# Table des matières

Liste des figures…….. 1
Liste des tableaux…… 1
Introduction générale……... 2
**Chapitre I : Classification des images.....** 3
I.1 Introduction…… 3
I.2 Les motivations de la classification des images……. 3
I.3 Notions de bases……. 4
I.3.1 Définition d'une image…….. 4
I.3.2 Les différents types du format d'image……. 4
I.3.3 Caractéristiques de l'image…… 5
I.3.3.1 Pixel…… 5
I.3.3.2 Dimension Résolution…… 5
I.3.3.3 Voisinage…… 5
I.3.3.4 Niveau de gris…… 6
I.3.3.5 Contraste…… 6
I.3.3.6 Luminance…….. 6
I.3.3.7 Bruit…… 6
I.3.3.8 Contour…….. 7
I.4 Méthodes de classification……. 7
I.4.1 Méthodes supervisées…… 7
I.4.2 Méthodes non supervisées……. 7
I.5 Indicateurs de performance en classification……. 7
I.5.1 Matrice de confusion……. 7
I.5.2 Courbe ROC…….. 9
I.6 Classification des images et l'apprentissage machine……... 9
I.7 Classification des images et les réseaux de neurones…… 10
I.8 Conclusion……. 11
**Chapitre II : Les réseaux de neurones convolutionnels……** 12
II.1 Introduction……... 13
II.2 Perceptron multicouches……... 14
II.2.1 Le modèle du perceptron…… 14

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

II.2.2 Le perceptron multicouche……. 17
II.3 Les différents types de réseaux de neurones……. 19
II.3.1 Adaline…… 19
II.3.2 Les réseaux de RBF…… 19
II.3.3 Réseaux de hopfield…… 20
II.3.4 Réseaux de kohonen……... 21
II.4 Deep learning…… 21
II.4.1 Quelques algorithmes de deep learning…….. 21
II.5 Les réseaux de neurones Convolutifs……... 22
II.5.1 Architecture de réseaux de neurone convolutif (CONV)……... 22
II.5.1.1 Couche de convolution (CONV)…… 24
II.5.1.2 Couche de pooling (POOL)…… 25
II.5.1.3 Couche de correction (RELU)…… 26
II.5.1.4 Couche entièrement connectée (FC)…… 26
II.5.1.5 Couche de perte (LOSS)……. 26
II.6 Exemple de modèles de CNN…….. 27
II.7 Choix des paramètres…… 27
II.7.1 Nombre de filtre……. 27
II.7.2 Forme du filtre…… 28
II.7.3 Forme du max pooling…… 28
II.8 Méthodes de régularisation……... 28
II.8.1 Empirique…… 28
II.8.1.1 Dropout…….. 28
II.8.1.2 Dropconnect……... 29
II.8.1.3 Pooling stochastique…….. 29
II.8.2 Explicite…….. 29
II.8.2.1 Taille du réseau…….. 29
II.8.2.2 Dégradation du poids……. 29
II.9 Conclusion…… 30
**Chapitre III : Implémentation…….** 31
III.1 Introduction……. 32
III.2 Logiciels et librairies utilisés dans l’implémentation…….. 32
III.2.1 Tensorflow……... 32

III.2.2 Keras…… 32
III.2.3 Python…….. 33
III.2.4 Scikit learn…….. 33
III.2.5 Configuration utilisé dans l'implémentation……... 33
III.3 Les base d'images…… 33
III.4 Architecture de notre réseau…… 36
III.5 Résultats obtenus et discussions…….. 42
III.6 Conclusion……... 50
Conclusion générale…… 51
Références bibliographiques……... 52

# Liste des figures

**Figure I.1** Voisinage à 4……. 6
**Figure I.2** Voisinage à 8……. 6
**Figure I.3** Courbe ROC…….. 9
**Figure II.1** Modèle du perceptron…….. 14
**Figure II.2** Fonction d'activations classiques……. 16
**Figure II.3** Schéma d'un perceptron multicouche…….. 17
**Figure II.4** Schéma d'un RBF…… 20
**Figure II.5** Architecture standard d'un réseau de neurone convolutionnel…… 22
**Figure II.6** Couche de CNN en 3 dimensions…… 23
**Figure II.7** Pooling avec filtre 2*2 et pas de 2……... 26
**Figure II.8** Exemples de modèles de CNN……. 28
**Figure III.1** Base d'image CIFAR-10…… 34
**Figure III.2** Base d'image CIFAR-100…….. 35
**Figure III.3** Architecture du Modèle 01…… 37
**Figure III.4** Architecture du Modèle 02…… 39
**Figure III.5** Architecture du Modèle 03…… 41
**Figure III.6** Précision et erreur pour le Modèle 01…… 42
**Figure III.7** Précision et erreur pour le Modèle 02…… 43
**Figure III.8** Matrice de Confusion pour le Modèle 02…….. 43
**Figure III.9** Nombre totale des images mal et bien classé……. 43
**Figure III.10** Taux d'erreur et précision…… 43
**Figure III.11** Nombre des images mal et bien classé par classe…. ……. 44
**Figure III.12** Précision et erreur pour le Modèle 03…….. 45
**Figure III.13** Matrice de Confusion pour le Modèle 03……. 45
**Figure III.14** Nombre totale des images mal et bien classé……...……... 45
**Figure III.15** Taux d'erreur et précision……... 45
**Figure III.16** Nombre des images mal et bien classé par classe……... 46

# Liste des tableaux

**Tableau III.1** Configuration du modèle 1 37
**Tableau III.2** Configuration du modèle 2 39
**Tableau III.3** Configuration du modèle 3 41

# Introduction générale

Nous vivons dans un monde numérique, où les informations sont stockées, traitées, indexées et recherchées par des systèmes informatiques, ce qui rend leur récupération une tâche rapide et pas cher. Au cours des dernières années, des progrès considérables ont été réalisés dans le domaine de classification d'images. Ce progrès est dû aux nombreux travaux dans ce domaine et à la disponibilité des bases d'images internationales qui ont permis aux chercheurs de signaler de manière crédible l'exécution de leurs approches dans ce domaine, avec la possibilité de les comparer à d'autres approches qu'ils utilisent les mêmes bases.

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

En 2011 et 2012 trois événements ont soudainement changé la situation. Tout d'abord, les GPU (Graphical Processing Unit) capables de plus de mille milliards d'opérations par seconde sont devenus disponibles pour un prix moins cher. Ces puissants processeurs spécialisés, initialement conçus pour le rendu graphique des jeux vidéo, se sont avérés être très performants pour les calculs des réseaux neuronaux. Deuxièmement, des expériences menées simultanément à Microsoft, Google et IBM avec l'aide du laboratoire de Geoff Hinton ont montré que les réseaux profonds pouvaient diminuer de moitié les taux d'erreurs des systèmes de reconnaissance vocale. Troisièmement plusieurs records en reconnaissance d'image ont été battus par des réseaux de neurones convolutionnels. L'événement le plus marquant a été la victoire éclatante de l'équipe de Toronto dans la compétition de reconnaissance d'objets « ImageNet ». La diminution des taux d'erreurs était telle qu'une véritable révolution. Du jour au lendemain, la majorité des équipes de recherche en parole et en vision ont abandonné leurs méthodes préférées et sont passées aux réseaux de neurones convolutionnels et autres réseaux neuronaux. L'industrie d'Internet a immédiatement saisi l'opportunité et a commencé à investir massivement dans des équipes de recherche et développements en apprentissage profond.

Dans notre projet on va utiliser les réseaux de neurones convolutionnels pour classifier les images, on va créer différents modèles avec différents architectures et par la suite on va appliquer ces modèles sur les bases d'images CIFAR-10 et CIFAR-100.

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

- Dans le premier chapitre on va présenter les notions de base de la classification des images, les différents types des images et les caractéristiques, ainsi que l'utilisation des réseaux de neurones dans la classification des images.

- Le deuxième chapitre est consacré à la description des réseaux de neurones convolutionnels ainsi que leurs l'intérêt dans le domaine de la classification des images.

- Dans la troisième chapitre, on va montrer la partie expérimentale de notre travail et on discute les différents résultats obtenus et à la fin on termine par une conclusion générale.

Rapport-Gratuit.com

# Chapitre I

# Classification des images

## I.1 Introduction

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

## I.2 Les motivations de la Classification des images

La classification des images consiste à répartir systématiquement des images selon des classes établies au préalable, classer une image lui fait correspondre une classe, marquant ainsi sa parenté avec d'autres images.

En général reconnaître une image est une taché aisée pour un humain au fil de son existence, il a acquis des connaissances qui lui permettent de s'adapter aux variations qui résultent de conditions différents d'acquisition.il lui est par exemple relativement simple de reconnaître un objet dans plusieurs orientations partiellement caché par un autre de près ou de loin et selon diverses illuminations.

Toutefois les progrès technologiques en terme d'acquisition d'images (microscopes, caméras, capteurs) et de stockage engendrent des bases de données riche en information et multiplient les domaines d'applications, il devient alors difficile pour l'humain d'analyser le nombre important d'images, le temps requis le caractère répétitif de la tâche et la concentration nécessaire sont problématiques. Toutefois celle-ci n'est pas forcément aisée pour un programme informatique pour lequel une image est un ensemble de valeur numérique

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

Les applications de la classification automatique d'images sont nombreuses et vont de l'analyse de documents à la médecine en passant par le domaine militaire. Ainsi on retrouve des applications dans le domaine médical comme la reconnaissance de cellules et de tumeurs, la reconnaissance d'écriture manuscrite pour les chèques les codes postaux. Dans le domaine urbain comme la reconnaissance de panneaux de signalisation la reconnaissance de piétons la détection de véhicules la reconnaissance de bâtiments pour aider à la localisation. Dans le domaine de la biométrie comme la reconnaissance de visage, d'empreintes, d'iris.

Le point commun à toutes ces applications est qu'elles nécessitent la mise en place d'une chaîne de traitement à partir des images disponibles composée de plusieurs étapes afin de fournir en sortie une décision. Chaque étape de la mise en place d'un tel système de classification nécessite la recherche de



Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

méthodes appropriées pour une performance globale optimale à savoir la phase d'extraction de caractéristiques et la phase d'apprentissage. Typiquement, nous disposons de données images desquelles il nous faut extraire des informations pertinentes traduites sous formes de vecteurs numériques. Cette phase d'extraction nous permet de travailler dans un espace numérique. Il s'agit ensuite d'élaborer dans la phase d'apprentissage, à partir de ces données initiales, une fonction de décision pour décider de l'appartenance d'une donnée nouvelle à l'une des classes en présence. **[01]**

## I.3 Notions de base

### I.3.1 Définition d'une image

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

L'image est considérée comme un ensemble de points auquel est affectée une grandeur physique (luminance, couleur). Ces grandeurs peuvent être continues (image analogique) ou bien discrètes (images digitales). Mathématiquement, l'image représente une fonction continue IF, appelée fonction image, de deux variables spatiales représentée par IF(x, y) mesurant la nuance du niveau de gris de l'image aux coordonnées (x, y). **[2]**

La fonction Image peut se représenter sous la forme suivante :

IF : R 2 →R Avec R: ensemble des réelles.

(x,y)→IF(x, y) x et y : Deux variables réelles.

### I.3.2 Les différents types de format d'image

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

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

- **Image binaire :** Une image binaire est une matrice rectangulaire dans l'élément valent 0 ou 1. Lorsque l'on visualise une telle image, les 0 sont représentés par du noir et les 1 par du blanc. **[02]**

### I.3.3 Caractéristiques de l'image

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

#### I.3.3.1 Pixel

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

#### I.3.3.2 Dimension & Résolution

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

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

#### I.3.3.3 Voisinage

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

On distingue deux types de voisinage :

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

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

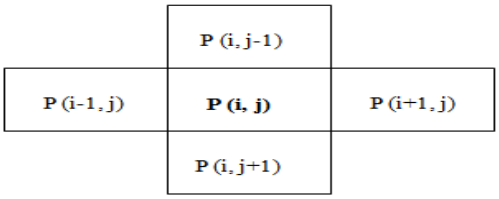



**Figure I. 1:** Voisinage à 4

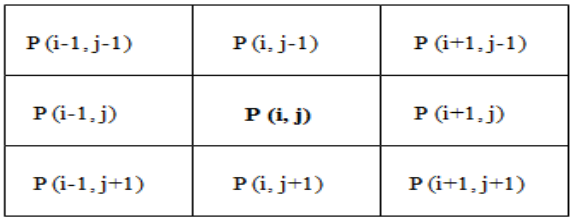



**Figure I. 2 :** Voisinage à 8

#### I.3.3.4 Niveau de gris

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

Pour 8 bits, on dispose de 256 niveaux de gris dont 40 sont reconnus à l'œil nue. Plus le nombre de bit est grand plus les niveaux sont nombreux et plus la représentation est fidèle. **[03]**

#### I.3.3.5 Contraste

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

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

#### I.3.3.6 Luminance

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

Une bonne luminance se caractérise par :

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

#### I.3.3.7 Bruit

Un bruit (parasite) dans une image est considéré comme un phénomène de brusque variation de l'intensité d'un pixel par rapport à ses voisins, il provient de l'éclairage des dispositifs optiques et électroniques du capteur. C'est un parasite qui représente certains défauts (poussière, petits nuages, baisse momentanée de l'intensité électrique sur les capteurs, ...etc.). Il se traduit par des taches de faible dimension et dont la distribution sur l'image est aléatoire. **[04]**

#### I.3.3.8 Contour

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

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

## I.4 Méthodes de classification

De nombreuses méthodes classiques ont été consacrés, elles peuvent être séparées en deux grandes catégories : les méthodes de classification supervisée et les méthodes de classification non supervisée.

### I.4.1 Méthodes supervisées

L'objectif de la classification supervisée est principalement de définir des règles permettant de classer des objets dans des classes à partir de variables qualitatives ou quantitatives caractérisant ces objets. On dispose au départ d'un échantillon dit d'apprentissage dont le classement est connu. Cet échantillon est utilisé pour l'apprentissage des règles de classement.

Il est nécessaire d'étudier la fiabilité de ces règles pour les comparer et les appliquer, évaluer les cas de sous apprentissage ou de sur apprentissage (complexité du modèle). On utilise souvent un deuxième échantillon indépendant, dit de validation ou de test.

### I.4.2 Méthodes non supervisées

Procède de la façon contraire. C'est à dire ne nécessitent aucun apprentissage et aucune tâche préalable d'étiquetage manuel. Elle consiste à représenter un nuage des points d'un espace quelconque en un ensemble de groupes appelé Cluster. Il lié généralement au domaine de l'analyse des données comme l'ACP. Un «Cluster» est une collection d'objets qui sont «similaires» entre eux et qui sont «dissemblables » par rapport aux objets appartenant à d'autres groupes. **[05]**

## I.5 Indicateurs de performance en classification

### I.5.1 Matrice de confusion

Prenons l'exemple d'un classifieur binaire, c'est-à-dire, qui prédit 2 classes notées classe 0 et classe 1.

Pour mesurer les performances de ce classifieur, il est d'usage de distinguer 4 types d'éléments classés pour la classe voulue :

1. Vrai positif VP : Elément de la classe 1 correctement prédit

2. Vrai négatif VN : Elément de la classe 0 correctement prédit
3. Faux positif FP : Elément de la classe 1 mal prédit
4. Faux négatif FN : Elément de la classe 0 mal prédit

Ces informations peuvent être rassemblés et visualisés sous forme de tableau dans une matrice de confusion. Dans le cas d'un classifieur binaire, on obtient :

| | | Classe prédite | |
|---|---|---|---|
| | | Classe 0 | Classe 1 |
| Classe réelle | Classe 0 | VN | FN |
| | Classe 1 | FP | VP |

En particulier, si la matrice de confusion est diagonale, le classifieur est parfait.

Notons que la matrice de confusion est aussi généralisable lorsqu'il y a $k > 2$ classes à prédire. **[06]**

Il est possible de calculer plusieurs indicateurs résumant la matrice de confusion. Par exemple si nous souhaitons rendre compte de la qualité de la prédiction sur la classe 1, on définit :

- **Précision**. Proportion d'éléments bien classés pour une classe donnée :

$$Precision_{de\ la\ classe\ 1} = \frac{VP}{VP + FP}$$

- **Rappel**. Proportion d'éléments bien classés par rapport au nombre d'éléments de la classe à prédire :

$$Rappel_{de\ la\ classe\ 1} = \frac{VP}{VP + FN}$$

- **F-mesure**. Mesure de compromis entre précision et rappel :

$$F - mesure_{de\ la\ classe\ 1} = \frac{2 * (Précision * Rappel)}{Précision + Rappel}$$

Il est possible de calculer tous ces indicateurs pour chaque classe. La moyenne sur chaque classe de ces indicateurs donne des indicateurs globaux sur la qualité du classifieur.

$$Precision = \frac{1}{k}\sum_{i=1}^{k}\frac{VP_i}{VP_i + FP_i}$$

$$Rappel = \frac{1}{k}\sum_{i=1}^{k}\frac{VP_i}{VP_i + FN_i}$$

$$F - mesure = \frac{2 * (Précision * Rappel)}{Précision + Rappel}$$

### I.5.2 Courbe ROC (Received Operating Characteristic):

Dans le cas d'un classifieur binaire, il est possible de visualiser les performances du classifieur sur ce que l'on appelle une **courbe ROC.** La courbe ROC est une représentation du taux de vrais positifs en fonction du taux de faux positifs. Son intérêt est de s'affranchir de la taille des données de test dans le cas où les données sont déséquilibrées.

Cette représentation met en avant un nouvel indicateur qui est l'**aire sous la courbe**. Plus elle se rapproche de 1, plus le classifieur est performant. **[06]**

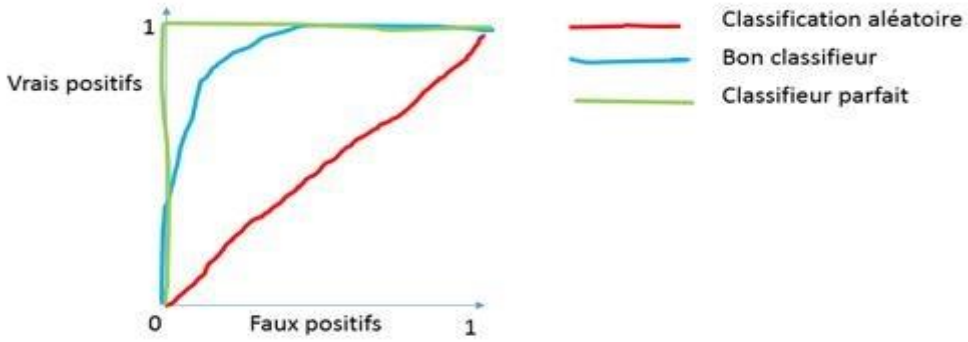



**Figure I.3 :** Courbe ROC

## I.6 Classification des images et l'apprentissage machine

Les méthodes manuelles se sont avérées très difficiles à appliquer pour des tâches en apparence très simples comme la classification des images, la reconnaissance d'objets dans les images ou la reconnaissance vocale. Les données venant du monde réel les échantillons d'un son ou les pixels d'une image sont complexes, variables et entachées de bruit.

Pour une machine, une image est un tableau de nombres indiquant la luminosité (ou la couleur) de chaque pixel, et un signal sonore une suite de nombres indiquant la pression de l'air à chaque instant.

Comment une machine peut-elle transcrire la suite de nombres d'un signal sonore en série de mots tout en ignorant le bruit ambiant, l'accent du locuteur et les particularités de sa voix? Comment une machine peut-elle identifier un chien ou une chaise dans le tableau de nombres d'une image quand l'apparence d'un chien ou d'une chaise et des objets qui les entourent peut varier infiniment ?

Il est virtuellement impossible d'écrire un programme qui fonctionnera de manière robuste dans toutes les situations. C'est là qu'intervient l'apprentissage machine (que l'on appelle aussi apprentissage automatique). C'est l'apprentissage qui anime les systèmes de toutes les grandes entreprises d'Internet. Elles l'utilisent depuis longtemps pour filtrer les contenus indésirables, ordonner des réponses à une recherche, faire des recommandations, ou sélectionner les informations intéressantes pour chaque utilisateur.

Un système entraînable peut être vu comme une boite noire avec une entrée, par exemple une image, un son, ou un texte, et une sortie qui peut représenter la catégorie de l'objet dans l'image, le mot prononcé, ou le sujet dont parle le texte. On parle alors de systèmes de classification ou de reconnaissance des formes.

Dans sa forme la plus utilisée, l'apprentissage machine est supervisé: on montre en entrée de la machine une photo d'un objet, par exemple une voiture, et on lui donne la sortie désirée pour une voiture. Puis on lui montre la photo d'un chien avec la sortie désirée pour un chien. Après chaque exemple, la machine ajuste ses paramètres internes de manière à rapprocher sa sortie de la sortie désirée. Après avoir montré à la machine des milliers ou des millions d'exemples étiquetés avec leur catégorie, la machine devient capable de classifier correctement la plupart d'entre eux. Mais ce qui est plus intéressant, c'est qu'elle peut aussi classifier correctement des images de voiture ou de chien qu'elle n'a jamais vues durant la phase l'apprentissage. C'est ce qu'on appelle la capacité de généralisation.

Jusqu'à récemment, les systèmes de reconnaissance des images classiques étaient composés de deux blocs: un extracteur de caractéristiques (feature extractor en anglais), suivi d'un classifieur entraînable simple. L'extracteur de caractéristiques est programmé «à la main», et transforme le tableau de nombres représentant l'image en une série de nombres, un vecteur de caractéristiques, dont chacun indique la présence ou l'absence d'un motif simple dans l'image. Ce vecteur est envoyé au classifieur, dont un type commun est le classifieur linéaire. Ce dernier calcule une somme pondérée des caractéristiques: chaque nombre est multiplié par un poids (positif ou négatif) avant d'être sommé. Si la somme est supérieure à un seuil, la classe est reconnue. Les poids forment une sorte de «prototype» pour la classe à laquelle le vecteur de caractéristiques est comparé. Les poids sont différents pour les classifieurs de chaque catégorie, et ce sont eux qui sont modifiés lors de l'apprentissage. Les premières méthodes de classification linéaire entraînable datent de la fin des années cinquante et sont toujours largement utilisées aujourd'hui. Elles prennent les doux noms de perceptron ou régression logistique. **[07]**

## I.7 Classification des images et les réseaux de neurones

Le problème de l'approche classique de la reconnaissance des images est qu'un bon extracteur de caractéristiques est très difficile à construire, et qu'il doit être repensé pour chaque nouvelle application.

C'est là qu'intervient l'apprentissage profond ou deep learning en anglais. C'est une classe de méthodes dont les principes sont connus depuis la fin des années 1980, mais dont l'utilisation ne s'est vraiment généralisée que depuis 2012, environ.

L'idée est très simple: le système entraînable est constitué d'une série de modules, chacun représentant une étape de traitement. Chaque module est entraînable, comportant des paramètres ajustables similaires aux poids des classifieurs linéaires. Le système est entraîné de bout en bout: à chaque exemple, tous les paramètres de tous les modules sont ajustés de manière à rapprocher la sortie produite par le système de la sortie désirée. Le qualificatif *profond* vient de l'arrangement de ces modules en couches successives.

Pour pouvoir entraîner le système de cette manière, il faut savoir dans quelle direction et de combien ajuster chaque paramètre de chaque module. Pour cela il faut calculer un gradient, c'est-à-dire pour chaque paramètre ajustable, la quantité par laquelle l'erreur en sortie augmentera ou diminuera lorsqu'on modifiera le paramètre d'une quantité donnée. Le calcul de ce gradient se fait par la méthode de rétropropagation, pratiquée depuis le milieu des années 1980.

Dans sa réalisation la plus commune, une architecture profonde peut être vue comme un réseau multicouche d'éléments simples, similaires aux classifieurs linéaires, interconnectés par des poids entraînables. C'est ce qu'on appelle un réseau neuronal multicouche.

Pourquoi neuronal? Un modèle extrêmement simplifié des neurones du cerveau les voit comme calculant une somme pondérée et activant leur sortie lorsque celle-ci dépasse un seuil. L'apprentissage modifie les efficacités des synapses, les poids des connexions entre neurones. Un réseau neuronal n'est pas un modèle précis des circuits du cerveau, mais est plutôt vu comme un modèle conceptuel ou fonctionnel. Le réseau neuronal est inspiré du cerveau un peu comme l'avion est inspiré de l'oiseau.

Ce qui fait l'avantage des architectures profondes, c'est leur capacité d'apprendre à représenter le monde de manière hiérarchique. Comme toutes les couches sont entraînables, nul besoin de construire un extracteur de caractéristiques à la main. L'entraînement s'en chargera. De plus, les premières couches extrairont des caractéristiques simples (présence de contours) que les couches suivantes combineront pour former des concepts de plus en plus complexes et abstraits: assemblages de contours en motifs, de motifs en parties d'objets, de parties d'objets en objets, etc. **[08]**

## I.8 Conclusion

Nous avons consacré ce chapitre à la présentation des notions de la classification ainsi que leurs intérêts dans le domaine d'imagerie et on a parlé aussi sur l'utilisation des réseaux de neurones dans ce domaine. Dans le deuxième chapitre on va bien détailler les réseaux de neurones et plus précisément l'utilisation des réseaux de neurones convolutionnels dans la classification des images.

Rapport-Gratuit.com

# Chapitre II

# Les Réseaux de neurones Convolutionnels

## II.1 Introduction

Les perceptrons multicouches ou MLP pour « Multi Layer Perceptron » ont montré leur efficacité comme technique d'apprentissage pour la classification de données. Ils sont en effet capables d'approximer des fonctions non– linéaires complexes afin de traiter des données de grande dimension. Dans le cadre de la classification d'images, deux approches sont possibles :

- Extraire des caractéristiques directement des données. Classiquement, ces caractéristiques sont extraites par un algorithme choisi par l'utilisateur. Les vecteurs de caractéristiques obtenus sont ensuite présentés en entrée d'un réseau de neurones.

- Présenter l'image en entrée d'un réseau de neurones. L'image nécessite cependant d'être vectorisée, c'est à dire mise sous forme d'un vecteur dont la dimension est égale au nombre de pixels de l'image.

Dans le premier cas, le réseau se contente d'effectuer une classification des vecteurs de caractéristiques. Le point sensible (l'extraction des caractéristiques) est laissé à la discrétion de l'utilisateur, et le choix de l'algorithme permettant l'extraction des caractéristiques est crucial.

Dans le deuxième cas, plusieurs problèmes se posent :

- Classiquement, les couches d'un réseau de neurones sont complètement connectées, c'est à dire que la valeur d'un neurone d'une couche n va dépendre des valeurs de tous les neurones de la couche $n - 1$. Ainsi le nombre de connexions (et donc de poids, de paramètres) peut être très grand. Par exemple pour une image de taille $15 \times 15$, la dimension de l'entrée d'un MLP est de 225. Si la couche cachée comporte 100 neurones, alors le nombre de paramètres de cette couche est de $100 \times 225 = 22500$. Le nombre de paramètres va ainsi augmenter exponentiellement avec la dimension de l'entrée (des images). Cette grande complexité du réseau impose d'avoir de nombreux échantillons d'apprentissage, ce qui n'est souvent pas le cas. Le réseau va donc avoir tendance à faire un sur apprentissage, et proposera donc une mauvaise capacité de généralisation.
- Un autre défaut des MLP pour une application à des images est qu'ils sont peu ou pas invariants à des transformations de l'entrée, ce qui arrive très souvent avec des images (légères translations, rotations ou distorsions).
- Enfin, les MLP ne prennent pas en compte la corrélation entre pixels d'une image, ce qui est un élément très important pour la reconnaissance de formes.

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

Les réseaux de neurones convolutionnels ou CNN pour « Convolutional Neural Network » sont une extension des MLP permettant de répondre efficacement aux principaux défauts des MLP. Ils sont conçus pour extraire automatiquement les caractéristiques des images d'entrée, sont invariants à de légères distorsions de l'image, et implémentent la notion de partage des poids permettant de réduire considérablement le nombre de paramètres du réseau. Ce partage des poids permet en outre de prendre en compte de manière forte les corrélations locales contenues dans une image. Les réseaux de neurones convolutionnels ont initialement été inspirés par la découverte faite par Hubel et Wiesel **[09]** de neurones sensibles aux aspects locaux et sélectifs en orientation dans le système visuel du chat.

La première utilisation des réseaux de neurones convolutionnels a été réalisée par Fukushima avec son Neocognitron **[10], [11], [12], [13].** Les poids sont forcés à être égaux pour détecter des lignes, des points ou des coins à tous les endroits possibles de l'image, implémentant de fait l'idée du partage des poids **[14].**

Une avancée importante a été effectuée par Y. Lecun et al. **[15]** avec l'utilisation d'un réseau de neurones convolutionnels dont l'apprentissage a été réalisé par propagation arrière (backpropagation). Ce modèle a notamment été appliqué avec succès pour la reconnaissance de caractères manuscrits. **[16]**

## II.2 Perceptron Multicouches

### II.2.1 Le modèle du perceptron

Le perceptron a été introduit en 1958 par Franck Rosenblatt. Il s'agit d'un neurone artificiel inspiré par la théorie cognitive de Friedrich Hayek et celle de Donald Hebb. Dans sa version la plus simple, le perceptron n'a qu'une seule sortie y à laquelle toutes les entrées xi sont connectées (voir Figure II.1), ses entrées et sorties étant booléennes. [**17]**

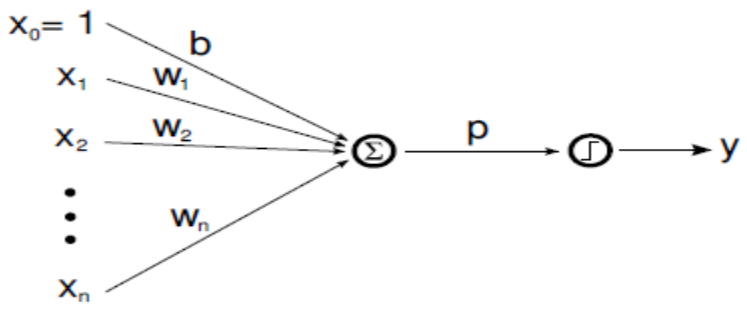



**Figure II.1 :** Modèle du perceptron



Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

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

$$p = \sum w_i x_i$$

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

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

De nombreuses variantes ont été développées, notamment la version la plus couramment utilisée où les entrées et sorties sont des nombres flottants. Les valeurs de sortie -1 et 1 remplacent fréquemment la valeur 0 et 1. Une valeur particulière, appelée biais a également été introduite. Ce biais peut être vu comme une entrée x0 supplémentaire dont la valeur est toujours de 1. Celui–ci a notamment été introduit pour un ajustement automatique du seuil, ou encore afin de pouvoir classer facilement un vecteur d'entrée dont toutes les composantes seraient nulles. Le principal obstacle de ce modèle est la détermination des poids.

Pour pallier ce problème, l'algorithme de rétro propagation du gradient a été mis au point. Soient S0 et S1 deux sous–ensembles de RN représentant les exemples négatifs et positifs de l'échantillon à apprendre. L'algorithme consiste à présenter successivement les éléments de S0 et S1 et de mettre à jour les poids pour que la sortie se rapproche de la sortie désirée.

Cet algorithme repose sur le calcul du gradient de sortie puis sur la rétro propagation de celui-ci à travers la fonction de seuil puis des poids. C'est pourquoi la fonction de seuil doit être dérivable. La fonction d'activation de Heaviside est donc remplacée par des fonctions d'activation lui ressemblant et qui sont dérivables. Les principales fonctions d'activation Φ sont : la fonction linéaire, la fonction sigmoïde et la fonction tangente hyperbolique.

$$\begin{aligned} \text{linéaire} \qquad & y = \Phi(p) = p \\ \text{sigmoïde} \qquad & y = \Phi(p) = \frac{1}{1+e^{-cp}} \quad (c > 0) \\ \text{tangente hyperbolique} \qquad & y = \Phi(p) = \frac{1-e^{-cp}}{1+e^{-cp}} \end{aligned}$$

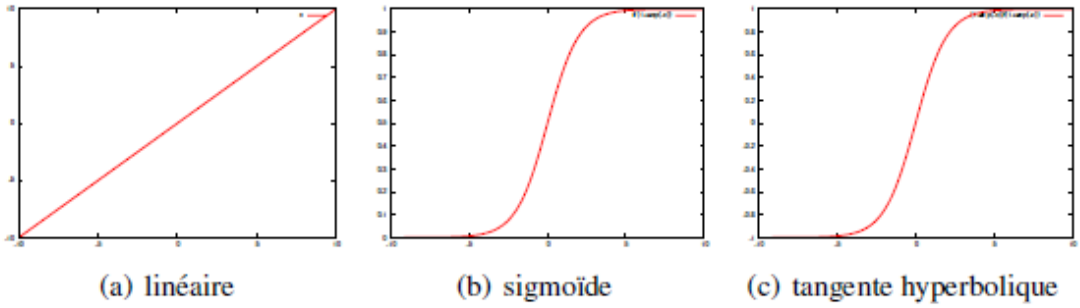



**Figure II.2 :** Fonctions d'activations classiques

La figure **II.2** montre les trois fonctions d'activation classiques. Il faut remarquer que la fonction linéaire est dans]-∞;+∞ [, la fonction sigmoïde dans] 0; 1[et la fonction tangente hyperbolique dans] -1; 1[.

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

$$\begin{aligned} p &= \sum w_i x_i \\ y &= \Phi(p) \end{aligned}$$

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

$$L = \frac{1}{2}\|o - d\|^2$$

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

Le gradient de l'erreur en sortie est donc exprimé par :

$$\frac{\partial L}{\partial y} = o - d$$

Le gradient de l'erreur pour le potentiel est :

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

Le gradient de l'erreur pour un poids $w_i$ est :

$$\begin{aligned}\frac{\partial L}{\partial w_i} &= \frac{\partial L}{\partial p}\frac{\partial p}{\partial w_i} \\ &= \frac{\partial L}{\partial p}x_i\end{aligned}$$

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

$$\begin{aligned}\frac{\partial L}{\partial x_i} &= \frac{\partial L}{\partial p}\frac{\partial p}{\partial x_i} \\ &= \frac{\partial L}{\partial p}w_i\end{aligned}$$

Cependant Minsky et Papert relèvent que le perceptron échoue pour des problèmes de classification simples, comme ceux où les classes ne sont pas Linéairement séparables. Le cas est la classification du XOR, où les motifs (0; 0) Et (1; 1) appartiennent à une classe tandis que les motifs (1; 0) et (0; 1) appartiennent à une autre classe.

Ces problèmes suggèrent l'utilisation de plusieurs perceptrons, qui organisés en couches forment le modèle du perceptron multicouches. Ce modèle est capable de traiter des problèmes non linéaires.

### II.2.2 Le perceptron multicouche (PMC)

Dans le modèle du Perceptron Multicouches, les perceptrons sont organisés en couches. Les perceptrons multicouches sont capables de traiter des données qui ne sont pas linéairement séparables. Avec l'arrivée des algorithmes de rétro propagation, ils deviennent le type de réseaux de neurones le plus utilisé. Les MLP sont généralement organisés en trois couches, la couche d'entrée, la couche intermédiaire (dite couche cachée) et la couche de sortie. L'utilité de plusieurs couches cachées n'a pas été démontrée. La figure 2.4 illustre la structure d'un MLP présentant quatre neurones en entrée, trois neurones sur la couche cachée et deux en sortie. Lorsque tous les neurones d'une couche sont connectés aux neurones de la couche suivante, on parle alors de couches complètement connectées. **[17]**

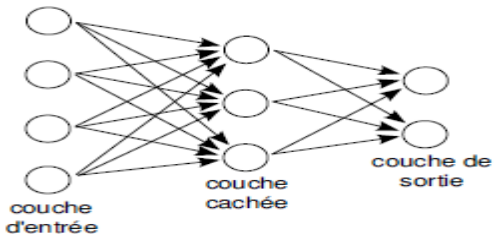



**Figure II.3 :** schéma d'un perceptron multicouche

Les équations de Propagation décrites plus haut s'appliquent à tous les neurones. Cependant, le passage d'une couche à l'autre peut être formalisé sous forme matricielle. Soit un MLP dont le nombre de neurones sur la couche d'entrée est n0, n1 sur la couche cachée et n2 sur la couche de sortie. Les poids de la couche cachée peuvent s'écrire sous la forme d'une matrice W $\epsilon$ $R^{n1*\ n0.}$ Pour un vecteur X 2 Rn0 présenté en entrée, le vecteur potentiel V et le vecteur de sortie Y pour la couche cachée s'écrivent donc :

$$\begin{aligned} V &= WX \\ Y &= \Phi(V) \end{aligned}$$

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

**Rétro propagation du gradient pour un Perceptron Multicouche**

La rétro propagation du gradient s'applique de manière récursive de la couche de sortie à la couche d'entrée. L'initialisation se fait donc par le calcul de l'erreur à la sortie de la dernière couche. Soit l'échantillon d'apprentissage X $\epsilon$ $R^{n0}$, et le vecteur de sortie désiré D $\epsilon$ $R^{n2.}$ L'erreur sur la couche de sortie peut être définie comme précédemment :

$$L = \frac{1}{2}\|Y_2 - D\|^2$$

Où le vecteur Y2 représente la sortie du MLP (i.e. la sortie de la couche 2). De manière immédiate, le gradient de l'erreur en sortie peut être calculé :

$$\frac{\partial L}{\partial Y_2} = Y_2 - D$$

La rétro propagation du gradient s'applique alors récursivement de la couche de sortie à la couche d'entrée. Pour une couche dont X est le vecteur d'entrée, V le potentiel, W la matrice de poids et Y la sortie, la rétro propagation s'écrit donc :

$$\begin{aligned} \frac{\partial L}{\partial V} &= \frac{\partial L}{\partial Y}\frac{\partial Y}{\partial V} \\ &= \Phi'(V)\frac{\partial L}{\partial Y} \\ \frac{\partial L}{\partial X} &= \frac{\partial L}{\partial V}\frac{\partial V}{\partial X} \\ &= W^T\frac{\partial L}{\partial V} \end{aligned}$$

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

W ⟵ W- $\lambda$ dW Où $\lambda$ est le taux d'apprentissage.

## II.3 Les différents types de réseaux de neurones

### II.3.1 ADALINE

L'ADAptive LINear Element a la même architecture que le perceptron, mais en revanche, il possède des valeurs d'activations continues, une fonction d'activation linéaire est une règle d'apprentissage un peu développée que celle du perceptron, elle s'appelle la règle de WIDROW & HOFF, en effet cette loi minimise l'erreur quadrique, en utilisant la méthode du gradient.

Contrairement à la règle du perceptron cette dernière opère sur les exemples bien classés et mal classés en optimisant toujours les coefficients des hyperplans séparateurs**. [18]**

Formellement, on définit la dynamique ainsi que la règle d'apprentissage comme suit :

$$S_i = \sum (W_{ij} * X_j - \theta_i)$$

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$$

$$\Delta w_{ij}(t) = \alpha * S_j * (d_i - X_i)$$

Avec :

$S_i$ : sortie réelle du neurone j.

$\alpha$ : Le pas d'apprentissage.

$X_i$ : La sortie réelle du neurone i.

$d_i$: La sortie désirée du neurone i.

$\theta_i$: Le seuil associé au neurone i.

$t$ : Numéro de l'itération.

### II.3.2 Les réseaux de RBF (Radial Basis Fonction)

Le réseau RBF est un réseau de neurones supervisé. Il s'agit d'une spécialisation d'un PMC. Un RBF est constitué uniquement de 3 couches (voir Figure II.4)

• La couche d'entrée : elle retransmet les entrées sans distorsion.

• La couche RBF : couche cachée qui contient les neurones RBF.

• La couche de sortie : simple couche qui contient une fonction linéaire.

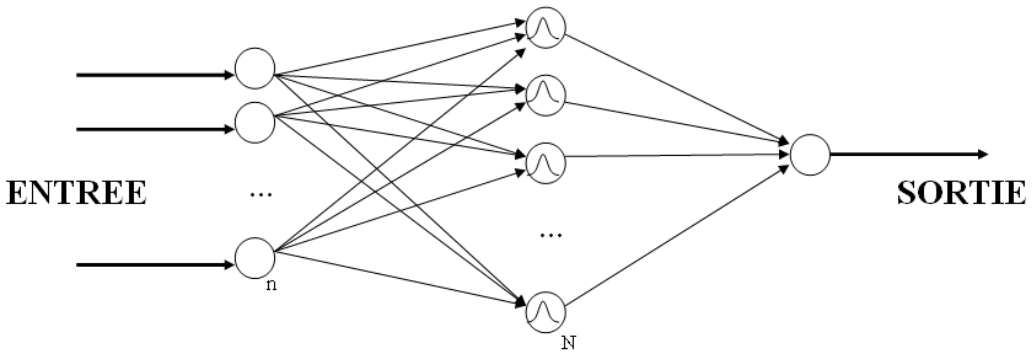



**Figure II.4 :** schéma d'un RBF

Chaque neurone RBF contient une gaussienne qui est centrée sur un point de l'espace d'entrée. Pour une entrée donnée, la sortie du neurone RBF est la hauteur de la gaussienne en ce point. La fonction gaussienne permet aux neurones de ne répondre qu'à une petite région de l'espace d'entrée, région sur laquelle la gaussienne est centrée. Donc il y a quatre paramètres principaux à régler dans un réseau RBF.

• Le nombre de neurones RBF (nombre de neurones dans l'unique couche cachée).

• La position des centres des gaussiennes de chacun des neurones.

• La largeur de ces gaussiennes.

• Le poids des connexions entre les neurones RBF et le(s) neurone(s) de sortie.

Toute modification d'un de ces paramètres entraîne directement un changement du comportement du réseau. **[18]**

### II.3.3 Réseaux de Hopfield

Il s'agit d'un réseau constitué de neurones à deux états (-1 et 1, ou 0 et 1), dont la loi d'apprentissage est la règle de Hebb (1949), qui veut qu'une synapse améliore son activité si et seulement si l'activité de ses deux neurones est corrélée (c'est-à-dire que le poids d'une connexion entre deux neurones augmente quand les deux neurones sont activés au même temps). **[18]**

### II.3.4 Réseaux de Kohonen

Contrairement aux réseaux de Hopfield où les neurones sont modélisés de la façon la plus simple possible, on recherche ici un modèle de neurone plus proche de la réalité. Ces réseaux sont inspirés des observations biologiques du fonctionnement des systèmes nerveux de perception des mammifères. Une loi de Hebb modifiée (tenant compte de l'oubli) est utilisée pour l'apprentissage. La connexion est renforcée dans le cas où les neurones reliés ont une activité simultanée et diminuée dans le cas contraire (alors qu'il ne se passait précédemment rien dans ce cas). Tous ces réseaux ont des applications dans la classification, le traitement d'image, l'aide à la décision et l'optimisation. **[18]**

## II.4 Le Deep Learning

A présent que nous avons précisé comment fonctionnent les réseaux de neurones de manière générale, nous allons aborder le domaine du Deep Learning. Cette famille d'algorithmes a permis de faire des progrès importants dans les domaines de la classification des images et du traitement du langage par exemple.

Les modèles de Deep Learning sont bâtis sur le même modèle que les perceptrons multicouches précédemment décrits. Cependant, il convient de souligner que les différentes couches intermédiaires sont plus nombreuses. Chacune des couches intermédiaires va être subdivisée en sous partie, traitant un sous problème, plus simple et fournissant le résultat à la couche suivante, et ainsi de suite. **[06]**

### II.4.1 Quelques algorithmes de Deep Learning

Il existe différents algorithmes de Deep Learning. Nous pouvons ainsi citer:

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

Rapport-Gratuit.com

## II.5 Les réseaux de neurones convolutionnels

Les réseaux de neurones convolutionnels sont à ce jour les modèles les plus performants pour classer des images. Désignés par l'acronyme CNN, de l'anglais Convolutional Neural Network, ils comportent deux parties bien distinctes. En entrée, une image est fournie sous la forme d'une matrice de pixels. Elle a 2 dimensions pour une image en niveaux de gris. La couleur est représentée par une troisième dimension, de profondeur 3 pour représenter les couleurs fondamentales [Rouge, Vert, Bleu].

La première partie d'un CNN est la partie convolutive à proprement parler. Elle fonctionne comme un extracteur de caractéristiques des images. Une image est passée à travers une succession de filtres, ou noyaux de convolution, créant de nouvelles images appelées cartes de convolutions. Certains filtres intermédiaires réduisent la résolution de l'image par une opération de maximum local. Au final, les cartes de convolutions sont mises à plat et concaténées en un vecteur de caractéristiques, appelé code CNN. **[19]**

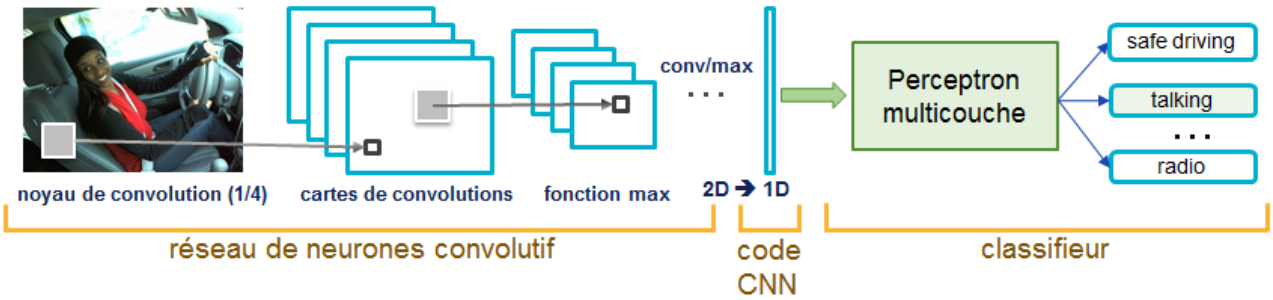



**Figure II.5** Architecture standard d'un réseau de neurone convolutionnel

Ce code CNN en sortie de la partie convolutive est ensuite branché en entrée d'une deuxième partie, constituée de couches entièrement connectées (perceptron multicouche). Le rôle de cette partie est de combiner les caractéristiques du code CNN pour classer l'image.

La sortie est une dernière couche comportant un neurone par catégorie. Les valeurs numériques obtenues sont généralement normalisées entre 0 et 1, de somme 1, pour produire une distribution de probabilité sur les catégories.

### II.5.1 Architecture de réseaux de neurone convolutionnel

Les réseaux de neurones convolutionnels sont basés sur le perceptron multicouche(MLP), et inspirés du comportement du cortex visuel des vertébrés. Bien qu'efficaces pour le traitement d'images, les MLP ont beaucoup de mal à gérer des images de grande taille, ce qui est dû à la croissance exponentielle du nombre de connexions avec la taille de l'image.

Par exemple, si on prend une image de taille 32x32x3 (32 de large, 32 de haut, 3 canaux de couleur), un seul neurone entièrement connecté dans la première couche cachée du MLP aurait 3072 entrées (32*32*3). Une image 200x200 conduirait ainsi à traiter 120 000 entrées par neurone ce qui, multiplié par le nombre de neurones, devient énorme.

Les CNN visent à limiter le nombre d'entrées tout en conservant la forte corrélation « spatialement locale » des images naturelles. Par opposition aux MLP, les CNN ont les traits distinctifs suivants :

1. ***'Volumes 3D de neurones'.*** La couche de neurones n'est plus simplement une surface (perceptron), mais devient un volume avec une profondeur. Si on considère un seul champ récepteur du CNN, les n neurones associés (sur la profondeur) forment l'équivalent de la première couche d'un MLP.
2. ***'Connectivité locale'.*** Grâce au champ récepteur qui limite le nombre d'entrées du neurone, tout en conservant l'architecture MLP, les CNN assurent ainsi que les «filtres» produisent la réponse la plus forte à un motif d'entrée spatialement localisé, ce qui conduit à une représentation parcimonieuse de l'entrée. Une telle représentation occupe moins d'espace en mémoire. De plus, le nombre de paramètres à estimer étant réduit, leur estimation (statistique) est plus robuste pour un volume de données fixé (comparé à un MLP).
3. ***'Poids partagés'* :** Dans les CNNs, les paramètres de filtrage d'un neurone (pour un champ récepteur donné) sont identiques pour tous les autres neurones d'un même noyau (traitant tous les autres champs récepteurs de l'image). Ce paramétrage (vecteur de poids et biais) est défini dans une « carte de fonction ». Cela signifie que tous les neurones dans une couche de convolution donnée détectent exactement la même caractéristique. En multipliant les champs récepteurs, il devient possible de détecter des éléments indépendamment de leur position dans le champ visuel, ce qui induit une propriété d'invariance par translation.

Ensemble, ces propriétés permettent aux réseaux de neurones convolutionnels d'obtenir une meilleure généralisation (en termes d'apprentissage) sur des problèmes de vision. Le partage de poids permet aussi de réduire considérablement le nombre de paramètres libres à apprendre, et ainsi les besoins en mémoire pour le fonctionnement du réseau. La diminution de l'empreinte mémoire permet l'apprentissage de réseaux plus grands donc souvent plus puissants. **[19]**

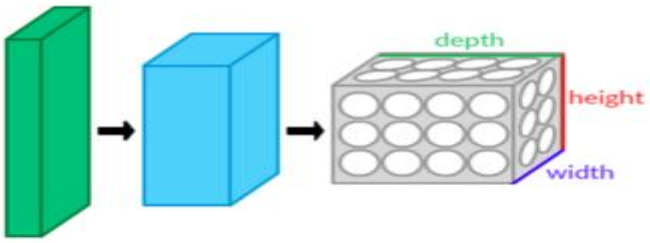



**Figure II.6 :** Une couche du CNN en 3 dimensions. (Vert = volume d'entrée, bleu = volume du champ récepteur, gris = couche de CNN, cercles = neurones artificiels indépendants)

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

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

- La couche de convolution (CONV) qui traite les données d'un champ récepteur.
- La couche de pooling (POOL), qui permet de compresser l'information en réduisant la taille de l'image intermédiaire (souvent par sous-échantillonnage).
- La couche de correction (ReLU), souvent appelée par abus 'ReLU' en référence à la fonction d'activation (Unité de rectification linéaire).
- La couche "entièrement connectée" (FC), qui est une couche de type perceptron.
- La couche de perte (LOSS).

### II.5.1.1 Couche de convolution(CONV)

La couche de convolution est le bloc de construction de base d'un CNN. Trois paramètres permettent de dimensionner le volume de la couche de convolution **la profondeur, le pas et la marge.**

1. **Profondeur de la couche :** nombre de noyaux de convolution (ou nombre de neurones associés à un même champ récepteur).
2. **Le pas:** contrôle le chevauchement des champs récepteurs. Plus le pas est petit, plus les champs récepteurs se chevauchent et plus le volume de sortie sera grand.
3. **La marge *(à 0)* ou zero padding :** parfois, il est commode de mettre des zéros à la frontière du volume d'entrée. La taille de ce *'zero-padding'* est le troisième hyper paramètre. Cette marge permet de contrôler la dimension spatiale du volume de sortie. En particulier, il est parfois souhaitable de conserver la même surface que celle du volume d'entrée.

Si le pas et la marge appliquée à l'image d'entrée permettent de contrôler le nombre de champs récepteurs à gérer (surface de traitement), la profondeur permet d'avoir une notion de volume de sortie, et de la même manière qu'une image peut avoir un volume, si on prend une profondeur de 3 pour les trois canaux RGB d'une image couleur, la couche de convolution va également présenter en sortie une profondeur. C'est pour cela que l'on parle plutôt de "volume de sortie" et de "volume d'entrée", car l'entrée d'une couche de convolution peut être soit une image soit la sortie d'une autre couche de convolution.

La taille spatiale du volume de sortie peut être calculée en fonction de la taille du volume d'entrée Wi la surface de traitement K (nombre de champs récepteurs), le pas S avec lequel ils sont appliqués, et la taille de la marge P La formule pour calculer le nombre de neurones du volume de sortie est $W0 = \frac{Wi-K+2P}{S} + 1$

Si W0 n'est pas entier, les neurones périphériques n'auront pas autant d'entrée que les autres. Il faudra donc augmenter la taille de la marge (pour recréer des entrées virtuelles).

Souvent, on considère un pas S=1, on calcule donc la marge de la manière suivante :

$P=\frac{K-1}{2}$ si on souhaite un volume de sortie de même taille que le volume d'entrée. Dans ce cas particulier la couche est dite "connectée localement".

### II.5.1.2 Couche de pooling (POOL)

Un autre concept important des CNNs est le pooling, ce qui est une forme de sous-échantillonnage de l'image. L'image d'entrée est découpée en une série de rectangles de n pixels de côté ne se chevauchant pas (pooling). Chaque rectangle peut être vu comme une tuile. Le signal en sortie de tuile est défini en fonction des valeurs prises par les différents pixels de la tuile.

Le pooling réduit la taille spatiale d'une image intermédiaire, réduisant ainsi la quantité de paramètres et de calcul dans le réseau. Il est donc fréquent d'insérer périodiquement une couche de pooling entre deux couches convolutives successives d'une architecture CNN pour contrôler l'overfitting (sur-apprentissage). L'opération de pooling créait aussi une forme d'invariance par translation.

La couche de pooling fonctionne indépendamment sur chaque tranche de profondeur de l'entrée et la redimensionne uniquement au niveau de la surface. La forme la plus courante est une couche de mise en commun avec des tuiles de taille 2x2 (largeur/hauteur) et comme valeur de sortie la valeur maximale en entrée. On parle dans ce cas de « Max-Pool 2x2 ».

Il est possible d'utiliser d'autres fonctions de pooling que le maximum. On peut utiliser un « average pooling » (la sortie est la moyenne des valeurs du patch d'entrée), du « L2-norm pooling ». Dans les faits, même si initialement l'average pooling était souvent utilisé il s'est avéré que le max-pooling était plus efficace car celui-ci augmente plus significativement l'importance des activations fortes. En d'autres circonstances, on pourra utiliser un pooling stochastique.

Le pooling permet de gros gains en puissance de calcul. Cependant, en raison de la réduction agressive de la taille de la représentation (et donc de la perte d'information associée), la tendance actuelle est d'utiliser de petits filtres (type 2x2). Il est aussi possible d'éviter la couche de pooling mais cela implique un risque sur-apprentissage plus important.



Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

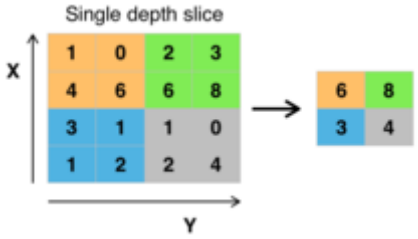



**Figure II.7 :** Pooling avec un filtre 2x2 et un pas de 2

### II.5.1.3 Couches de correction (RELU)

Il est possible d'améliorer l'efficacité du traitement en intercalant entre les couches de traitement une couche qui va opérer une fonction mathématique (fonction d'activation) sur les signaux de sortie.

La fonction ReLU (abréviation de Unités Rectifié linéaires) : **F(x)=max(0,x)** Cette fonction force les neurones à retourner des valeurs positives.

### II.5.1.4 Couche entièrement connectée (FC)

Après plusieurs couches de convolution et de max-pooling, le raisonnement de haut niveau dans le réseau neuronal se fait via des couches entièrement connectées. Les neurones dans une couche entièrement connectée ont des connexions vers toutes les sorties de la couche précédente. Leurs fonctions d'activations peuvent donc être calculées avec une multiplication matricielle suivie d'un décalage de polarisation.

### II.5.1.5 Couche de perte (LOSS)

La couche de perte spécifie comment l'entrainement du réseau pénalise l'écart entre le signal prévu et réel. Elle est normalement la dernière couche dans le réseau. Diverses fonctions de perte adaptées à différentes tâches peuvent y être utilisées. La fonction « Softmax » permet de calculer la distribution de probabilités sur les classes de sortie.

## II.6 Exemples de modèles de CNN

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

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

L'empilage des couches CONV avec de petits filtres de pooling (plutôt un grand filtre) permet un traitement plus puissant, avec moins de paramètres. Cependant, avec l'inconvénient de demander plus de puissance de calcul (pour contenir tous les résultats intermédiaires de la couche CONV).

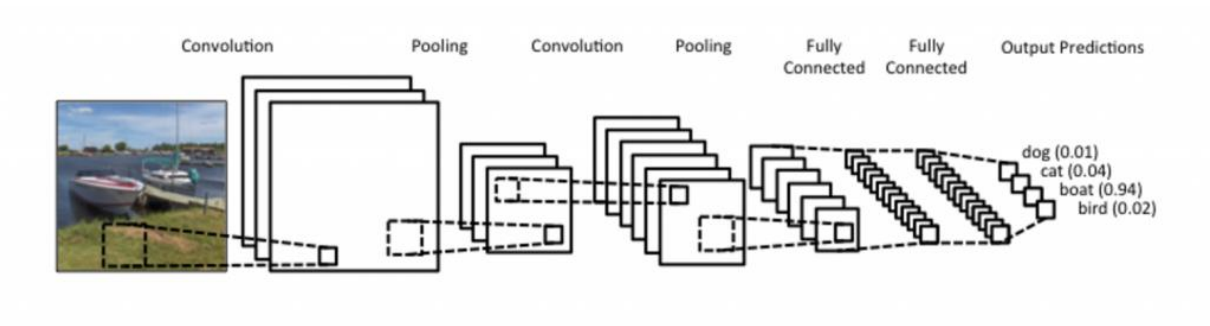



**Figure II.8 :** Exemples de modèles de CNN

## II.7 Choix des paramètres

Les CNNs utilisent plus de paramètres qu'un MLP standard. Même si les règles habituelles pour les taux d'apprentissage et des constantes de régularisation s'appliquent toujours, il faut prendre en considération les notions de nombre de filtres, leur forme et la forme du max pooling. **[19]**

### II.7.1 Nombre de filtres

Comme la taille des images intermédiaires diminue avec la profondeur du traitement, les couches proches de l'entrée ont tendance à avoir moins de filtres tandis que les couches plus proches de la sortie peuvent en avoir davantage. Pour égaliser le calcul à chaque couche, le produit du nombre de caractéristiques et le nombre de pixels traités est généralement choisi pour être à peu près constant à travers les couches. Pour préserver l'information en entrée, il faudrait maintenir le nombre de sorties intermédiaires (nombre

d'images intermédiaire multiplié par le nombre de positions de pixel) pour être croissante (au sens large) d'une couche à l'autre. le nombre d'images intermédiaires contrôle directement la puissance du système, dépend du nombre d'exemples disponibles et la complexité du traitement.

### II.7.2 Forme du filtre

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

### II.7.3 Forme du Max Pooling

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

## II.8 Méthodes de régularisation

Pour ne pas tomber dans le problème de  sur apprentissage il y a des méthodes de régularisation à utiliser.

### II.8.1 Empirique

#### II.8.1.1 Dropout

Les couches "FC" (Fully Connected) occupent la majeure partie de la mémoire du CNN. D'ailleurs le concept de FC crée un problème exponentiel de mémoire appelé "overfitting" ("sur-connexion" conduisant au sur-apprentissage) ralentissant le traitement de l'information. Pour prévenir cela, la méthode du dropout est utilisée pour "éteindre" les neurones aléatoirement (avec une probabilité prédéfinie, souvent un neurone sur deux) ainsi que les neurones périphériques. Ainsi, avec moins de neurones, le réseau est plus réactif et peut donc apprendre plus rapidement. À la fin de la séance d'apprentissage, les neurones "éteints" sont "rallumés" (avec leurs poids originaux). Plus la couche FC est proche de l'image source, moins on éteindra de neurones.

L'objectif est d'éteindre et rallumer les neurones aléatoirement, dans le cadre d'entraînements successifs. Une fois les séries d'entraînements terminées, on rallume tous les neurones et on utilise le réseau comme d'habitude. Cette technique a montré non seulement un gain dans la vitesse d'apprentissage, mais en déconnectant les neurones, on a aussi limité des effets marginaux, rendant le réseau plus robuste et capable de mieux généraliser les concepts appris. **[19]**

#### II.8.1.2 DropConnect

Le DropConnect est une évolution du dropout, où on ne va non plus éteindre un neurone, mais une connexion (l'équivalent de la synapse), et ce de manière toujours aléatoire. Les résultats sont similaires (rapidité, capacité de généralisation de l'apprentissage), mais présentent une différence au niveau de l'évolution des poids des connexions. Une couche FC avec un DropConnect peut s'apparenter à une couche à connexion "diffuse".

#### II.8.1.3 Pooling stochastique

Le pooling stochastique reprend le même principe que le Max-pooling, mais la sortie choisie sera prise au hasard, selon une distribution multinomiale définie en fonction de l'activité de la zone adressée par le pool. Dans les faits, ce système s'apparente à faire du Max-pooling avec un grand nombre d'images similaires, qui ne varient que par des déformations localisées. On peut aussi considérer cette méthode comme une adaptation à des déformations élastiques de l'image. C'est pourquoi cette méthode est très efficace sur les images MNIST (base de données d'images représentant des chiffres manuscrits). La force du pooling stochastique est de voir ses performances croître de manière exponentielle avec le nombre de couches du réseau.

### II.8.2 Explicite

#### II.8.2.1 Taille du réseau

La manière la plus simple de limiter le sur apprentissage est de limiter le nombre de couches du réseau et de libérer les paramètres libres (connexions) du réseau. Ceci réduit directement la puissance et le potentiel prédictif du réseau. C'est équivalent à avoir une "norme zéro".

#### II.8.2.2 Dégradation du poids

Le concept est de considérer le vecteur des poids d'un neurone (liste des poids associés aux signaux entrants), et de lui rajouter un vecteur d'erreur proportionnel à la somme des poids (norme 1) ou du carré des poids (norme 2 ou euclidienne). Ce vecteur d'erreur peut ensuite être multiplié par un coefficient de proportionnalité que l'on va augmenter pour pénaliser davantage les vecteurs de poids forts.

- La régularisation par norme 1 : La spécificité de cette régulation est de diminuer le poids des entrées aléatoires et faibles et d'augmenter le poids des entrées "importantes". Le système devient moins sensible au bruit.
- La régularisation par norme 2 : (norme euclidienne) La spécificité de cette régulation est de diminuer le poids des entrées fortes, et de forcer le neurone à plus prendre en compte les entrées de poids faible.

Les régularisations par norme 1 et norme 2 peuvent être combinées : c'est la "régularisation de réseau élastique" (Elastic net regulation).

## II.9 Conclusion

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

Les réseaux de neurones convolutionnels présentent cependant un certain nombre de limitations, en premier lieu, les hyper paramètres du réseau sont difficiles à évaluer a priori. En effet, le nombre de couches, les nombre de neurones par couche ou encore les différentes connexions entre couches sont des éléments cruciaux et essentiellement déterminés par une bonne intuition ou par une succession de tests/calcul d'erreurs (ce qui est coûteux en temps). Le nombre d'échantillons d'apprentissage est également un élément déterminant, et il arrive souvent que celui–ci soit trop faible en comparaison du nombre de paramètres (poids) du réseau. Des solutions existent comme augmenter artificiellement leur nombre ou encore en réduisant le nombre de paramètres libres (en réalisant un préapprentissage des premières couches par exemple). Dans le chapitre suivant, on va proposer notre modèle et ensuite interpréter les résultats obtenus dans la phase d'apprentissage et de test et les discuter et critiquer.

# Chapitre III

# Implémentation

Rapport-Gratuit.com

## III.1 Introduction

Dans ce chapitre, on va définir l'architecture des trois modèles qu'on a créé et par la suite on va appliquer ces modèles sur la base d'images CIFAR 10 et CIFAR 100. Pour cela, on va travailler avec les bibliothèques Tensorflow et Keras pour l'apprentissage et la classification et afin d'améliorer les performances des modèles on va utiliser quelques techniques simple et efficaces comme data augmentation et dropout.

## III.2 Logiciels et librairies Utilisés dans l'implémentation

### III.2.1 TensorFlow

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

### III.2.2 Keras

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

En 2017, l'équipe TensorFlow de Google a décidé de soutenir Keras dans la bibliothèque principale de TensorFlow. Chollet a expliqué que Keras a été conçue comme une interface plutôt que comme un cadre d'apprentissage end to end. Il présente un ensemble d'abstractions de niveau supérieur et plus intuitif qui facilitent la configuration des réseaux neuronaux indépendamment de la bibliothèque informatique de backend. Microsoft travaille également à ajouter un backend CNTK à Keras aussi.

https://www.tensorflow.org/

https://keras.io/

### III.2.3 Python

Python est un langage de programmation de haut niveau interprété (il n'y a pas d'étape de compilation) et orienté objet avec une sémantique dynamique. Il est très sollicité par une large communauté de développeurs et de programmeurs. Python est un langage simple, facile à apprendre et permet une bonne réduction du cout de la maintenance des codes. Les bibliothèques (packages) python encouragent la modularité et la réutilisabilité des codes. Python et ses bibliothèques sont disponibles (en source ou en binaires) sans charges pour la majorité des plateformes et peuvent être redistribués gratuitement.

### III.2.4 Scikit-learn

Scikit-learn est une bibliothèque libre Python dédiée à l'apprentissage automatique. Elle est développée par de nombreux contributeurs notamment dans le monde académique par des instituts français d'enseignement supérieur et de recherche comme Inria et Télécom ParisTech.

Elle comprend notamment des fonctions pour estimer des forêts aléatoires, des régressions logistiques, des algorithmes de classification, et les machines à vecteurs de support. Elle est conçue pour s'harmoniser avec des autres bibliothèques libre Python, notamment NumPy et SciPy.

### III.2.5 Configuration Utilisé dans l'implémentation

La configuration du matériel utilisé dans notre implémentation est :

- Un PC portable Dell i5 CPU 2.40 GHZ
- Carte graphique Nvidia GeForce GT525M
- RAM de taille 4 GO
- Disque dur de taille 500 GO
- Système d'exploitation Linux Ubuntu version 16.04

## III.3 Les base d'images

CIFAR-10 et CIFAR-100 sont des sous-ensembles d'images étiquetés parmi 80 millions d'images. Ils ont été collectés par Alex Krizhevsky, Vinod Nair et Geoffrey Hinton dont le but de les utiliser dans le domaine de la reconnaissance d'objet. Ils sont disponibles sur lien suivant http://www.cs.toronto.edu/~kriz/cifar.html

https://scikit-learn.org/

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

- **CIFAR-10**

La base d'image de CIFAR-10 se compose de 60000 images couleur, chaque image à une taille de 32x32, ces images sont réparties en 10 classes, avec 6000 images par classe. Dans cette base on trouve 50000 images pour l'apprentissage et 10000 images pour le test.

**Figure III.1** Base d'image CIFAR-10

- **CIFAR-100**

C'est une base d'image qui contient les mêmes caractéristiques que CIFAR-10, sauf qu'elle possède 100 classes contenant 600 images pour chaque classe. Il y a 500 images pour l'apprentissage et 100 images pour le test par classe. Les 100 classes du CIFAR-100 sont regroupées en 20 superclasses. Chaque image est livrée avec une étiquette "fine" (la classe à laquelle elle appartient) et une étiquette "grossière" (la superclasse à laquelle elle appartient). Voici la liste des classes dans le CIFAR-100:

| Superclass | Classes |
|---|---|
| **aquatic mammals** | beaver, dolphin, otter, seal, whale |
| **fish** | aquarium fish, flatfish, ray, shark, trout |
| **flowers** | orchids, poppies, roses, sunflowers, tulips |
| **food containers** | bottles, bowls, cans, cups, plates |
| **fruit and vegetables** | apples, mushrooms, oranges, pears, sweet peppers |
| **household electrical devices** | clock, computer keyboard, lamp, telephone, television |
| **household furniture** | bed, chair, couch, table, wardrobe |
| **insects** | bee, beetle, butterfly, caterpillar, cockroach |
| **large carnivores** | bear, leopard, lion, tiger, wolf |
| **large man-made outdoor things** | bridge, castle, house, road, skyscraper |
| **large natural outdoor scenes** | cloud, forest, mountain, plain, sea |
| **large omnivores and herbivores** | camel, cattle, chimpanzee, elephant, kangaroo |
| **medium-sized mammals** | fox, porcupine, possum, raccoon, skunk |
| **non-insect invertebrates** | crab, lobster, snail, spider, worm |
| **people** | baby, boy, girl, man, woman |
| **reptiles** | crocodile, dinosaur, lizard, snake, turtle |
| **small mammals** | hamster, mouse, rabbit, shrew, squirrel |
| **trees** | maple, oak, palm, pine, willow |
| **vehicles 1** | bicycle, bus, motorcycle, pickup truck, train |
| **vehicles 2** | lawn-mower, rocket, streetcar, tank, tractor |

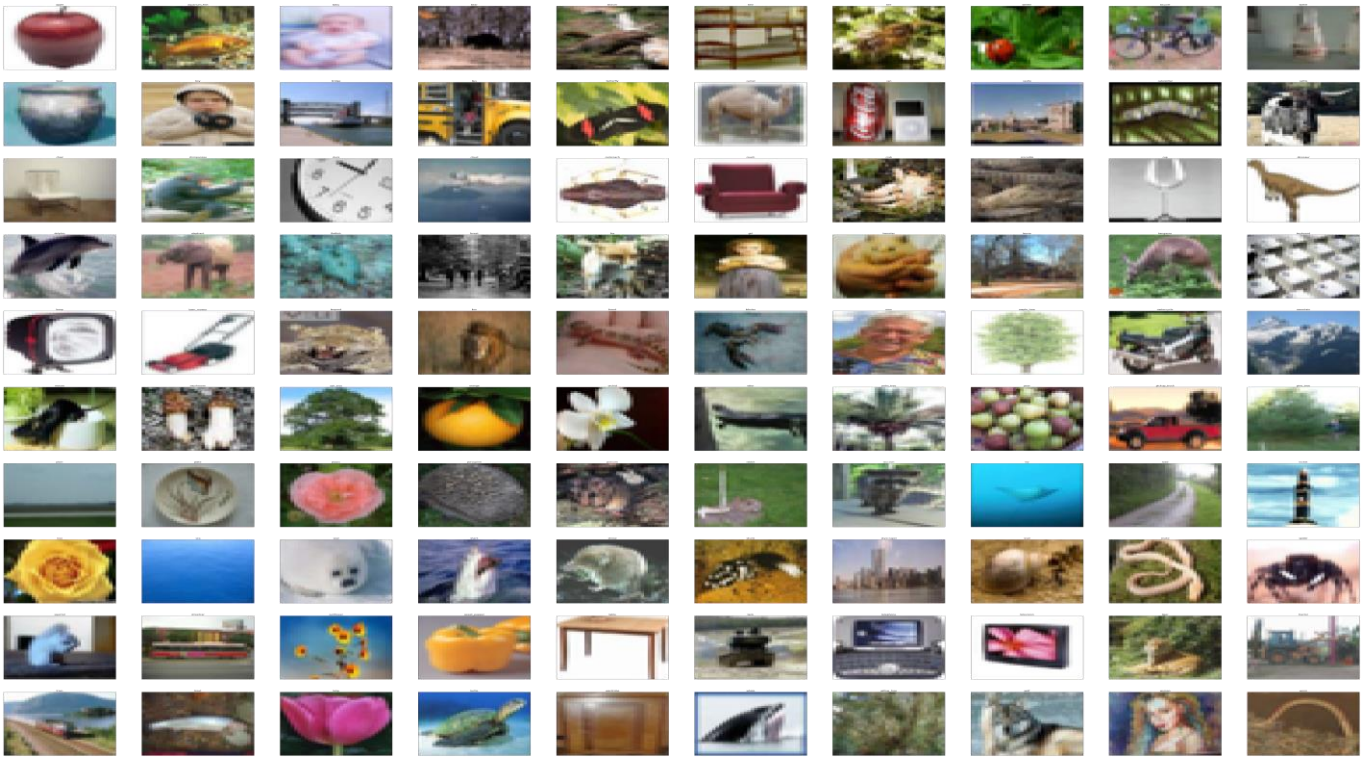


**Figure III.2** Base d'image CIFAR-100

## III.4 Architecture de notre réseau

Au cours de nos expérimentations, nous avons créé trois modèles (modèle 1, modèle 2 et modèle 3) avec différents architectures, où on a appliqué le modèle 1 sur la base d'image CIFAR100 et les modèles 2 et 3 sur la base d'image CIFAR10.

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

### Architecture du modèle 01

Le premier modèle que nous présentons dans la **figure III.3** est composé de cinq couches de convolution et deux couches de maxpooling et trois couches de fully connected.

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

Les 32 feature maps qui sont obtenus auparavant ils sont donnés en entrée de la deuxième couche de convolution qui est composée aussi de 32 filtres, une fonction d'activation RELU est appliquée sur la couche de convolution, ensuite on applique Maxpooling pour réduire la taille de l'image ainsi la quantité de paramètres et de calcul. À la sortie de cette couche, nous aurons 32 feature maps de taille 16*16.

On répète la même chose avec les couches de convolutions trois, quatre et cinq, ces couches sont composées de 64 filtres, la fonction d'activation ReLU est appliquée toujours sur chaque convolution. Une couche de Maxpooling est appliquée après la couche de convolution cinq. À la sortie de cette couche, nous aurons 64 feature maps de taille 8*8. Le vecteur de caractéristiques issu des convolutions a une dimension de 4096.

Après ces cinq couches de convolution, nous utilisons un réseau de neurones composé de trois couches fully connected. Les deux premières couches ont chacune 1 024 neurones où la fonction d'activation utilisée est le ReLU, et la troisième couche est un softmax qui permet de calculer la distribution de probabilité des 100 classes (nombre de classe dans la base d'image CIFAR100).



Rapport-gratuit.com

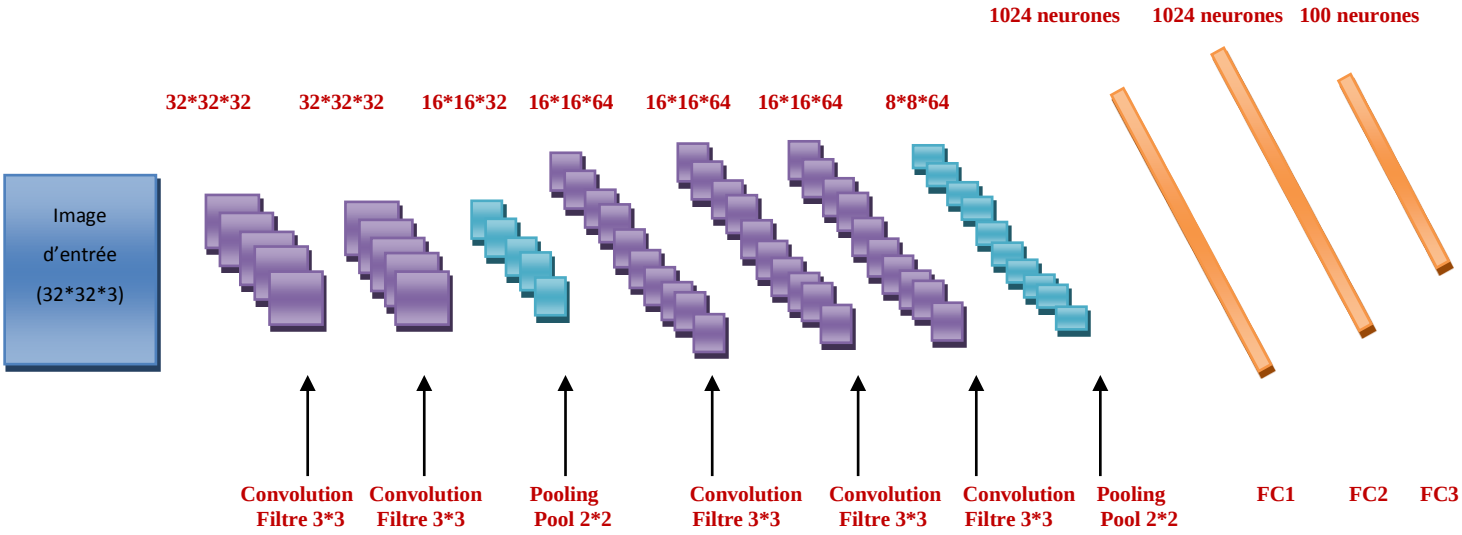



**Figure III.3** Architecture du modèle 1

| Layer (type) | Output Shape | Param # | Connected to |
|---|---|---|---|
| input_1 (InputLayer) | (None, 3, 32, 32) | 0 | |
| convolution2d_1 (Convolution2D) | (None, 32, 32, 32) | 896 | input_1[0][0] |
| convolution2d_2 (Convolution2D) | (None, 32, 32, 32) | 9248 | convolution2d_1[0][0] |
| maxpooling2d_1 (MaxPooling2D) | (None, 32, 16, 16) | 0 | convolution2d_2[0][0] |
| convolution2d_3 (Convolution2D) | (None, 64, 16, 16) | 18496 | maxpooling2d_1[0][0] |
| convolution2d_4 (Convolution2D) | (None, 64, 16, 16) | 36928 | convolution2d_3[0][0] |
| convolution2d_5 (Convolution2D) | (None, 64, 16, 16) | 36928 | convolution2d_4[0][0] |
| maxpooling2d_2 (MaxPooling2D) | (None, 64, 8, 8) | 0 | convolution2d_5[0][0] |
| flatten_1 (Flatten) | (None, 4096) | 0 | maxpooling2d_2[0][0] |
| dense_1 (Dense) | (None, 1024) | 4195328 | flatten_1[0][0] |
| dropout_1 (Dropout) | (None, 1024) | 0 | dense_1[0][0] |
| dense_2 (Dense) | (None, 1024) | 1049600 | dropout_1[0][0] |
| dropout_2 (Dropout) | (None, 1024) | 0 | dense_2[0][0] |
| dense_3 (Dense) | (None, 100) | 102500 | dropout_2[0][0] |

Total params: 5,449,924

**Tableau III.1** Configuration du modèle 1

### Architecture du modèle 02

Le deuxième modèle que nous présentons dans la **figure III.4** est composé de six couches de convolution et trois couches de maxpooling et de trois couches de fully connected.

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

Ensuite, les 32 feature maps qui sont obtenus ils sont données en entrée de la deuxième couche de convolution qui est composée aussi de 32 filtres. La fonction d'activation ReLU est appliquée sur les couches de convolutions. Le Maxpooling est appliqué après pour réduire la taille de l'image et des paramètres. À la sortie de cette couche, nous aurons 32 feature maps de taille 16*16.

On répète la même chose avec les couches de convolutions trois, quatre, cinq et six (les couches trois et quatre sont composées de 64 filtres et les couches cinq et six sont composées de 128 filtres), la fonction d'activation ReLU est appliquée toujours sur chaque convolution. Une couche de Maxpooling est appliquée après les couches de convolutions quatre et six. À la sortie de la dernière couche Maxpooling, nous aurons 128 feature maps de taille 4*4. Le vecteur de caractéristiques issu des convolutions a une dimension de 2048.

Après ces six couches de convolution, nous utilisons un réseau de neurones composé de trois couches fully connected. Les deux premières couches ont chacune 1 024 neurones où la fonction d'activation utilisée est le ReLU, et la troisième couche est un softmax qui permet de calculer la distribution de probabilité des 10 classes (nombre de classe dans la base d'image CIFAR10).

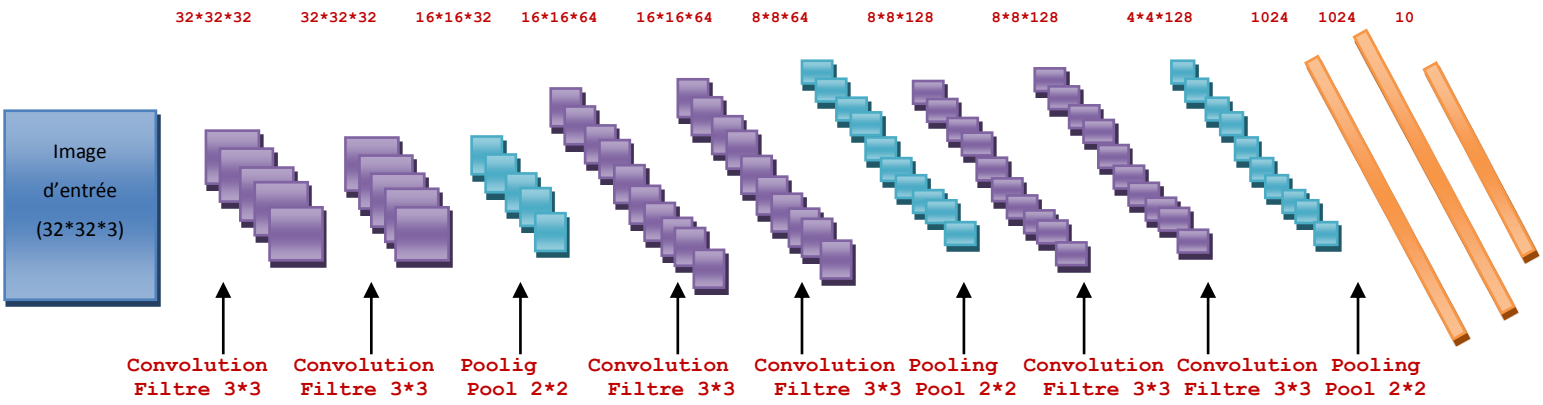



**Figure III.4** Architecture du modèle 2

| Layer (type) | Output Shape | Param # | Connected to |
|---|---|---|---|
| input_1 (InputLayer) | (None, 3, 32, 32) | 0 | |
| convolution2d_1 (Convolution2D) | (None, 32, 32, 32) | 896 | input_1[0][0] |
| convolution2d_2 (Convolution2D) | (None, 32, 32, 32) | 9248 | convolution2d_1[0][0] |
| maxpooling2d_1 (MaxPooling2D) | (None, 32, 16, 16) | 0 | convolution2d_2[0][0] |
| convolution2d_3 (Convolution2D) | (None, 64, 16, 16) | 18496 | maxpooling2d_1[0][0] |
| convolution2d_4 (Convolution2D) | (None, 64, 16, 16) | 36928 | convolution2d_3[0][0] |
| maxpooling2d_2 (MaxPooling2D) | (None, 64, 8, 8) | 0 | convolution2d_4[0][0] |
| convolution2d_5 (Convolution2D) | (None, 128, 8, 8) | 73856 | maxpooling2d_2[0][0] |
| convolution2d_6 (Convolution2D) | (None, 128, 8, 8) | 147584 | convolution2d_5[0][0] |
| maxpooling2d_3 (MaxPooling2D) | (None, 128, 4, 4) | 0 | convolution2d_6[0][0] |
| flatten_1 (Flatten) | (None, 2048) | 0 | maxpooling2d_3[0][0] |
| dense_1 (Dense) | (None, 1024) | 2098176 | flatten_1[0][0] |
| dropout_1 (Dropout) | (None, 1024) | 0 | dense_1[0][0] |
| dense_2 (Dense) | (None, 1024) | 1049600 | dropout_1[0][0] |
| dropout_2 (Dropout) | (None, 1024) | 0 | dense_2[0][0] |
| dense_3 (Dense) | (None, 10) | 10250 | dropout_2[0][0] |

Total params: 3,445,034

**Tableau III.2** Configuration du modèle 2

### Architecture du modèle 03

Le troisième modèle que nous présentons dans la **figure III.5** est composé de quatre couches de convolution et deux couches de maxpooling et de deux couches de fully connected.

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

Ensuite, les 32 feature maps qui sont obtenus ils sont données en entrée de la deuxième couche de convolution qui est composée aussi de 32 filtres. La fonction d'activation ReLU est appliquée sur cette couche. Le Maxpooling est appliqué après pour réduire la taille de l'image. À la sortie de cette couche, nous aurons 32 feature maps de taille 16*16.

On répète la même chose avec les couches de convolutions trois et quatre qui sont composées de 64 filtres, la fonction d'activation ReLU est appliquée toujours sur chaque convolution. Une couche de Maxpooling est appliqué après la couche de convolution quatre. À la sortie de cette couche, nous auros 64 feature maps de taille 8*8. Le vecteur de caractéristiques issu des convolutions a une dimension de 4096.

Après ces quatre couches de convolution, nous utilisons un réseau de neurones composé de deux couches fully connected. La première couche est composée de 512 neurones où la fonction d'activation utilisée est le ReLU, la deuxième couche utilise la fonction softmax qui permet de calculer la distribution de probabilité des 10 classes (nombre de classe dans la base d'image CIFAR10).

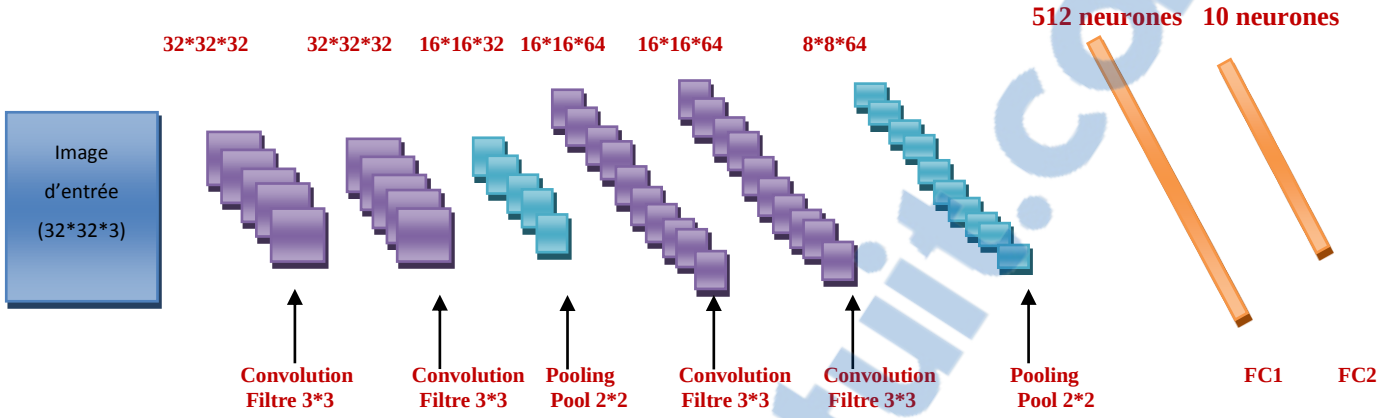



**Figure III.5** Architecture du modèle 3

| Layer (type) | Output Shape | Param # | Connected to |
|---|---|---|---|
| input_1 (InputLayer) | (None, 3, 32, 32) | 0 | |
| convolution2d_1 (Convolution2D) | (None, 32, 32, 32) | 896 | input_1[0][0] |
| convolution2d_2 (Convolution2D) | (None, 32, 32, 32) | 9248 | convolution2d_1[0][0] |
| maxpooling2d_1 (MaxPooling2D) | (None, 32, 16, 16) | 0 | convolution2d_2[0][0] |
| dropout_1 (Dropout) | (None, 32, 16, 16) | 0 | maxpooling2d_1[0][0] |
| convolution2d_3 (Convolution2D) | (None, 64, 16, 16) | 18496 | dropout_1[0][0] |
| convolution2d_4 (Convolution2D) | (None, 64, 16, 16) | 36928 | convolution2d_3[0][0] |
| maxpooling2d_2 (MaxPooling2D) | (None, 64, 8, 8) | 0 | convolution2d_4[0][0] |
| dropout_2 (Dropout) | (None, 64, 8, 8) | 0 | maxpooling2d_2[0][0] |
| flatten_1 (Flatten) | (None, 4096) | 0 | dropout_2[0][0] |
| dense_1 (Dense) | (None, 512) | 2097664 | flatten_1[0][0] |
| dropout_3 (Dropout) | (None, 512) | 0 | dense_1[0][0] |
| dense_2 (Dense) | (None, 10) | 5130 | dropout_3[0][0] |

Total params: 2,168,362

**Tableau III.3** Configuration du modèle 3

Rapport-Gratuit.com

## III.5 Résultats obtenus et discussion

Afin de montrer les résultats obtenus pour les trois modèles, on illustre dans ce qui suit les résultats en termes de précision et d'erreur ainsi que matrice de confusion pour chacun des trois modèles.

### A. Résultats obtenus pour le modèle 1 :

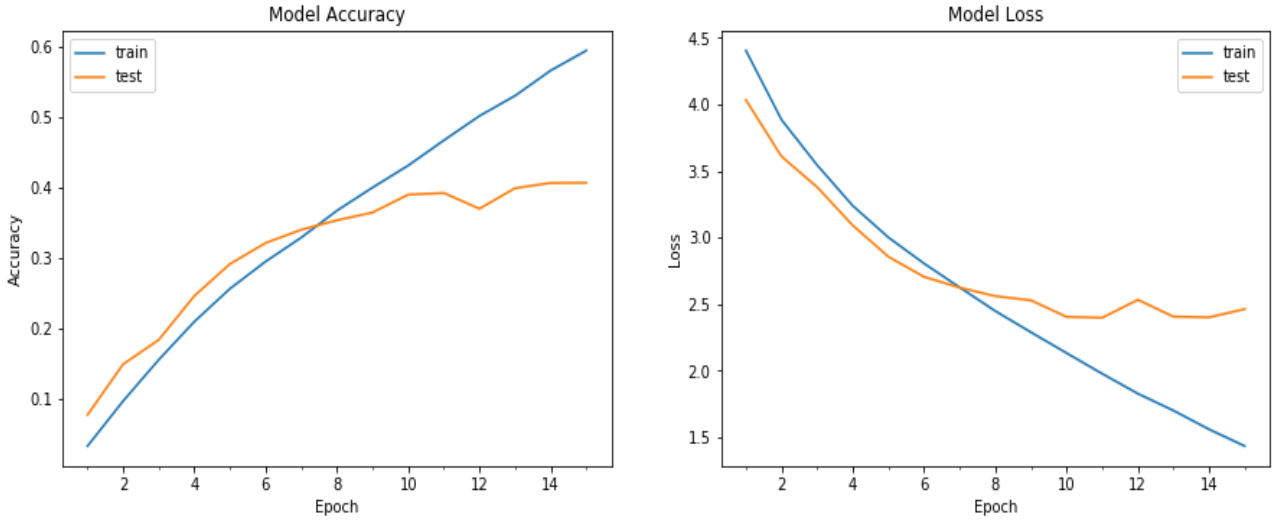



**Figure III.6** Précision et Erreur pour le Modèle 01

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

D'après **la Figure III.6** La précision de l'apprentissage et de test augmente avec le nombre d'époque, ceci reflète qu'à chaque époque le modèle apprenne plus d'informations. Si la précision est diminuée alors on aura besoin de plus d'information pour faire apprendre notre modèle et par conséquent on doit augmenter le nombre d'époque et vice versa.

De même, l'erreur d'apprentissage et de la validation diminue avec le nombre d'époque.

Nous remarquons aussi que la totalité des images mal classées est de 7977 images, un taux d'erreur de 79,77% et la totalité des images bien classées est de 2023 un taux de précision de 20,23%.

### B. Résultats obtenus pour le modèle 2 :

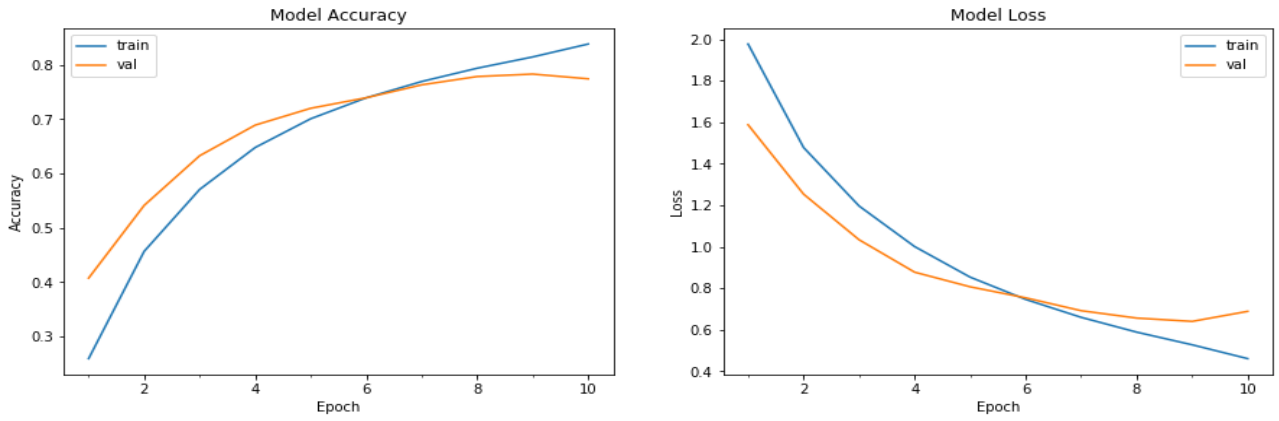



**Figure III.7** Précision et Erreur pour le Modèle 02

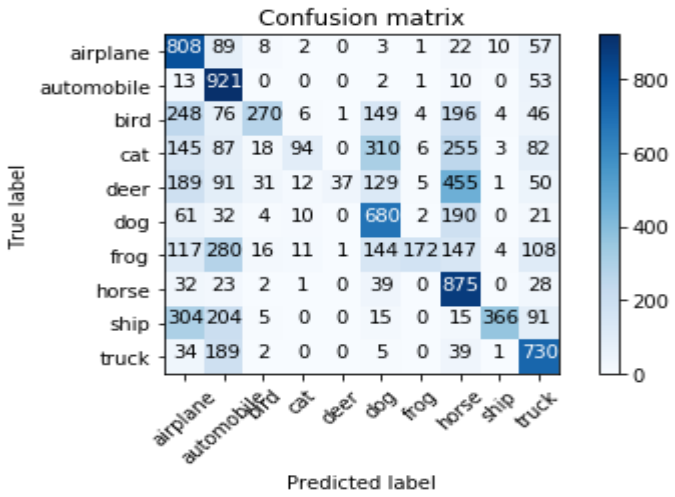
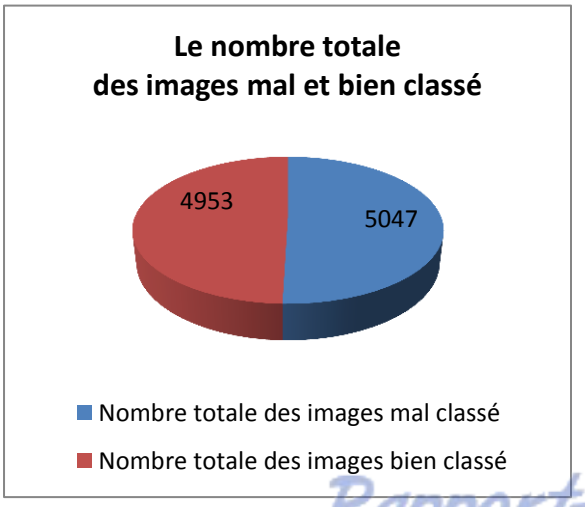



**Figure III.8** Matrice de Confusion pour le Modèle 02



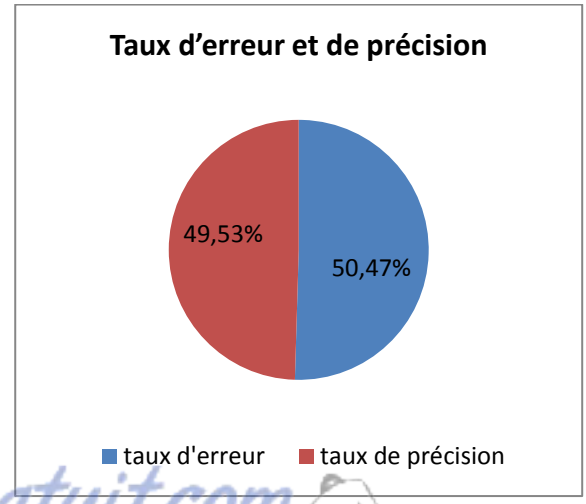




**Figure III.9** nombre total des images mal et bien classé

**Figure III.10** Taux d'erreur et de précision

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MEMOIRES

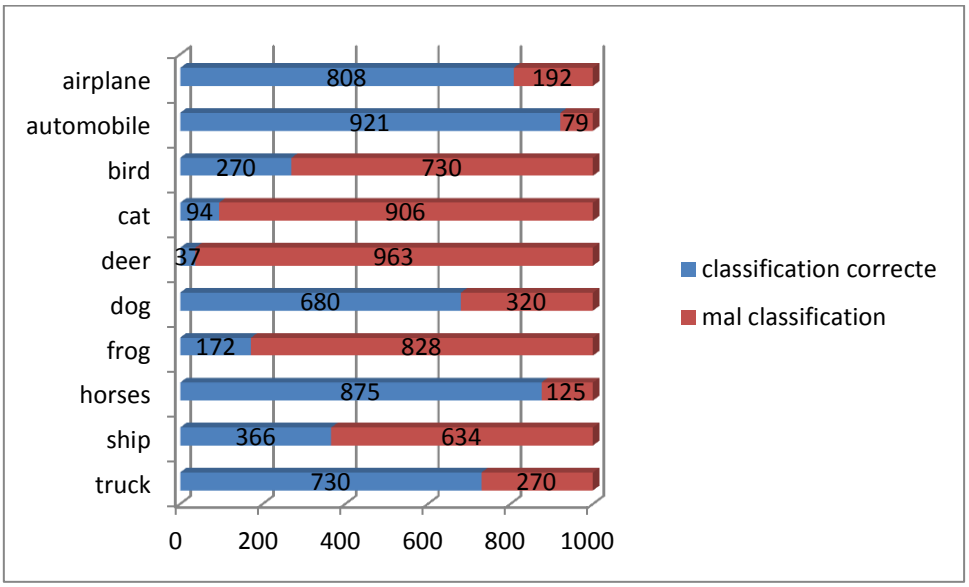



**Figure III.11** Nombre des images mal et bien classé par classe

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

D'après **la Figure III.7** La précision de l'apprentissage et de la validation augmente avec le nombre d'époque, ceci reflète qu'à chaque époque le modèle apprenne plus d'informations. Si la précision est diminuée alors on aura besoin de plus d'information pour faire apprendre notre modèle et par conséquent on doit augmenter le nombre d'époque et vice versa.

De même, l'erreur d'apprentissage et de la validation diminue avec le nombre d'époque.

D'après **la figure III.9** nous remarquons que la totalité des images mal classées est de 5047 images, un taux d'erreur de 50,47% et la totalité des images bien classées est de 4953 un taux de précision de 49,53%.

La matrice de confusion permet d'évaluer la performance de notre modèle, puisqu'elle reflète les métriques du Vrai positif, Vrai négatif, Faux positif et Faux négatif. **La figure III.8** illustre de près la position de ces métriques pour chaque classe. A titre d'exemple le modèle a bien classé les images automobiles, horses et airplane et il a mal classé les images deer, cat et frog

### C. Résultats obtenus pour le modèle 3 :

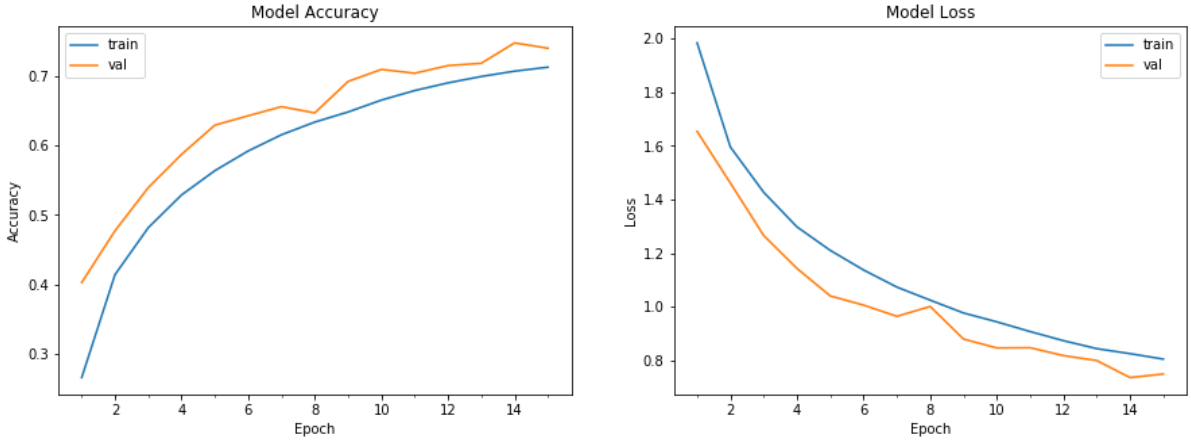



**Figure III.12** Précision et Erreur pour le modèle 03

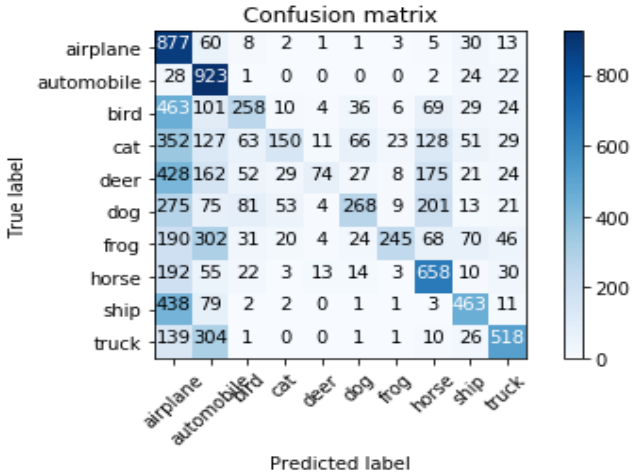



**Figure III.13** Matrice de confusion pour le modèle 03

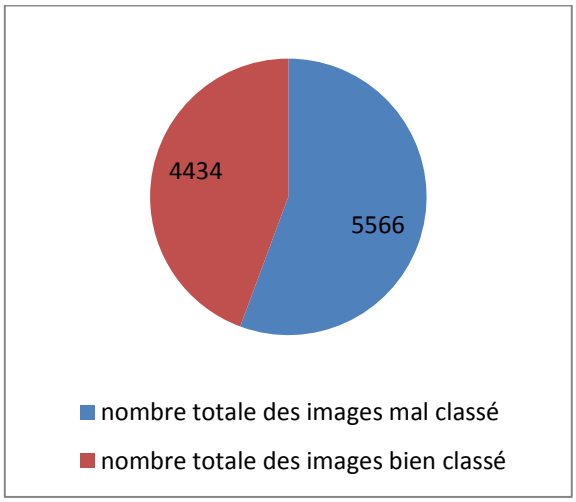



**Figure III.14** nombre total des images mal et bien classé

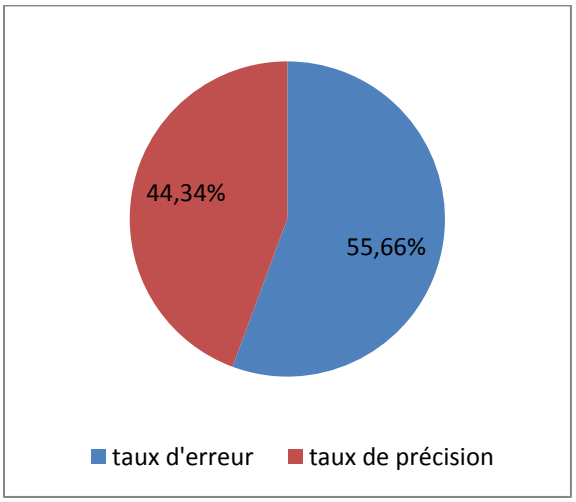



**Figure III.15** Taux d'erreur et de précision

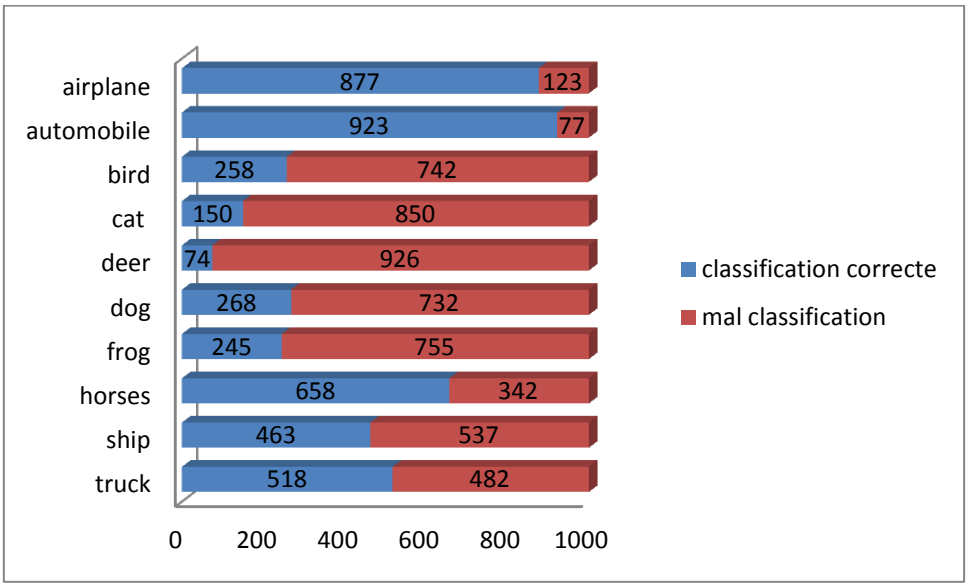



**Figure III.16** Nombre des images mal et bien classé par classe

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

D'après **la Figure III.12** La précision de l'apprentissage et de la validation augmente avec le nombre d'époque, ceci reflète qu'à chaque époque le modèle apprenne plus d'informations. Si la précision est diminuée alors on aura besoin de plus d'information pour faire apprendre notre modèle et par conséquent on doit augmenter le nombre d'époque et vice versa.

De même, l'erreur d'apprentissage et de la validation diminue avec le nombre d'époque.

D'après **la figure III.14** nous remarquons que la totalité des images mal classées est de 5566 images, un taux d'erreur de 55,65% et la totalité des images bien classées est de 4434 un taux de précision de 44,34%.

La matrice de confusion permet d'évaluer la performance de notre modèle, puisqu'elle reflète les métriques du Vrai positif, Vrai négatif, Faux positif et Faux négatif. **La figure III.13** illustre de près la position de ces métriques pour chaque classe. A titre d'exemple le modèle a bien classé les images automobiles et airplane et il a mal classé les images deer et cat.

## D. Tableau de comparaison des résultats

Le tableau ci-dessous montre les différents résultats obtenus sur les trois modèles

| | Architecture Utilisé | | | Nombre époque | Précision obtenu Sur la base d'apprentissage | Précision obtenu Sur la base de validation | Erreur | Temps d'exécution |
|---|---|---|---|---|---|---|---|---|
| | Couche de Convolution | Couche de Pooling | Fully connected | | | | | |
| Modèle 01 | 04 | 02 | 03 | 15 | 60,09% | 40,10 % | 79,77% | 60575 secondes (16 heures) |
| Modèle 02 | 06 | 03 | 03 | 10 | 83.85% | 77.44 % | 50,47% | 42594 secondes (12 heures) |
| Modèle 03 | 04 | 02 | 02 | 15 | 71,29 % | 74,03% | 55,66% | 54242 secondes (15 heures) |

Le tableau montre l'architecture utilisée dans chaque modèle ainsi que le nombre d'époque. Les résultats obtenus sont exprimés en termes de précision d'apprentissage, de validation, de test et erreur et enfin de temps d'exécution. Le temps d'exécution est trop couteux. Ceci revient à la grande dimension de la base ce qui nécessite l'utilisation d'un GPU au lieu d'un CPU. Le modèle 2 a présenté les meilleurs résultats trouvés. Le nombre d'époque et de couches de convolution reflètent ces bons résultats, cependant le temps d'exécution était très couteux (à cause du nombre d'époque).

D'une manière générale, un réseau de neurone convolutionnel important et profond donne des bons résultats et la performance de notre réseau se dégrade si une couche convolutive est supprimée. Par exemple, d'après le tableau L'élimination de l'une des deux couches intermédiaires entraîne une perte d'environ 5% pour la performance du réseau. Donc, la profondeur est primordiale pour atteindre des bons résultats.

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

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

## Code Source du modèle 02

```
filtre = 3                    #taille de filtre
strides=1                     #le pas de convolution
convolution1=32               # nombre de filtre (32 features maps)
convolution2 = 64             # nombre de filtre (64 features maps)
convolution3=128              # nombre de filtre (128 features maps)
pool_size=2                   # taille de pool
dropout1 = 0.25               #appliquer un dropout avec probabilité de 25 %
dropout2=0.50                 #appliquer un dropout avec probabilité de 50 %
couchecache1=1024             #créer une première couche de 1024 neurones
couchecache2=1024             # créer une deuxième couche de 1024 neurones
(x_train, y_train), (x_test, y_test) = cifar10.load_data()    # charger la base d'image de cifar10
num_train, depth, height, width = x_train.shape    #50000 images d'apprentissages
num_test = x_test.shape[0]                    #10000 images de test
num_classes = np.unique(y_train).shape[0]     # 10 classes
input_img= Input(shape=(depth, height, width)) # image d'entrée de taille 3*32*32
1 : conv_1 = Convolution2D(convolution1, filtre, filtre, border_mode='same', activation='relu')(input_img)
2 : conv_2 = Convolution2D(convolution1,filtre, filtre, border_mode='same', activation='relu')(conv_1)
```

**1 :** Cette commande permet de créer 32 features maps on utilisant un filtre de taille 3 par 3 pixels et avec un mode de bordure égal à la taille de l'image précédente et une fonction d'activation de type RELU.

**2 :** Cette commande permet de créer 32 features maps on utilisant un filtre de taille 3 par 3 pixels et avec un mode de bordure égal à la taille de l'image précédente et une fonction d'activation de type RELU.

```
3: pool_1 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_2)
```

**3 :** Cette ligne de commande permet de réduire la taille de l'image, La méthode Max Pooling est utilisée et la taille de l'image sera divisée sur 2.

```
4 : conv_3 = Convolution2D(convolution2,filtre,filtre, border_mode='same', activation='relu')(pool_1)
5 : conv_4 = Convolution2D(convolution2,filtre,filtre, border_mode='same', activation='relu')(conv_3)
```

**4 :** Cette commande permet de créer 64 features maps on utilisant un filtre de taille 3 par 3 pixels et avec un mode de bordure égal à la taille de l'image précédente et une fonction d'activation de type RELU.

**5 :** Cette commande permet de créer 64 features maps on utilisant un filtre de taille 3 par 3 pixels et avec un mode de bordure égal à la taille de l'image précédente et une fonction d'activation de type RELU.

**6: pool_2 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_4)**

**6 :** Cette ligne de commande permet de réduire la taille de l'image, La méthode Max Pooling est utilisée et la taille de l'image sera divisée sur 2.

**7 : conv_5 = Convolution2D(convolution3,filtre,filtre, border_mode='same', activation='relu')(pool_2)**

**8 : conv_6 = Convolution2D(convolution3,filtre,filtre, border_mode='same', activation='relu')(conv_5)**

**7 :** Cette commande permet de créer 128 features maps on utilisant un filtre de taille 3 par 3 pixels et avec un mode de bordure égal à la taille de l'image précédente et une fonction d'activation de type RELU.

**8 :** Cette commande permet de créer 128 features maps on utilisant un filtre de taille 3 par 3 pixels et avec un mode de bordure égal à la taille de l'image précédente et une fonction d'activation de type RELU.

**9: pool_3 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_6)**

**9 :** Cette ligne de commande permet de réduire la taille de l'image, La méthode Max Pooling est utilisée et la taille de l'image sera divisée sur 2.

**10 : flat = Flatten()(pool_3)**

**10 :** Cette commande permet de créer un seul vecteur 1D puis connecter avec la première couche cachée pour commencer la classification.

**11 : couchecache1= Dense(couchecache1, activation='relu')(flat)**

**11 :** Cette commande permet de créer une couche cachée avec une taille de 1024 neurones, la fonction RELU est utilisé comme fonction d'activation.

**12 : drop_1= Dropout(dropout1)(couchecache1)**

**12 :** Pour ne pas tomber dans le problème de sur apprentissage il faut utiliser dropout elle est très efficace pour les réseaux de neurones pour le régulariser et elle n'a besoin que de deux paramètres pour être défini, dont

- Le paramètre type avec pour valeur 'dropout' ;
- Le paramètre rate avec pour valeur 0.25 et 0.5.

**13 : couchecache2= Dense(couchecache2, activation='relu')(drop_1)**

**13 :** Cette commande permet de créer une couche cachée avec une taille de 1024 neurones, la fonction RELU est utilisé comme fonction d'activation.

**14 : drop_2= Dropout(dropout2)(couchecache2)**

**14 :** Appliquer dropout sur la couche caché 2

**15 : sortie= Dense(num_classes, activation='softmax')(drop_2)**

**15 :** Cette commande permet de créer une couche de sortie composée de 10 neurones (nombre de classes) la fonction softmax est utilisé pour calculer la probabilité de chaque classe.

**16: Sgd=SGD(lr=0.01,decay=1e-6,momentum=0.9,nesterov=True)**

**17: model.compile(loss='categorical_crossentropy', optimizer=sgd)**

**16:** Cette commande permet de définir optimizer à utiliser, on a choisi sgd (Stochastic Gradient Descent)

**17:** Cette commande permet de compiler notre modèle, elle prend deux paramètres la fonction loss et Opimizer. On a choisi la fonction categorical_crossentropy comme fonction loss et sgd comme optimizer

**18: model_info = model.fit(x_train, y_train, batch_size=batch_size, nb_epoch=num_epochs, verbose=1, validation_split=0.2)**

**18:** Cette commande permet de lancer l'apprentissage puis la validation

**19: score = model.evaluate(x_test, y_test, verbose=0) print('Test score:', score[0]) print('Test accuracy:', score[1])**

**19:** Cette commande permet d'évaluer notre modèle sur la base de test

## III.6 Conclusion

Nous avons présenté dans ce chapitre une approche de classification basée sur les réseaux de neurones convolutionnels, pour cela on a utilisé trois modèles avec différents architectures et on a montré les différents résultats obtenus en termes de précision et d'erreur. La comparaison des résultats trouvés a montré que le nombre d'époque, la taille de la base et la profondeur de réseaux, sont des facteurs importants pour l'obtention de meilleurs résultats.

## Conclusion générale

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

Les paramètres du réseau sont difficiles à définir a priori. C'est pour cette raison que nous avons défini différents modèles avec des architectures différentes afin d'obtenir des meilleurs résultats en terme de précision et d'erreur.

Nous avons rencontré quelques problèmes dans la phase d'implémentation, l'utilisation d'un CPU a fait que le temps d'exécution était trop couteux. Afin de régler ce problème on doit utiliser des réseaux de neurones convolutionnels plus profonds déployé sur un GPU au lieu d'un CPU sur des bases plus importantes.

Rapport-Gratuit.com

## Bibliographie

[01] Chesner Desir, Classification Automatique d'Images, Application à l'Imagerie du Poumon Profond, Université de Rouen, 2013

[02] Naciri H., Chaoui N, Conception et Réalisation d'un système automatique d'identification des empreintes digitales, Mémoire de PFE, Université de Tlemcen, 2003.

[03] Hadjila F. & Bouabdellah R, Reconnaissance des visages par les réseaux de neurones, Mémoire de PFE, Université de Tlemcen, 2003.

[04] Rafael C.Gonzalez & Richard E.Woods , Digital Image Processing, Pearson Education Inc, 2008.

[05] https://www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-m-Intro-ApprentStat.pdf

[06]https://msdn.microsoft.com/big_data_france/2014/06/17/evaluer-un-modle-en-apprentissage-automatique/

[07] Yann LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[08] http://deeplearning.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/

[09] D. H. Hubel and T. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. Physiol, 160 :106–154, 1962.

[10] K. Fukushima. Neocognitron : A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics, 36 :193–202, 1980.

[11] K. Fukushima. A neural network model for selective attention in visual pattern recognition. Biological Cybernetics, 55 :5–15, 1986.

[12] K. Fukushima. Analysis of the process of visual pattern recognition by the neocognitron. Neural Networks, 2(6) :413–420, 1989.

[13] K. Fukushima and T. Imagawa. Recognition and segmentation of connected characters with selective attention. Neural Networks, 6(1) :33–41, 1993.

[14] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In Parallel distributed processing : Explorations in the microstructure of cognition, Volume 1 : Foundations. MIT Press, 1986.

[15] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In D. Touretzky, editor, Neural Information Processing Systems, volume 2. Morgan Kaufman, 1990.

[16] Y. Le Cun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of IEEE, 86(11) :2278–2324, November 1998.

[17] Pierre Buyssens, Fusion de différents modes de capture pour la reconnaissance du visage appliquée aux e_ transactions DOCTORAT de l'UNIVERSIT´E de CAEN Le 4 Janvier 2011

[18] Kadous Djamila, Utilisation des réseaux de neurones comme outil du datamining : Génération de modèle comportemental d'un processus physique à partir de données

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

## Résumé

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

## ملخص

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

## Abstract

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