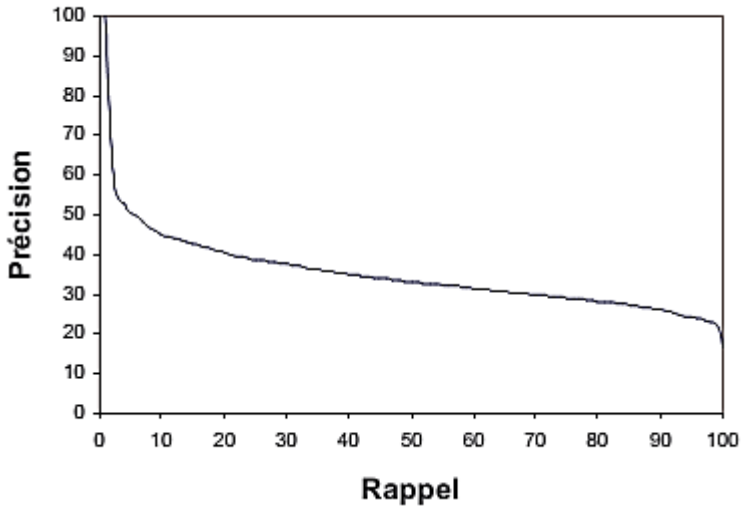

Figure 5.04 : *Exemple de courbe de rappel et précision*

En pratique, on utilise plusieurs requêtes. Dans ces cas pour évaluer un système, on calcule la précision moyenne pour toutes les requêtes correspondant à chaque niveau de rappel :

$$\overline{P(r)} = \sum_{i=1}^{N_q} \frac{P_i(r)}{N_q}$$

5.5 Résultats et interprétations

Le temps d'apprentissage a été plus long par rapport à celui que nous avons pensé. Par souci de sécurité de matériels, l'entrainement a été arrêté plus tôt que prévu. Toutefois nous allons tester le système proposé via les méthodes cité ci-dessus. Le CNN a été préalablement testé en prenant compte du problème lors de l'apprentissage avant de l'intégrer dans le système de recherche.

5.5.1 Le réseau de neurone à convolution

L'entrainement du réseau de neurone a durée quatre jours pour un avancement de 75%. Puisqu'on a pu entrainer le réseau jusqu'à ce stade, la demande en mémoire vidéo a été respectée par le GPU mais sa vitesse de calcul n'est pas assez rapide, causant alors un ralentissement de l'apprentissage .L'idéale serait donc de réduire le nombre de couches et le nombre de données d'apprentissage afin de s'adapter à nos matériels.

a. Apprentissage

La courbe illustrée sur la figure 5.05 montre l'évolution de l'erreur de prédiction du réseau pendant la phase d'apprentissage.

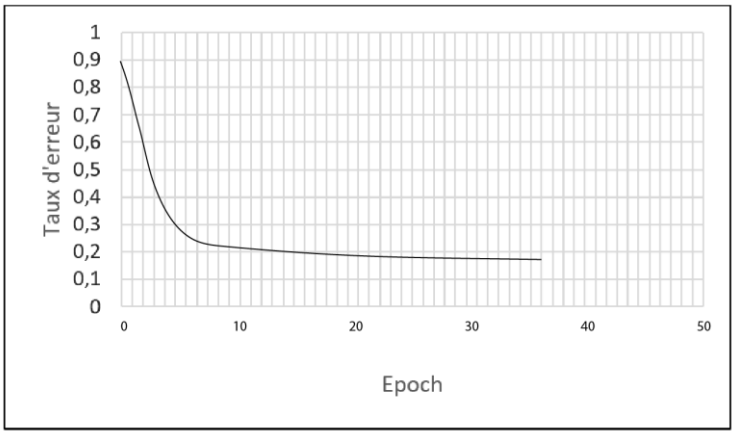

Figure 5.05 : *Courbe de l'erreur de prédiction*

On constate que l'apprentissage a été arrêté prématurément avant de trouver la valeur minimale de l'erreur. Ceci altère la précision du réseau de neurone mais offre une plus grande généralisation.

L'utilisation du mometum a permis d'adapter la vitesse d'apprentissage à la progression. On aperçoit cela à la brusque descente mesurée en début d'apprentissage. [34][35][41]

b. Précision du réseau

Le tableau ci-dessous compare la précision mesurée sur le réseau par rapport à l'état de l'art :

Architecture	Top-1	Top-5
Alexenet	40.7	18.2
GoogleNet		6.67
Notre architecture	37.01	16.12

Tableau 5.01: *Comparaison des erreurs de prédiction*

Avec moins de paramètres et de temps d'apprentissage, notre réseau surpasse le modèle AlexeNet. Cette différence est favorisée par l'utilisation des couches de convolution de petite taille, généralement des 3x3 contre des 11x11 dans AlexNet. En comparant les résultats à ***GoogLeNet,*** notre architecture est dépassé ; le module Inception est très efficace en terme d'extraction de descripteurs Quoique les résultats sont loin derrière le GoogLeNet, notre architecture donne des résultats acceptable vu le nombre de couche et le matériel utilisé. Evaluation du système de recherche proposé. [36][37]

Les mesures ont été effectuées par le biais d'une interface conçue pour l'occasion. L'interface imite une utilisation réelle de l'indexation et de recherche. Puisque l'évaluation de la pertinence d'une image ne peut se faire que par un humain, il est impossible d'automatiser la tâche. A cause de cela le nombre d'images dans la base et les classes à évaluer ont été réduits.

Figure 5.06 : *Photo d'un chat utilisé pour les tests d'invariance*

5.5.2 Test d'invariance à certaine transformation de l'entrée

Dans ce test, on utilise le mode de recherche par l'exemple. On effectue d'abord une requête avec l'image originale et on note les résultats. Ensuite, des versions modifiées de cette image ont été utilisées.

Figure 5.07 : *Images de test résultant de différents transformations*

La figure 5.06 montre l'image de référence auquel on applique des transformations spatiale : rotation de 90 et de 180 ° et aussi une translation, on a aussi caché une partie de l'image. La dernière modification est l'ajout d'un filtre de température élevé.

Les 44 premiers résultats de la recherche par l'image originale sont représentés sur la figure suivante.

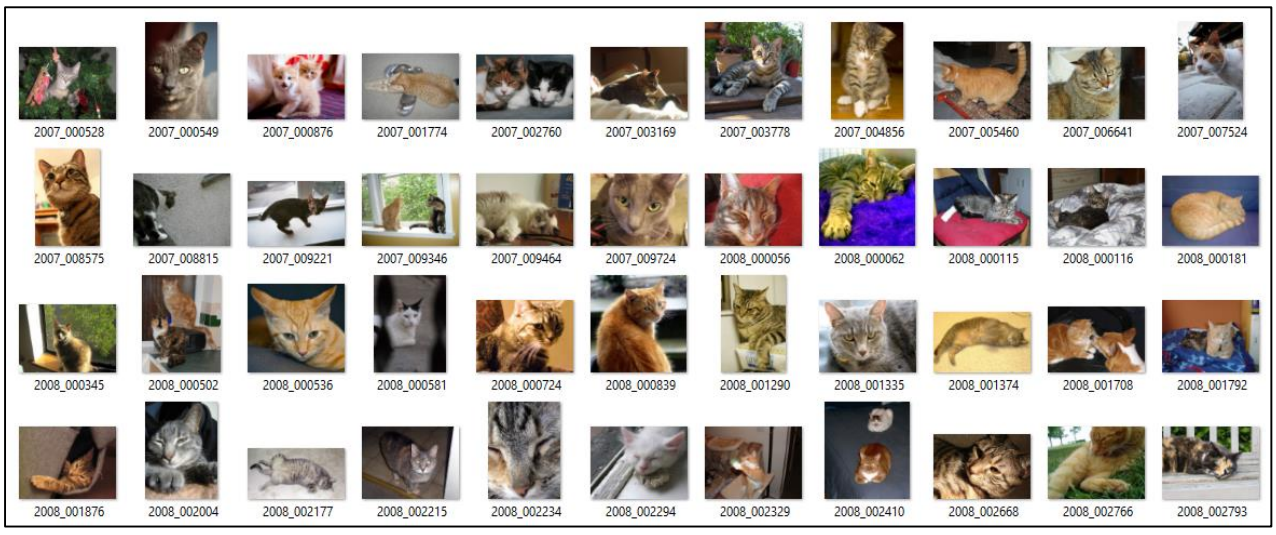

Figure 5.08 : *40 premiers résultats de la recherche par l'image originale*

On constate qu'en utilisant une image de chat comme requête, on obtient biens des images de chat en retour. On peut déjà observer que le système ne prend pas en compte les caractéristiques de bas niveau telle que la couleur, la texture. Quoiqu'il y ait des erreurs, le système connait vraiment les traits caractéristiques d'un chat.

Les transformations : rotation de 90° et masquage de l'image produisent les mêmes résultats que la requête par l'image originale. On peut en tirer que le système peut reconnaitre des formes même si cette forme est très petite ou partiellement incomplète. Ceci est dû par les champs de réception dans les premières couches du réseau de neurone à convolution. L'utilisation du Pooling favorise aussi ces invariances, comme nous l'avons évoqué au chapitre précédent.

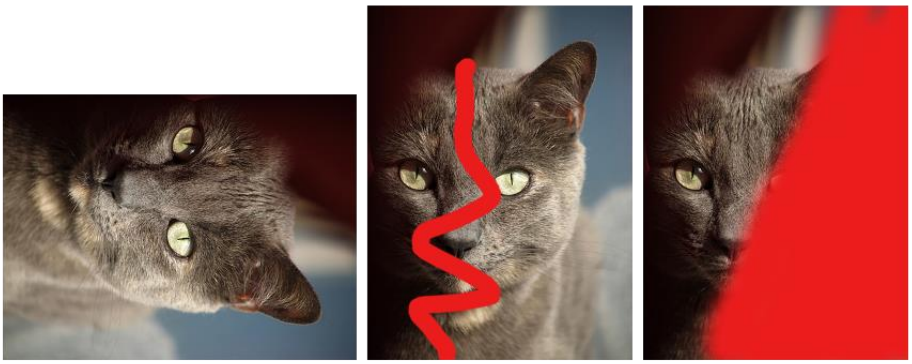

Figure 5.09 : *Les transformations auquel le système est invariant*

La rotation de 180 ° a produit des résultats très médiocres. On a obtenu très peu d'images en retour, mais aussi elles appartiennent à des classes très différentes. Il semble que le système n'identifie plus l'image comme étant un chat lorsque celle-ci est retournée.

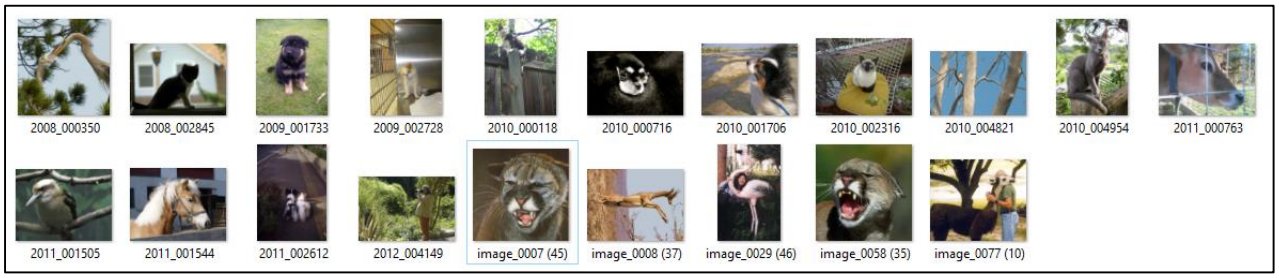

Figure 5.10 : *Résultat de la rotation de 180°*

Ceci montre une limite des réseaux de neurone à convolution. On peut attribuer cette erreur à l'absence de photos renversées dans le lot d'apprentissage du réseau. Même si l'on a utilisé la méthode d'augmentation de données, cela a été insuffisant pour limiter la dépendance du système au lot d'apprentissage.

Le test par l'image avec un filtre suit le même chemin que le précédent. Les images retournées ne disposent d'aucun point commun connu. Ce problème est toujours dû à la dépendance au lot d'apprentissage. [32]

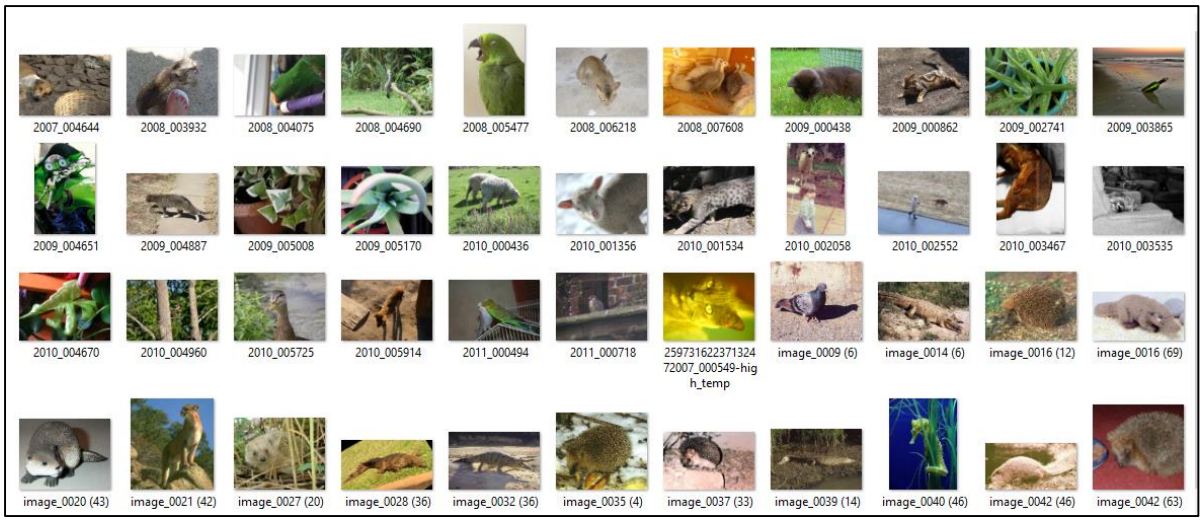

Figure 5.11 : *Images retournées en utilisant comme requête l'image modifié par un filtre*

On peut tirer de ces tests que les réseaux de neurone à convolution sont très performants pour la classification. Il nécessite cependant de disposer de nombreuses images pour la phase d'apprentissage. Si on manque de données, la méthode du « data augmentation » est recommandée.

5.5.3 Courbe de rappel et de précision

Le comptage des résultats afin de tracer la courbe de rappel et de précision doit se faire manuellement. Nous avons donc limité la base d'image à près de 400 images reparties en 10 classes de taille variables : 47 Ours, 50 oiseaux, 80 chats, 57 chiens, 8 motos, 50 avions, 32 bateaux, 52 tigres et 35 trains. Nous avons tracé 2 différentes courbes correspondantes à la recherche textuelle et à la requête par image. Chaque courbe est issue de 2 requêtes par classes d'images soit au total 20 par types de recherche.

Recherche textuelle :

La recherche textuelle s'est fait par le biais de mot clé comme : chien, chat, avion. Nous avons ainsi obtenu la courbe représentée sur la figure 5.12.

On constate que le système offre des performances médiocres à partir du rappel de 0.68. Le système ne semble pas aussi retourner que 85% des images contenue dans la base de données. On peut expliquer cela par le fait que les étiquettes utilisé pour l'apprentissage sont mal catégorisé. On retrouve par exemple 5 différentes classes pour le type 'chien'. Les images sont alors reparties sur ces classes, faussant le résultat de la recherche. Malgré cela le système est très performant et est indifférent des descripteurs de bas niveau comme les couleurs, la texture.

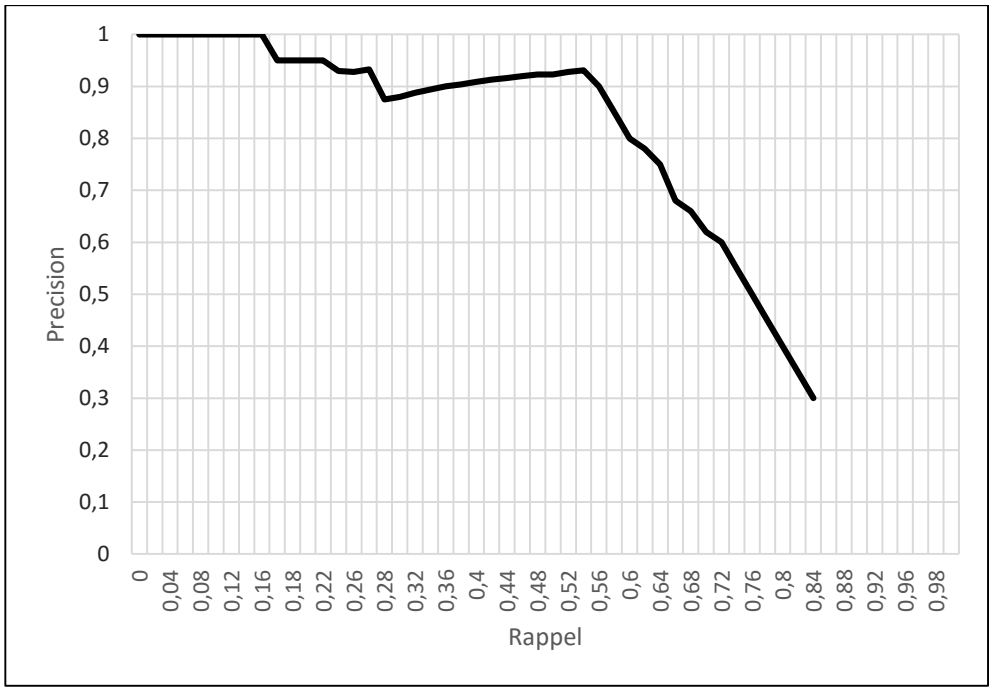

Figure 5.12 : *Courbe de rappel et précision de la recherche textuelle*

Requête par image :

Dans le test de requête par image, on a deux fois testé les catégories énoncées plus haut. Par exemple pour rechercher les images d'oiseau dans la base on a utilisé la photo sur la figure 5.10.

Figure 5.13 : *Exemple d'image requête pour l'oiseau*

Notre système de recherche est dans ce domaine très performant, il a en général retourné toutes les images dans la base avec un taux élevé de précision. On a d'autre part remarqué que la taille de l'image en entrée n'influe sur la classification de ce dernier.

Les cas où la recherche est défaillante concernent les recherches sur des classes d'une même catégorie séparée, par exemple pour la locomotive à vapeur et la locomotive électrique. Le problème

en concerne donc que l'exploitation des indexes et non le réseau de neurones. La courbe de rappelle/précision (figure 5.13) tiré des résultats des requête montre la haute précision de notre système d'indexation et de recherche, la précision oscille entre 0.9 et 1. Et même lorsque la plupart des images pertinentes sont tirées de la base, la précision du système n'est pas diminuée.

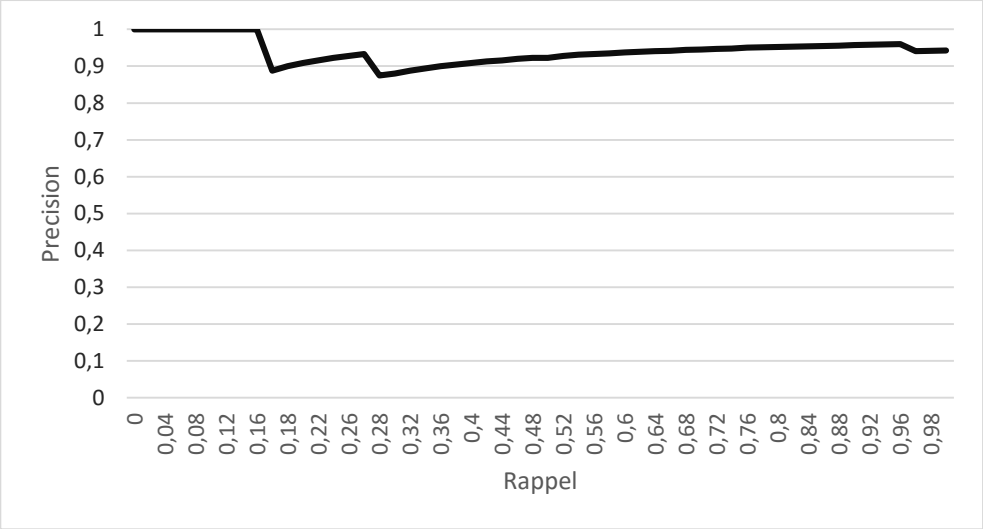

Figure 5.14 : *Courbe de rappel/précision pour la requête par image*

On voit ici la performance de classification du réseau de neurone à convolution. Nous avons obtenue plus de précision puisque la traduction du texte de l'utilisateur en requête dans la base est ici réduite en une instruction mieux compréhensible par le système (Les classe avec des probabilités élevé).

5.6 Conclusion

Dans ce chapitre nous avons simulé et entrainé le réseau de neurone à convolution sur le Framework Caffe. Malgré le matériel utilisé on a obtenu un réseau exploitable permettant d'étudier l'indexation textuelle.

Le test utilisé concerne les caractéristiques d'invariance du système et sa performance. Le système est prouvé invariant au masquage d'une partie de l'image, à la rotation et à la translation ; L'indexation ne distingue cependant pas les images auxquelles on a appliqué des filtres, et celles totalement retournées. Cela indique une forte dépendance aux images d'apprentissage. Les courbes de rappel/précision tracé montrent que notre système est très performant. Les cas de défaillance constatés sont attribués au problème de catégorisation des labels utilisés.

On peut en tirer que l'indexation textuelle par réseau de neurone à convolution offre des résultats très prometteuse. Sa limitation concerne le coût de matériel informatique nécessaire pour l'apprentissage principalement la carte graphique.

CONCLUSION GENERALE

L'augmentation exponentielle de la quantité de documents générés par les utilisateurs des systèmes informatiques nécessite une organisation efficace des données. C'est dans ce cadre que s'inscrivent l'indexation et la recherche d'image. Il existe principalement deux types d'indexation d'image, l'indexation textuelle et l'indexation basé sur des descripteurs. La première méthode a été négligé ces dernières années vue la nécessité de l'intervention humaine. Le sujet de ce mémoire vise à remplacer ce facteur humain par un réseau de neurone à convolution.

Un réseau de neurone artificiel est un variant des neurones biologique assemblé pour résoudre un problème donnée. Ils sont initiés par les travaux de Mc Culloch sur le neurone formel. Avant son utilisation un réseau de neurone doit être entrainé pour ajuster ses paramètres internes. La phase d'entrainement se fait souvent en présentant à l'entrée et à la sortie un couple exemple, les poids sont ensuite calculés par la méthode de descente de gradient et la propagation en arrière.

Un réseau de neurone à convolution utilisé dans ce mémoire est un réseau de neurone qui imite cortex visuel animal. Il dispose de champ visuel et de différents types de cellule disposée en plusieurs couches, cela lui permet de représenter les informations sous forme hiérarchisé. L'avantage de l'utilisation de ces types de réseau est son performance en traitement d'image cependant l'apprentissage nécessite d'énorme capacité de calcul et un grand base de donnée. Comme les réseaux de neurone standard, les réseaux de neurone à convolution sont entrainés par descente de gradient et la propagation en arrière. Il existe toutefois plusieurs améliorations propres à ces réseaux. Les architectures de référence sont ceux du GoogLeNet, AlexNet et le LeNet-5.

Le système d'indexation proposé compote un réseau de neurone à convolution qui dispose de 12 couches cacheront 7 comporte des paramètres. Le réseau de neurone prend en entrée une image et procure un index composé d'un vecteur de 1000 valeurs. La méthode de comparaison utilisée pendant la requête utilise une comparaison booléenne. La forme particulière du système nous permet d'utiliser deux types de requête : une requête textuelle et une requête par l'exemple.

Le réseau de neurone à convolution est simulé et entrainé sur le Framework Caffe en langage python, et le reste du système utilise le même langage en utilisant un GPU GTX 660 avec 2Gb de ram. L'évaluation du réseau de neurone à convolution a donné une valeur de prédiction des top-5 de 16.12% d'erreur, une valeur au-dessus de l'Alex Net. Le système d'indexation et de recherche d'image est évalué par un test d'invariance envers certaines transformations et l'étude de la courbe de rappel/précision. Les résultats ont alors montré que la méthode proposée est invariant à la

rotation modérée et au masquage d'une partie de l'image, le renversement de l'entrée ainsi que l'application de filtre perturbe le système. Causé certainement à une forte dépendance au lot d'apprentissage. La courbe de rappel et précision montre que la méthode proposée est performante et offre des bons résultats. La requête textuelle, cependant, présente des résultats moins probant que la requête par l'exemple ; un problème causé par la translation du texte de l'utilisateur en requête compréhensible au système.

L'indexation et la recherche d'image par réseau de neurone à convolution pourra donc offrir des performances bien plus élevées si on réduit le fossé sémantique de la recherche textuelle. L'utilisation de réseau de neurone spécialisé en traitement de langage naturel comme le LSTM est un exemple.

ANNEXES

ANNEXE 1

ARCHITECTURE DU SYSTEME D'INDEXATION PROPOSE

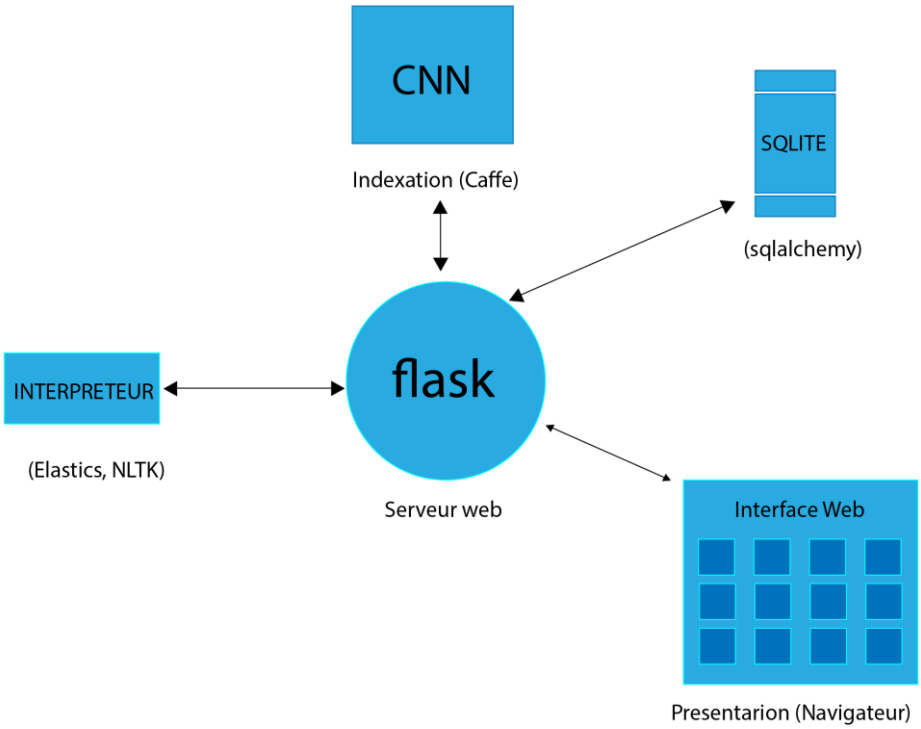

Figure A1.1 *: Architecture du système d'indexation et de recherche d'image par un CNN*

Le CNN : Chargée d'indexer les images fourni par l'utilisateur ou dans une base. Il comporte un réseau de neurone à convolution préalablement entrainé sur le set de donnée IMAGENET

Interpréteur : Entrait de la requête de l'utilisateur les termes pertinent via le Framework NLTK (Natural Langage Tool Kit). Avec l'aide du module de recherche elastics, il convertit la requête en requête compréhensible par la base de donnée.

SQLITE : Base de donnée relationnel utilisé par le biais de sqlalchemy. Le module sqlalchemy etablit une relation dierct entre un objet et des entrée dans la base.

Flask est un serveur web puissant très utilisé pour la présentation de résultat scientifique. Il a pour avantage d'être facile à mettre en place et contient plusieurs modules.

Présentation : L'utilisation de flask permet de visualisé les résultats des requête par le biais d'un navigateur web.

ANNEXE 2

EXTRAIT DES PARAMETRES D'ENTRAINEMENT DU CNN

```
layer {
  name: "norm2"
  type: "LRN"
  bottom: "pool2"
  top: "norm2"
  lrn_param {
    local_size: 5
    alpha: 0.0001
    beta: 0.75
  }
}
layer {
  name: "conv3"
  type: "Convolution"
  bottom: "norm2"
  top: "conv3"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  convolution_param {
    num_output: 384
    pad: 1
    kernel_size: 3
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}
layer {
  name: "relu3"
  type: "ReLU"
  bottom: "conv3"
  top: "conv3"
}
layer {
  name: "conv4"
  type: "Convolution"
  bottom: "conv3"
  top: "conv4"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
```

ANNEXE 3

EXTRAIT DES LABELS ASSOCIEE AUX IMAGES D'ENTRAINEMENT

n01440764 tench, Tinca tinca

n01443537 goldfish, Carassius auratus

n01484850 great white shark, white shark, man-eater, man-eating shark, Carcharodon carcharias

n01491361 tiger shark, Galeocerdo cuvieri

n01494475 hammerhead, hammerhead shark

n01496331 electric ray, crampfish, numbfish, torpedo

n01498041 stingray

n01514668 cock

n01514859 hen

n01518878 ostrich, Struthio camelus

n01530575 brambling, Fringilla montifringilla

n01531178 goldfinch, Carduelis carduelis

n01532829 house finch, linnet, Carpodacus mexicanus

n01534433 junco, snowbird

n01537544 indigo bunting, indigo finch, indigo bird, Passerina cyanea

n01558993 robin, American robin, Turdus migratorius

n01560419 bulbul

n01580077 jay

n01582220 magpie

n01592084 chickadee

n01601694 water ouzel, dipper

n01608432 kite

n01614925 bald eagle, American eagle, Haliaeetus leucocephalus

n01616318 vulture

n01622779 great grey owl, great gray owl, Strix nebulosa

n01629819 European fire salamander, Salamandra salamandra

n01630670 common newt, Triturus vulgaris

n01631663 eft

n01632458 spotted salamander, Ambystoma maculatum

n01632777 axolotl, mud puppy, Ambystoma mexicanum

n01641577 bullfrog, Rana catesbeiana

n01644373 tree frog, tree-frog

n01644900 tailed frog, bell toad, ribbed toad, tailed toad, Ascaphus trui

n01664065 loggerhead, loggerhead turtle, Caretta caretta

n01665541 leatherback turtle, leatherback, leathery turtle, Dermochelys coriacea

n01667114 mud turtle

n01667778 terrapin

n01669191 box turtle, box tortoise

n01675722 banded gecko

n01677366 common iguana, iguana, Iguana iguana

n01682714 American chameleon, anole, Anolis carolinensis

n01685808 whiptail, whiptail lizard

n01687978 agama

n01688243 frilled lizard, Chlamydosaurus kingi

ANNEXE 4

COMPARAISON DES SOLVER DANS CAFFE

AdaGrad

```
I0901 13:36:30.007884 24952 solver.cpp:232] Iteration 65000, loss = 64.1627
I0901 13:36:30.007922 24952 solver.cpp:251] Iteration 65000, Testing net (#0) # train
set
I0901 13:36:33.019305 24952 solver.cpp:289] Test loss: 63.217
I0901 13:36:33.019356 24952 solver.cpp:302]     Test net output #0: cross_entropy_loss =
63.217 (* 1 = 63.217 loss)
```

SGD

```
I0901 13:35:20.426187 20072 solver.cpp:232] Iteration 65000, loss = 61.5498
I0901 13:35:20.426218 20072 solver.cpp:251] Iteration 65000, Testing net (#0) #
train set
I0901 13:35:22.780092 20072 solver.cpp:289] Test loss: 60.8301
I0901 13:35:22.780138 20072 solver.cpp:302]     Test net output #0:
```

Nesterov

```
I0901 13:36:52.466069 22488 solver.cpp:232] Iteration 65000, loss = 59.9389
I0901 13:36:52.466099 22488 solver.cpp:251] Iteration 65000, Testing net (#0) #
train set
I0901 13:36:55.068370 22488 solver.cpp:289] Test loss: 59.3663
I0901 13:36:55.068410 22488 solver.cpp:302]     Test net output #0:
cross_entropy_loss = 59.3663 (* 1 = 59.3663 loss)
I0901 13:36:55.068418 22488 solver.cpp:302]     Test net output #1: l2_error =
1.79998
```

ANNEXE 5

REPRESENTATIONS DES CONNAISSANCES APPRIS PAR LE CNN

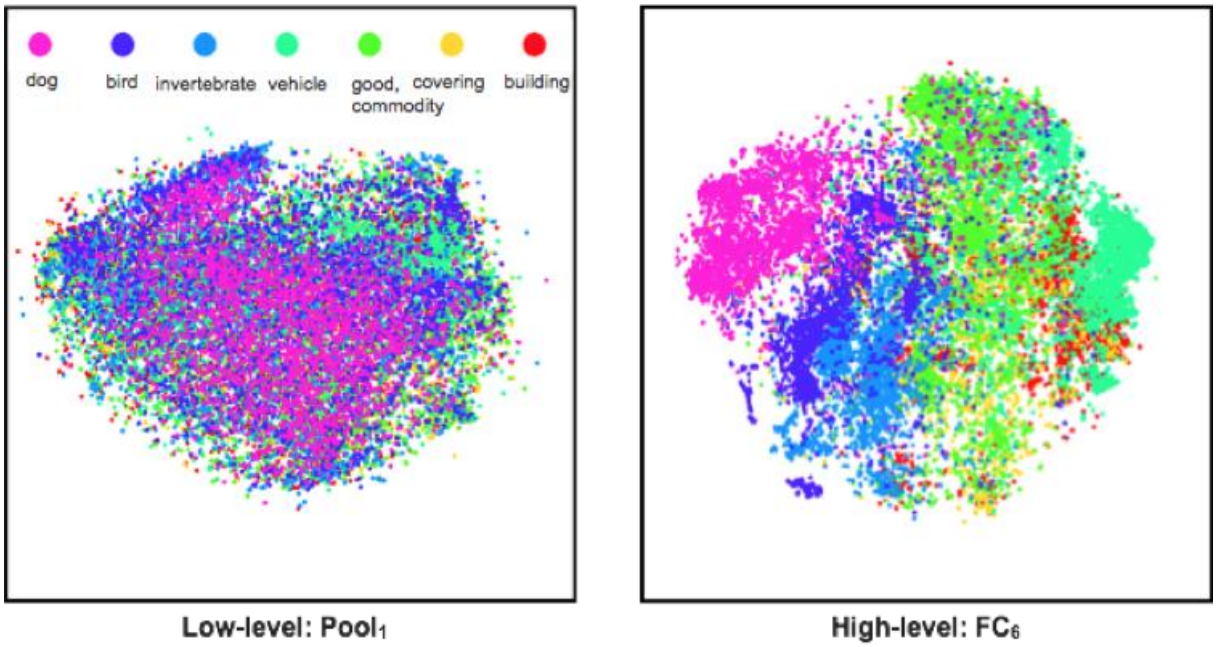

Figure A5.1 : *Séparation des classes à l'intérieure des couche du CNN*

Figure A5.2 : *Visualisation des filtres appris par le CNN par classe*

BIBLIOGROGRAPHIE

[1] S. Bedouhene, « *Recherche d'image par contenu* », Mémoire de Magister, Université Mouloud Mammeri, 2010

[2] H.M. Ramafiarisona, « *Modélisation d'un système de recherche d'images numériques par le contenu avec l'analyse factorielle de données dans une base hétérogène* », Thèse ESPA Vontovorona,

[3] M.A. Yung, « *Content-Based Image Retrieval* », Mémoire de Master en génie logiciel, Université de Monash, 2006

[4] B. Thomée, « *A picture is worth a thousand words, Content-based image retrieval techniques* », Geboren te Delft, 1981

[5] H.M. Ramafiarisona, *« Indexation d'image* », Cours M1, ESPA Vontovorona 2014

[6] A. Bougouin, « *Etat de l'art des méthodes d'extraction automatique de termes-clé* », Rencontre des Etudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RECITAL), Jun 2013, Sables d'Olonne, France. 2013.

[7] A. Pujol, « *Contribution à la Classification Sémantique d'images* », Thèse de Doctorat, Ecole Doctorale Informatique et Information pour la société(EDIIS), Juin 2009

[8] O. Agereau, « Reconnaissance Et Classification D′images De Documents », Thèse de Doctorat,

Ecole Supérieure Mathématique et Informatique, Février 2016

[9] B. Saïda, « *Traitement d'images et Reconnaissance de Formes* », Mémoire de Magister, Université Mouloud Mammeri, TIZI-OUZOU 2010

[10] A.A. Salamah, « *Efficient Content Based Image Retrieval* », Thèse, Université Islamic de Gaza, 2010

[11] E.S. Konak, « *A content-based Image Retrieval System for texture and color queries* », Thèse de Master, institut d'ingénierie et de science de l'université de Bilkent, Aout 2002

[12] R. Marée, « *Classification automatique d'image par arbres de décision* », Thèse de Doctorat, Institut Montefiore, Université de Liège, Février 2005

[13] A. Porebski, « *Sélection d'attributs de texture couleur pour la classification d'images. Application à l'identification de defauts sur les décors verriers imprimés par serigraphie* », Thèse de Doctorat, Université de LILLE 1, 20 Novembre 2009

[14] S. Bissol, « *Indexation symbolique d'images : une approche basée sur l'apprentissage non supervisé de régularités* », Interface homme-machine, Université Joseph-Fourier Grenoble I, 2005

[15] G. Roussel, « *Développement et évaluation de nouvelles méthodes de classification spatiale-spectrale d'images hyper spectrales* », Thèse de Doctorat, Université de Toulouse, 2012

[16] P. Manuel Da Silva, « *Scene Image Classification and Segmentation with Quantized Local Descriptors and Latent Aspect Modeling* », Thèse de Doctorat, Ecole Polytechnique Fédérale de Lausanne, Décembre 2006

[17] C. Touzet, « *Introductions au connexionnisme* », Cours sur www.touzet.org, Juillet 1992

[18] G. Burel, *« RESEAUX de NEURONES en TRAITEMENT d'IMAGES: Des Modèles Théoriques aux Applications Industrielles* », Thèse de Doctorat, Université de Bretagne Occidentale, 6 Décembre 1991

[19] A. Salvail-Berard, « *Réseaux De Neurones* », Cours universitaire, Sherbrook 6 Septembre 2012

[20] A. VERGE, « *Réseaux de neurones artificiels* », Lycée Michelet TIPE 2009

[21] N. FREITAS, « *Deep Learning: Lecture 5 Optimization* », Cours à l'Université d'Oxford, 2 Février 2015

[22] I. Goodfellow et A. Courville, « *Deep Learning* », MIT Press book, 2016

[23] L. Hengliang, *« Convolutional Neural Networks (CNN) Algorithm and Some Applications in Conputer Vision* », 10 Juin 2014

[24] H. Larochelle, « *Réseaux de neurones à convolution et autoencodeur* », Cours à l'Université de Sherbrook 2015

[25] J. Huang, « *Convolutional Neural Networks* », Cours vision par ordinateur à l'université d'Illinois (CS 543), 5 Juin 2015

[26] Y. LeCun, L. Botton, Y. Bengio et P. Haffner, *« Gradient-Based Learning Applied to Documernt Recognition* », IEEE, Novembre 1998

[27] A. Karpathy, « *Convolutional Neural Network for Visual Recognition* », Standford Fevrier 2015

[28] M.A. Rakotomalala, « *Cours sur le traitement d'image*», M1 TCO ESPA , Vontovorona 2014

[29] K. He, X. Zhang, S. Ren, J. Sun, « *Spatial Pyramid Pooling in Deep Convolutional Neural Network for Visual Recognition* », ECCV 2014

[30] J. Tobias, A. Dosovitskiy, T. Brox et M. Riedmiller, « *Strining for simplicity : The All Convolutional Net* », Department of Computer Science, University of Freibur, Angleterre 2015

[31] B. Graham, « *Fractional Max-Pooling* », Dept of Statistics, Un iversity of Warwick, Royaume Unis 13 May 2015

[32] X. Glorot et Y. Bengio, « *Understanding the difficulty of training deep feedforwrd neural networks* », Université de Montréal, Québec, Canada 2010

[33]N. Cohen, A. Shashua, « *SimNets : A Generalisation of Convolutional Network* », arXiv :1410.0781v3, 7 Décembre 2014

[34] T LIU, S. Fang, *« Implémentation of training Convolutional Neural Networks* », Université de Chine Academy des sciences, Beijing Chine 2014

[35] H. Wu, X. Gu, « *Towards Dropout Training for Convolutional Neural networks* », Université de Funda, Chine 2015

[36] K. He, X Zhang, S. Ren, J. Sun, « *Delving Deep into Rectifiers : Surpassing Human-Level Performance on ImageNet Classification* », Microsoft, arXiv :1502.01852v1,6 Fevrier 2015

[37] C. Szegedy, W. Liu, Y. Jia, P. Sermaner, « *Going Deeper with Convolution* », CVPR 2015

[38] D. Clevert, T. Unterthinner et S. Hochreiter, « *Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)* », Institute of Bioinformatics Johannes Kepler University, Linz, Austria, ICLR 2016

[39] K. Ovtcharov, O. Runwase, J. Kim, J. Fowers, K. Strauss et E.S. Chung, « *Accelerating Deep Convolutional Neural Network Using Specialized Hardware* », Microsoft, 22 Février 2015

[40] A. Krizhevsky, I. Sutskeer, G. E. Hinton, « *ImageNet Classification with Deep Convolutional Neural Network* », ILCVR 2012

[41] N. Qian, « *On the momentum term in gradient descent learning algorithms* », Center for Neurobiology and Behavior, Columbia University, New York 1997

[42] Z. Liano et G. Carmeiro, « *On the importance of Normalisation Layers in Deep Learning with Piecewise Linear Avtivation Units* », ARC Centre of Excellence for Robotic Vision, University of Adelaide, Australie 2015

RENSEIGNEMENTS SUR L'AUTEUR

Nom : RAMASIARISON

Prénoms : Georges Larissen

Adresse de l'auteur : Lot VO 5A Miandrarivo Ambanidia

Antananarivo 101-Madagascar

Téléphone : +261 34 80 209 32

E-mail : rgeorgeslarissen@gmail.com

Titre du mémoire :

INDEXATION ET RECHERCHE D'IMAGE PAR RESEAU DE NEURONE PROFOND A CONVOLUTION

Nombre de pages : 101

Nombre de tableaux : 5

Nombre de figures : 63

Directeur de mémoire : Monsieur RANDRIAMITANTSOA Andry Auguste

Téléphone : +261 33 07 446 66

Mail : andri23@gmail.com

RESUME

L'indexation d'image est un outil très performant pour l'organisation et le stockage de base d'image. Les méthodes de recherches d'images les plus utilisé se basent sur des descripteurs de bas niveau tel que la couleur, la texture. L'indexation textuelle a été délaissée ces dernières à parce qu'il nécessite l'intervention humaine. On propose ici un système d'indexation et de recherche d'image utilisant un réseau de neurone à convolution. La simulation de ce système a offert des résultats concluant sur son efficacité et sa performance. Cependant il nécessite de grosses ressources informatiques en termes de calcul pour la phase d'entrainement. A l'avenir, ce type de système pourra se généraliser vers d'autre domaine tel que la robotique.

Mots clés: Deep Learning, CNN, CBIR, Indexation textuelle d'image, CUDA

ABSTRACT

The image indexing is a powerful tool for organizing and storing image. The most used image search method is based on low level descriptors such as color. The textual indexing was neglected in recent years because it requires human intervention. Here we propose an indexing system and image search using a convolutional neural network. The simulation of this system has shown conclusive results on its efficiency and performance. However, in need large amount of computational resource during the training phase. In the future, this type of system will be generalized to other field such as robotics.

Keywords: Deep Learning, CNN, CBIR, Textual Image indexing, CUDA