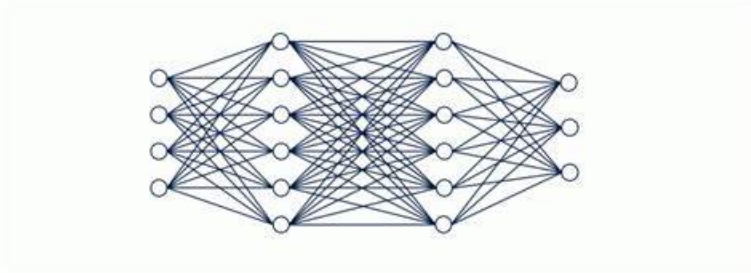


les Réseaux de Neurones et Développement d'une Application de Reconnaissance Faciale

Module : Calcul Scientifique

Spécialité : Master 2 Ingénierie des Systèmes Complexes

Session : 1er Semestre 2024/2025

Auteur : Abdelouahed AIT RAHOU

Encadrant : Mr. Khalid JBILOU

Ecole d'ingénieur du Littorale Côté d'Opale

15 Janvier 2025

Table des matières

Introduction **5**

1 Introduction aux réseaux de neurones **7**
1.1 Définition et context 8
1.2 Importance en machine learning 9
1.2.1 Machine learning 9
1.2.2 Types machine learning 9
1.2.3 Apprentissage supervisé vs non supervisé 11
1.3 Brève histoire du développement des réseaux de neurones 15

2 Concepts de base **17**
2.1 Neurones et synapses 17
2.2 Fonctions d'activation 17
2.2.1 sigmoïde 18
2.2.2 ReLU 19
2.2.3 Avantages de ReLU 20
2.2.4 Inconvénients de ReLU 20

3 Types de réseaux de neurones et Application **22**
3.1 Réseaux de neurones feedforward 22
3.2 Réseaux de neurones convolutifs (CNN) 23
3.3 Réseaux de neurones récurrents (RNN) 24
3.4 Réseaux adversariaux génératifs (GAN) 25
3.5 Applications des réseaux de neurones 26
3.5.1 Reconnaissance d'images 26
3.5.2 Traitement du langage naturel 27
3.5.3 Jeux et stratégies 27
3.5.4 Applications en santé 27

4 Entraînement des réseaux de neurones **28**
4.1 Prétraitement des données 28
4.2 Algorithme de rétropropagation 28
4.3 Fonctions de perte et optimiseurs 31
4.4 Surapprentissage et techniques de régularisation 32
4.4.1 Le surapprentissage 32

4.4.2 La régularisation 33

5 Reconnaissance faciale avec les réseaux de neurones 35
5.1 Vue d'ensemble de la technologie de reconnaissance faciale 35
5.2 Application des CNN en reconnaissance faciale 36
5.2.1 Résultats 36

6 Tendances futures et perspectives de recherche 38

Conclusion 40

Références 41

Table des images

1.1 Structure d'un réseau de neurone biologique et le cerveau humain . 8
1.2 Types machine learning 10
1.3 Illustration de l'apprentissage non supervisé par rapport à l'apprentissage supervisé. Dans les panneaux (a) et (c), l'apprentissage non supervisé tente de trouver des regroupements pour les données afin de les classer en deux groupes. Pour des données bien séparées (a), la tâche est simple et des étiquettes peuvent être facilement produites. Pour des données qui se chevauchent (c), il s'agit d'une tâche très difficile à accomplir pour un algorithme non supervisé. Dans les panneaux (b) et (d), l'apprentissage supervisé fournit un certain nombre d'étiquettes : des boules vertes et des boules magenta. Les données restantes non étiquetées sont ensuite classées comme vertes ou magenta. Pour des données bien séparées (b), l'étiquetage des données est facile, tandis que des données qui se chevauchent présentent un défi significatif. 12
1.4 Les modèles de classification et de régression peuvent s'avérer difficiles lorsque les données comportent des fonctions non linéaires qui les séparent. Dans ce cas, la fonction séparant les boules vertes et magenta peut être difficile à extraire. De plus, si seul un petit échantillon des données D_0 est disponible, il peut alors être impossible de construire un modèle généralisable pour D. Le jeu de données dans le panneau (a) représente deux formes de demi-lunes superposées, tandis que les anneaux concentriques dans le panneau (b) nécessitent un cercle comme frontière de séparation entre les données. Les deux présentent des défis à réaliser. 14

2.1 Fonction sigmoïde et sa dérivée 19

3.1 Réseau de neurones à action directe(feedforward) 23
3.2 Couches de CNN disposées en volumes tridimensionnels 24
3.3 Schéma d'un réseau de neurones récurrents à une unité reliant l'entrée et la sortie du réseau. A droite la version dépliée de la structure. 24
3.4 Architecture d'un GAN 25

3.5 Les architectures de réseaux neuronaux couramment étudiées dans la littérature incluent différents types de structures. Les réseaux de neurones (RN) se composent de nœuds d'entrée, de nœuds de sortie et de nœuds cachés. Ces nœuds peuvent également intégrer des fonctionnalités telles que la mémoire, la convolution, le regroupement (pooling) ou encore la transformation par noyau. Chaque type de réseau et son acronyme sont expliqués dans le texte. 26

4.1 Illustration de l'algorithme de rétropropagation sur un réseau à une couche cachée et un seul nœud. 29
4.2 Gauche : Over fiting - Droite : Proper fiting 33

5.1 Résultats de la détection et classification des émotions à partir des images 36

Introduction

Depuis l'Antiquité, l'humanité rêve de créer des machines capables de penser. Ce désir, visible dans les mythes grecs de Pygmalion, Dédale et Héphaïstos, met en scène des figures légendaires inventant des formes de vie artificielle, telles que Galatée, Talos ou Pandore (Ovid et Martin, 2004 ; Sparkes, 1996 ; Tandy, 1997). Avec l'apparition des ordinateurs programmables, la question de savoir si les machines pouvaient devenir intelligentes a émergé bien avant leur développement (Lovelace, 1842). Aujourd'hui, l'intelligence artificielle (IA) est un domaine en pleine expansion, offrant de nombreuses applications pratiques et ouvrant de nouvelles perspectives de recherche.

Nous attendons des logiciels intelligents qu'ils automatisent des tâches, interprètent la parole ou les images, aident à poser des diagnostics en médecine et soutiennent la recherche scientifique fondamentale. Aux débuts de l'IA, les chercheurs ont résolu des problèmes intellectuellement complexes pour les humains mais formellement simples pour les machines, grâce à des règles mathématiques précises. Cependant, le véritable défi est apparu face aux tâches intuitives pour les humains mais difficiles à formaliser, comme la reconnaissance vocale ou faciale.

Les premiers succès de l'intelligence artificielle (IA) ont principalement été réalisés dans des environnements simples et strictement définis, où les ordinateurs n'avaient pas besoin de connaissances approfondies du monde réel. Un exemple marquant est le système de jeu d'échecs Deep Blue d'IBM, qui a battu le champion du monde Garry Kasparov en 1997 (Hsu, 2002). Bien que la victoire de Deep Blue ait représenté une avancée significative, elle reposait sur un environnement de jeu restreint et formel : les échecs se déroulent sur un plateau de soixante-quatre cases avec trente-deux pièces dont les mouvements sont strictement réglementés. Cette simplicité formelle permet au programmeur de décrire toutes les règles nécessaires en un ensemble fini d'instructions.

Il est ainsi ironique que les tâches mentales abstraites et formelles, très exigeantes pour un humain, soient les plus faciles à automatiser pour un ordinateur. Les machines surpassent depuis longtemps les humains dans des domaines comme les échecs, mais ce n'est que récemment qu'elles commencent à atteindre des compétences comparables à celles de l'humain moyen pour des tâches complexes comme la reconnaissance d'objets ou la compréhension de la parole. Or, pour évoluer dans notre quotidien, il faut une vaste compréhension du monde, largement intuitive et subjective, rendant difficile sa formalisation pour les ordinateurs.

L'un des principaux défis de l'IA réside donc dans la capacité à intégrer ce type de connaissances informelles et tacites dans les systèmes intelligents.

Ce rapport explore une introduction aux réseaux de neurones, incluant leurs concepts de base, leurs différents types, leurs applications diverses, et une application spécifique à la reconnaissance faciale.

Partie 1

Introduction aux réseaux de neurones

Les premiers succès de l'IA, bien que impressionnants, se déroulaient souvent dans des environnements contrôlés et exigeaient des règles explicites et formelles pour chaque tâche, limitant la polyvalence des systèmes. À mesure que l'intelligence artificielle a évolué, la nécessité de construire des systèmes capables d'apprendre et de s'adapter est devenue essentielle pour relever des défis comme la reconnaissance d'images ou la compréhension de la langue naturelle, qui demandent une compréhension bien plus nuancée du monde. C'est dans ce contexte que les réseaux de neurones artificiels, inspirés par la structure biologique du cerveau, ont émergé.

Les réseaux de neurones offrent une approche radicalement différente : au lieu d'exécuter des instructions précises, ils permettent aux ordinateurs d'apprendre par l'expérience en traitant de grandes quantités de données.

Dans cette partie, nous aborderons la définition et le contexte des réseaux de neurones, en explorant leur importance dans le domaine de l'apprentissage automatique. Enfin, nous présenterons une brève histoire du développement des réseaux de neurones, retraçant les étapes clés qui ont conduit à leur utilisation actuelle pour des applications variées et complexes.

1.1 Définition et context

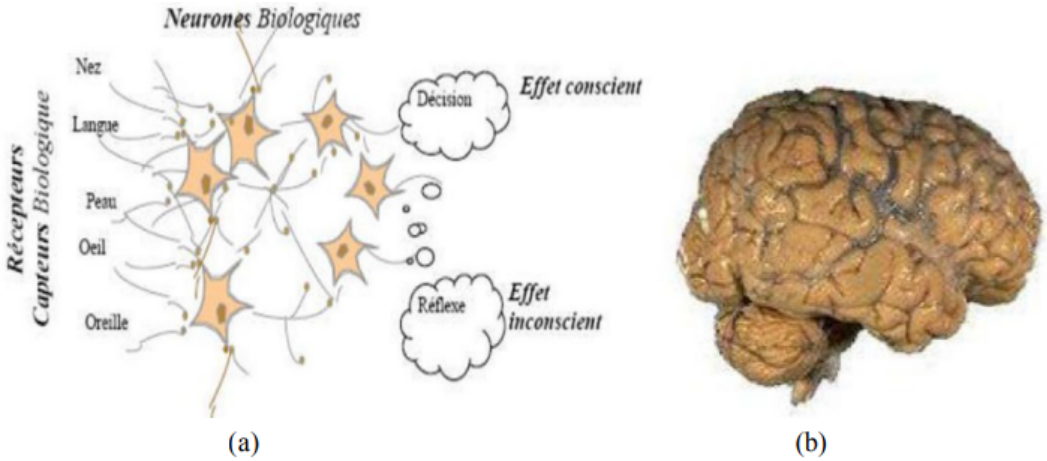


IMAGE 1.1 – Structure d'un réseau de neurone biologique et le cerveau humain

L'origine des réseaux de neurones vient de l'essai de modélisation mathématique du cerveau humain les premiers travaux datent de 1943 et sont l'œuvre de W.M. Culloch et W. Pitts. Ils supposent que l'impulsion nerveuse est le résultat d'un calcul simple effectué par chaque neurone et que la pensée naît grâce à l'effet collectif d'un réseau de neurones interconnectés (figure 1).
Un réseau de neurones est un assemblage de constituants élémentaires interconnectés (appelés « neurones » en hommage à leur modèle biologique), qui réalisent chacun un traitement simple mais dont l'ensemble en interaction fait émerger des propriétés globales complexes. Chaque neurone fonctionne indépendamment des autres de telle sorte que l'ensemble forme un système massivement parallèle. L'information est stockée de manière distribuée dans le réseau sous forme de coefficients synaptiques ou de fonctions d'activation, il n'y a donc pas de zone de mémoire et de zone de calcul, l'une et l'autre sont intimement liées.
Un réseau de neurones ne se programme pas, il est entraîné grâce à un mécanisme d'apprentissage. Les tâches particulièrement adaptées au traitement par réseau de neurones sont : l'association, la classification, la discrimination, la prévision ou l'estimation, et la commande de processus complexes.

Les réseaux de neurones sont un type de modèle en machine learning inspiré de la structure et du fonctionnement du cerveau humain. Ils sont constitués de neurones artificiels organisés en couches (entrée, cachée, et sortie) et permettent de capturer des relations complexes dans les données. Bien que les réseaux de neurones soient une technique de machine learning, ils ont une structure particulière qui les rend particulièrement performants pour des tâches complexes comme la reconnaissance d'image et de voix.

Les réseaux de neurones sont un type de modèle en machine learning inspiré de la structure et du fonctionnement du cerveau humain. Ils sont constitués de neurones artificiels organisés en couches (entrée, cachée, et sortie) et permettent de capturer des relations complexes dans les données. Bien que les réseaux de neurones soient une technique d'apprentissage automatique (machine learning), ils ont une structure particulière qui les rend particulièrement performants pour des tâches complexes comme la reconnaissance d'image et de voix.

1.2 Importance en machine learning

Le machine learning, ou apprentissage automatique, est un domaine d'intelligence artificielle en pleine expansion ces dernières années. De plus en plus d'entreprises et d'organisations cherchent à intégrer des algorithmes d'apprentissage automatique dans leurs processus et leurs produits. Et au sein de cet écosystème en évolution rapide, les réseaux de neurones artificiels, ou réseaux neuronaux, jouent un rôle central.
Qu'est-ce que le machine learning ?

1.2.1 Machine learning

Apprentissage (machine learning) = discipline visant à la construction de règles d'inférence et de décision pour le traitement automatique des données.

Le machine learning désigne la capacité pour une machine d'apprendre à partir de données, sans être explicitement programmée pour une tâche. Grâce à des algorithmes statistiques et mathématiques, un système d'intelligence artificielle est capable d'identifier des tendances et des motifs dans de grands ensembles de données, et d'en tirer des modèles prédictifs.

1.2.2 Types machine learning

Il existe trois grands types machine learning :

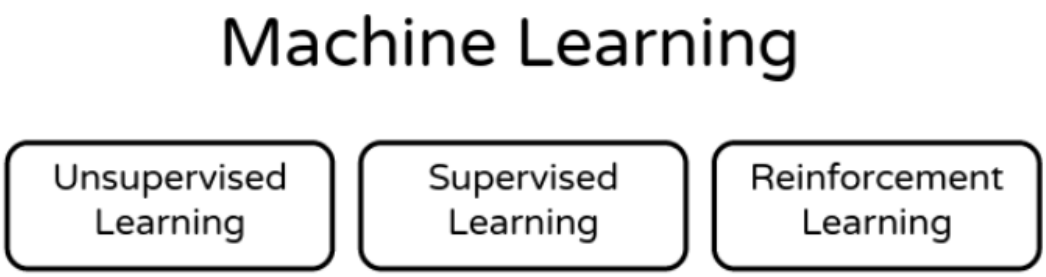


IMAGE 1.2 – Types machine learning

— **L'apprentissage supervisé** : où l'algorithme est entraîné à partir de données annotées et labellisées.

> **Apprentissage supervisé :**
> À partir d'un échantillon d'apprentissage $D_n = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, inférer la relation entre x et y.
> Synonymes : discrimination, reconnaissance de formes
> Voc : x_i = caractéristique = feature = variable explicative

— **L'apprentissage non supervisé** : où l'algorithme doit trouver par lui-même les structures dans des données non labellisées.

> **Apprentissage non supervisé :**
> À partir d'un échantillon d'apprentissage $D_n = \{x_1, \ldots, x_n\} \subset X$, partitionner X en classes pertinentes.
> Voc : parfois appelé 'Classification' en français (jamais en anglais)

— **L'apprentissage par renforcement** : où un agent apprend par essais-erreurs en interagissant avec un environnement.

> **À chaque date n, prendre une décision à l'aide des données passées.**

Les capacités d'apprentissage automatique sont aujourd'hui utilisées dans des applications très variées : moteurs de recommandation, détection de fraudes, véhicules autonomes, diagnostic médical, etc.

Les réseaux neuronaux artificiels sont des algorithmes d'apprentissage automatique dont la structure et le fonctionnement s'inspirent du cerveau humain. Ils sont composés d'un ensemble de nœuds interconnectés, appelés neurones artificiels, organisés en couches.

Chaque neurone reçoit des données depuis les neurones de la couche précédente, effectue des opérations mathématiques simples dessus, et transmet le résultat aux neurones de la couche suivante. C'est en faisant transiter les données dans ce réseau de neurones et en ajustant les paramètres des nœuds que l'algorithme parvient à « apprendre » à partir des exemples.

1.2.3 Apprentissage supervisé vs non supervisé

L'objectif de l'exploration de données et de l'apprentissage automatique est de développer et d'exploiter les caractéristiques essentielles d'un ensemble de données pour créer des modèles prédictifs et des classificateurs. Deux grandes catégories d'apprentissage existent : supervisé et non supervisé.

L'apprentissage supervisé utilise des données étiquetées pour entraîner des modèles basés sur des exemples clairs d'entrées et de sorties, souvent à travers des méthodes de régression. Des variantes comme l'apprentissage semi-supervisé et l'apprentissage par renforcement enrichissent cette approche.

En contraste, l'apprentissage non supervisé travaille sans étiquettes préétablies, cherchant à identifier des structures ou des groupes au sein des données de manière autonome. L'efficacité de ces méthodes dépend largement de la séparation des données ; des distributions bien distinctes simplifient la tâche, tandis que des chevauchements importants la compliquent.

Soit :

$$D \subset \mathbb{R}^n$$

de sorte que D est un ensemble ouvert borné de dimension n. De plus, soit :

$$D' \subset D$$

Le but de la classification est de construire un classificateur étiquetant toutes les données dans D données. Pour rendre notre énoncé du problème plus précis, considérons un ensemble de points de données $x_j \in \mathbb{R}^n$ et étiquette y_j pour chaque point, où $j = 1, 2, \ldots, m$. Étiquettes pour les données peuvent prendre de nombreuses formes, depuis les valeurs numériques, y compris les étiquettes entières, jusqu'au texte cordes. Pour plus de simplicité, nous étiquetterons les données de manière binaire comme étant soit plus un ou moins un, de sorte que

$$y_j \in \{1, -1\}$$

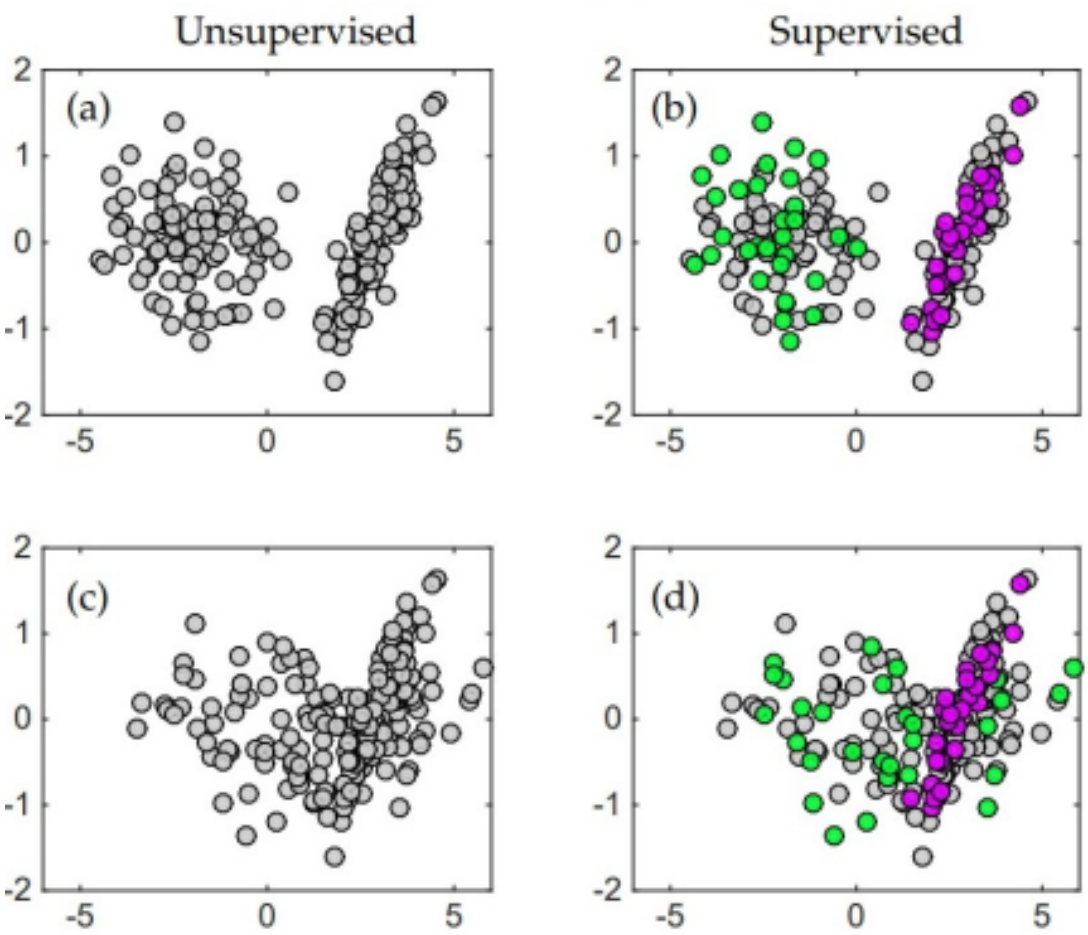


Image 1.3 – Illustration de l'apprentissage non supervisé par rapport à l'apprentissage supervisé. Dans les panneaux (a) et (c), l'apprentissage non supervisé tente de trouver des regroupements pour les données afin de les classer en deux groupes. Pour des données bien séparées (a), la tâche est simple et des étiquettes peuvent être facilement produites. Pour des données qui se chevauchent (c), il s'agit d'une tâche très difficile à accomplir pour un algorithme non supervisé. Dans les panneaux (b) et (d), l'apprentissage supervisé fournit un certain nombre d'étiquettes : des boules vertes et des boules magenta. Les données restantes non étiquetées sont ensuite classées comme vertes ou magenta. Pour des données bien séparées (b), l'étiquetage des données est facile, tandis que des données qui se chevauchent présentent un défi significatif.

Pour l'apprentissage non supervisé, les entrées et sorties suivantes sont alors associées à l'apprentissage d'une tâche de classification :

Input

data $\{x_j \in \mathbb{R}^n, j \in \mathbb{Z} = \{1, 2, ..., m\}\}$

Output

labels $\{y_j \in \{1, -1\}, j \in \mathbb{Z} = \{1, 2, ..., m\}\}$

Ainsi, la structuration mathématique de l'apprentissage non supervisé se concentre sur la production d'étiquettes y_j pour toutes les données. Généralement, les données x_j utilisées pour entraîner le classificateur proviennent de D'. Le classificateur est ensuite appliqué de manière plus large, c'est-à-dire qu'il se généralise, au domaine ouvert et borné D. Si les données utilisées pour construire un classificateur ne représentent qu'une petite partie du domaine plus large, il arrive souvent que le classificateur ne se généralise pas bien. L'apprentissage supervisé fournit des étiquettes pour la phase d'entraînement. Les entrées et les sorties pour cette tâche de classification par apprentissage peuvent être énoncées comme suit :

Input

data $\{x_j \in \mathbb{R}^n, j \in \mathbb{Z} = \{1, 2, ..., m\}\}$, labels $\{y_j \in \{1, -1\}, j \in \mathbb{Z}' \subset \mathbb{Z}\}$

Output

labels $\{y_j \in \{1, -1\}, j \in \mathbb{Z} = \{1, 2, ..., m\}\}$

Dans ce cas, un sous-ensemble des données est étiqueté et les étiquettes manquantes sont fournies pour les données restantes. Techniquement parlant, ceci constitue une tâche d'apprentissage semi-supervisé, étant donné que certaines des étiquettes d'entraînement sont manquantes. Pour l'apprentissage supervisé, toutes les étiquettes sont connues afin de construire des classificateur. Le classificateur est ensuite appliqué à D. Comme pour l'apprentissage non supervisé, si les données utilisées pour construire un classificateur ne représentent qu'une petite portion du domaine plus large, il est souvent constaté que le classificateur ne se généralise pas bien. Pour les ensembles de données considérés dans notre section de sélection de caractéristiques et de fouille de données, nous pouvons examiner plus en détail les composants clés nécessaires pour construire un modèle de classification : x_j, y_j, D, et D'. Les données des iris de Fisher de la figure 2.1 sont un exemple classique pour lequel nous pouvons détailler ces quantités. Nous commençons par les données collectées : $x_j = \{$ longueur du sépale, largeur du sépale, longueur du pétale, largeur du pétale$\}$. Ainsi, chaque mesure d'iris contient quatre champs de données, ou caractéristiques, pour notre analyse. Les étiquettes peuvent être l'une des suivantes : $y_j = \{$ setosa, versicolor, virginica$\}$. Dans ce cas, les étiquettes sont des chaînes de texte, et il y en a trois. Notez que, dans notre formulation de l'apprentissage supervisé et non supervisé, il n'y avait que deux sorties binaire, qui étaient étiquetées soit ± 1. Généralement, il peut y avoir de nombreuses étiquettes, et elles sont souvent des chaînes de texte. Enfin, il y a le domaine des données. Pour ce cas,

Données

$$D_0 \in \{150\ \textit{échantillons d'iris} : 50\ \textit{setosa},\ 50\ \textit{versicolor},\ \textit{et}\ 50\ \textit{virginica}\}$$

$$D \in \{\textit{l'univers des iris setosa},\ \textit{versicolor},\ \textit{et virginica}\}$$

Nous pouvons de même évaluer les données sur les chiens et les chats comme suit :

$$x_j = \{64 \times 64 \text{ image} = 4096 \text{ pixels}\}$$

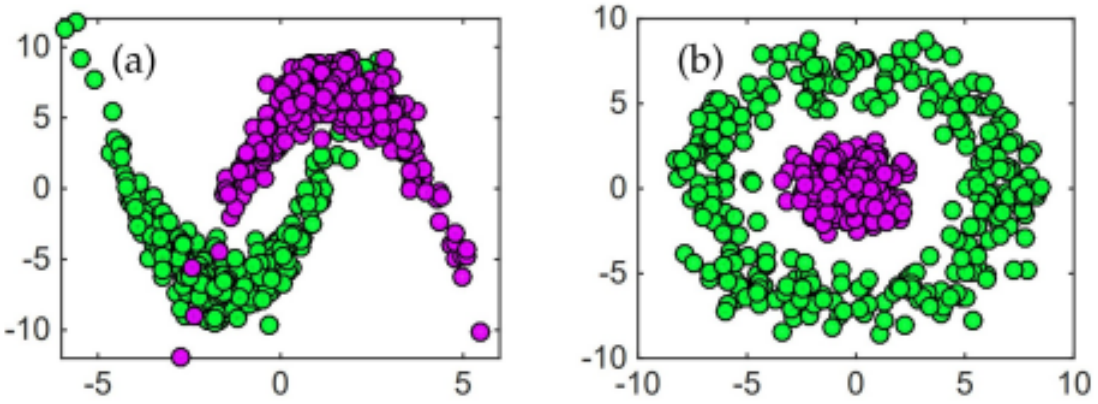


IMAGE 1.4 – Les modèles de classification et de régression peuvent s'avérer difficiles lorsque les données comportent des fonctions non linéaires qui les séparent. Dans ce cas, la fonction séparant les boules vertes et magenta peut être difficile à extraire. De plus, si seul un petit échantillon des données D_0 est disponible, il peut alors être impossible de construire un modèle généralisable pour D. Le jeu de données dans le panneau (a) représente deux formes de demi-lunes superposées, tandis que les anneaux concentriques dans le panneau (b) nécessitent un cercle comme frontière de séparation entre les données. Les deux présentent des défis à réaliser.

Chaque chien et chat est étiqueté comme

$$y_j = \{chien, chat\} = \{1, -1\}$$

Les étiquettes peuvent également être traduites en valeurs numériques, cohérentes avec notre formulation de l'apprentissage supervisé et non supervisé, où il n'y a que deux sorties (binaire), étiquetées soit ± 1. Le domaine des données est :

Données

$$D_0 \in \{160\ \textit{échantillons d'images} : 80\ \textit{chiens et}\ 80\ \textit{chats}\}$$

$$D \in \{\textit{l'univers des chiens et deschats}\}$$

Les méthodes d'apprentissage supervisé et non supervisé visent à créer des algorithmes pour la classification, le regroupement ou la régression. L'objectif est de construire un modèle à partir des données sur D_0 qui peut se généraliser à D. La généralisation peut être très difficile, et les stratégies de validation croisée sont cruciales. Les réseaux de neurones profonds, souvent utilisés pour la régression et la classification, ont du mal à se généraliser. La création de schémas de généralisation solides est au premier plan de la recherche en apprentissage automatique.

1.3 Brève histoire du développement des réseaux de neurones

• **Neurone en biologie.** Entre 1840 et 1900 on découvre que la cellule est l'élément fondamental du vivant ; par observation, Santiago Ramón y Cajal et d'autres mettent en évidence que les cellules du système nerveux forment un réseau : les neurones. On sait maintenant que le cerveau humain contient entre 80 et 100 milliards de neurones, mais nous avons aussi 500 millions de neurones dans le ventre (soit autant que dans le cerveau d'un chien). Un neurone est composé d'un élément central prolongé par un axone au bout duquel se trouvent les synapses. Les neurones sont organisés en réseau, un neurone étant en moyenne relié à 10 000 autres neurones et transmet ainsi un signal électrique à tous ses neurones voisins.

• **De la biologie vers l'informatique.** À la fin des années 1940, Donald Hebb émet l'hypothèse que lors de l'apprentissage certaines connexions synaptiques se renforcent et que ce renforcement persiste. Un modèle mathématique est déduit par Frank Rosenblatt : le perceptron (1958). L'algorithme est mis en œuvre dans une machine dédiée à la reconnaissance d'images. Le perceptron, qui correspond à un réseau formé d'un seul neurone, montre très vite ses limites à la fois théoriques (puisqu'il ne peut réaliser le « ou exclusif ») et aussi en termes de résultats pratiques. Ces deux problèmes sont résolus en considérant des réseaux à plusieurs couches, mais les calculs à mener sont trop longs à la fois en raison de la puissance des ordinateurs de l'époque et des algorithmes utilisés.

• **Rétropropagation.** Un progrès important est fait grâce l'algorithme de rétropropagation (par Rumelhart, Hinton et Williams en 1986) qui permet un calcul efficace des poids des neurones. On conserve encore aujourd'hui ce principe : des calculs essentiellement numériques, avec un minimum de calculs symboliques au niveau de chaque neurone (afin de calculer la dérivée), le tout avec un grand nombre d'itérations. Des améliorations se succèdent : pooling, dropout, calculs en parallèle, meilleures descentes de gradient.

• **Hiver.** Cependant les attentes de l'intelligence artificielle (peut être le nom a-t-il été mal choisi ?) sont largement déçues car ses applications restent limitées. L'intérêt des scientifiques diminue drastiquement. De 1980 à 2000 on parle

de l'hiver de l'intelligence artificielle.

• **Deep learning.** À partir des années 2000 et surtout après 2010 les réseaux de neurones font des progrès fulgurants grâce à l'apprentissage profond. Yann Le Cun démontre l'efficacité des couches de convolution pour la reconnaissance des chiffres. On réalise et entraîne alors des réseaux ayant de plus en plus de couches grâce à des progrès matériels (par exemple le calcul sur les processeurs graphiques GPU) mais surtout grâce aux couches de convolution qui extraient des caractéristiques abstraites des images.

• **Présent et avenir.** Les réseaux de neurones s'appliquent à de nombreux domaines : la reconnaissance d'images (par exemple la détection de cancer sur une radiographie), les transports (par exemple la conduite autonome des voitures), les jeux (les ordinateurs battent les champions du monde d'échecs, de go et des jeux vidéos les plus complexes), l'écriture (classement, résumé, traduction).. . Il persiste cependant une certaine méfiance vis à vis des décisions prises par une machine (sentence de justice, diagnostic médical, publicité ciblée). Une meilleure compréhension du fonctionnement des réseaux de neurones par tous est donc indispensable !

Partie 2

Concepts de base

2.1 Neurones et synapses

Les neurones et les synapses sont des éléments fondamentaux des réseaux de neurones artificiels, inspirés par la structure et le fonctionnement du cerveau humain :

- **Neurone** : Unité de base dans un réseau de neurones, chaque neurone reçoit une ou plusieurs entrées, les pondère, puis applique une fonction d'activation pour générer une sortie. Cette sortie est ensuite transmise aux neurones suivants dans le réseau.
- **Synapse** : Les synapses relient les neurones entre eux dans le réseau. Chaque synapse possède un poids, qui représente l'importance de la connexion. Ces poids sont ajustés lors de l'apprentissage, permettant au réseau de "se souvenir" et d'apprendre à partir des données.

Les neurones et synapses, en imitant les connexions neuronales du cerveau, permettent aux réseaux de neurones d'apprendre des représentations complexes à partir de grandes quantités de données.

2.2 Fonctions d'activation

Une fonction d'activation est une équation mathématique appliquée à un neurone dans un réseau neuronal qui détermine s'il doit être activé ou non.

Cette fonction décide comment transformer l'entrée reçue en un signal de sortie qui est envoyé à la couche suivante de neurones. Essentiellement, les fonctions d'activation introduisent de la non-linéarité dans le réseau, lui permettant d'apprendre des motifs complexes et d'effectuer des tâches au-delà des opérations linéaires simples.

La fonction d'activation, un mécanisme fondamental qui dicte la sortie des neurones au sein d'un réseau neuronal, se trouve au cœur de chaque réseau neuronal.

2.2.1 sigmoïde

La fonction d'activation sigmoïde est largement utilisée dans les réseaux de neurones pour introduire de la non-linéarité. Elle transforme les entrées en une valeur comprise entre 0 et 1, facilitant ainsi l'interprétation des sorties en tant que probabilités. La formule de la fonction sigmoïde est donnée par :

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- **Propriétés** :
  - **Valeurs de sortie** : La fonction produit une sortie entre 0 et 1, ce qui est particulièrement utile pour les tâches de classification binaire.
  - **Non-linéarité** : Grâce à sa courbe en "S", la sigmoïde permet de modéliser des relations complexes dans les données en introduisant de la non-linéarité dans le réseau.
  - **Gradient** : Le gradient de la fonction sigmoïde est plus fort autour de 0, mais devient très faible aux valeurs extrêmes, ce qui peut entraîner un problème de *vanishing gradient* pour les réseaux profonds.
- **Application** : En raison de sa capacité à générer des probabilités, la fonction sigmoïde est couramment utilisée dans la couche de sortie des réseaux de neurones pour les problèmes de classification binaire.

Listing 2.1 – Fonction sigmoïde et sa dérivée en Python

```
import numpy as np
import matplotlib.pyplot as plt

def sigmoid(x):
    return 1 / (1 + np.exp(-x))
def sigmoid_derivative(x):
    return np.exp(-x) / (1+ np.exp(-x))**2
x = np.linspace(-10, 10, 100)
y = sigmoid(x)
y_derivative = sigmoid_derivative(x)
plt.figure(figsize=(10, 6))
plt.plot(x, y, label='sigmoid', color='blue')
plt.plot(x, y_derivative, label="Derivee de la sigmoid",
   color='red', linestyle='--')
plt.xlabel('x')
plt.ylabel('Valeur')
plt.title('Courbes sigmoid/ derivee sigmoid')
plt.legend()
plt.grid()
plt.show()
```

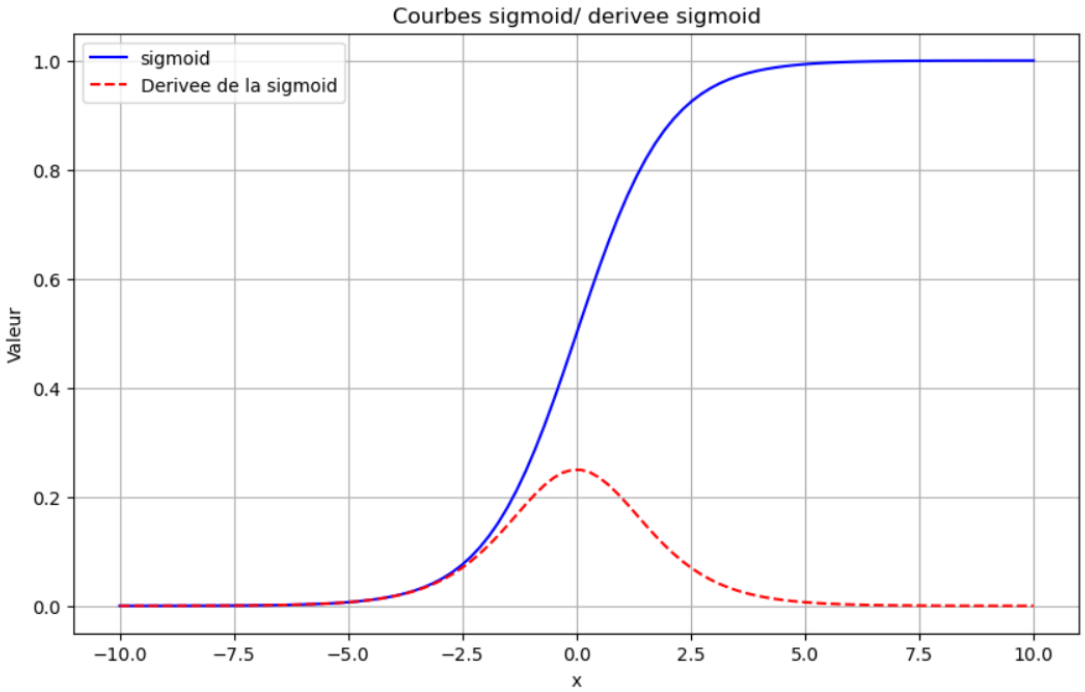


IMAGE 2.1 – Fonction sigmoïde et sa dérivée

2.2.2 ReLU

Cette fonction permet d'appliquer un filtre en sortie de couche. Elle laisse passer les valeurs positives dans les couches suivantes et bloque les valeurs négatives. Ce filtre permet alors au modèle de se concentrer uniquement sur certaines caractéristiques des données, les autres étant éliminées. La fonction d'activation **ReLU** (Rectified Linear Unit) est définie comme suit :

$$y = \max(0, x)$$

Cette fonction retourne x si $x > 0$ et 0 sinon, soit :

$$y = \begin{cases} x & \text{si } x > 0 \\ 0 & \text{sinon} \end{cases}$$

La dérivée est donnée par :

$$y' = f(x) = \begin{cases} 1 & \text{si } x > 0 \\ 0 & \text{sinon} \end{cases}$$

Listing 2.2 – Fonction ReLU et sa dérivée en Python

```
import numpy as np
import matplotlib.pyplot as plt

def relu(x):
    return np.maximum(0, x)

def relu_derivative(x):
    return np.where(x > 0, 1, 0)

x = np.linspace(-10, 10, 100)
y = relu(x)
y_derivative = relu_derivative(x)

plt.figure(figsize=(10, 6))
plt.plot(x, y, label='ReLU', color='blue')
plt.plot(x, y_derivative, label="Derivee de ReLU", color='
   orange', linestyle='--')
plt.xlabel('x')
plt.ylabel('Valeur')
plt.title("Fonction dactivation ReLU et sa derivee")
plt.legend()
plt.grid()
plt.show()
```

2.2.3 Avantages de ReLU

- **Simplicité de calcul** : ReLU est rapide et peu coûteuse à calculer.
- **Réduction du gradient évanescent** : ReLU maintient un gradient constant pour les valeurs positives.
- **Sparsité** : En annulant les valeurs négatives, ReLU introduit une certaine parcimonie.

2.2.4 Inconvénients de ReLU

- **Neurones morts** : Les valeurs négatives d'entrée peuvent bloquer le neurone à zéro.
- **Instabilité pour des valeurs élevées** : Sans régularisation, de grandes valeurs peuvent déstabiliser l'apprentissage.

Il existe d'autres possibilités, mais celles-ci sont peut-être les plus couramment prises en compte en pratique et elles suffiront pour nos besoins. Il est important que la fonction choisie $f(x)$ soit dérivable afin d'être utilisée dans les algorithmes de descente de gradient pour l'optimisation. Chacune des fonctions mentionnées ci-dessus est soit différentiable, soit différentiable par morceaux. La

fonction d'activation la plus couramment utilisée est actuellement la *ReLU*, que nous notons

$$f(x) = \text{ReLU}(x).$$

Avec une fonction d'activation non linéaire $f(x)$, ou s'il y a plus d'une couche, alors les routines d'optimisation linéaire standard telles que le pseudo-inverse et le LASSO ne peuvent plus être utilisées. Bien que cela puisse ne pas sembler immédiatement significatif, rappelons que nous optimisons dans un espace de grande dimension où chaque entrée de la matrice A doit être trouvée par optimisation. Même des problèmes de taille modeste peuvent être coûteux en calcul à résoudre sans utiliser des méthodes d'optimisation spécialisées. Heureusement, les deux composants d'optimisation dominants pour l'entraînement des réseaux de neurones (NN), la descente de gradient stochastique (SGD) et la rétropropagation (*backprop*), sont inclus avec les appels de fonction de réseau neuronal dans `MATLAB`. Comme ces méthodes sont cruciales, elles sont toutes deux considérées en détail dans les deux prochaines sections de ce chapitre.

Plusieurs couches peuvent également être prises en compte comme montré dans l'équation suivante :

$$\overline{A} = A_M \cdots A_2 A_1.$$

Dans ce cas, l'optimisation doit identifier simultanément plusieurs matrices de connectivité $A_1, A_2, \ldots, A_M$, contrairement au cas linéaire où une seule matrice est déterminée. La structure à plusieurs couches augmente considérablement la taille du problème d'optimisation, car chaque élément matriciel des matrices M doit être déterminé. Même pour une structure à une seule couche, une routine d'optimisation telle que `fminsearch` sera sévèrement mise à l'épreuve lors de la considération d'une fonction de transfert non linéaire, et il est nécessaire de passer à un algorithme basé sur la descente de gradient.

La boîte à outils de réseau de neurones de `MATLAB`, tout comme `TensorFlow` en `Python`, dispose d'une large gamme de fonctionnalités, ce qui la rend exceptionnellement puissante et pratique pour construire des NN. Dans le code suivant, nous allons entraîner un NN pour classifier entre chiens et chats comme dans l'exemple précédent. Cependant, dans ce cas, nous permettons à la couche unique d'avoir une fonction de transfert non linéaire qui mappe l'entrée à la couche de sortie. La couche de sortie pour cet exemple sera modifiée comme suit :

$$y = \begin{cases} 1 & \text{pour } \{\text{chien}\}, \\ 0 & \text{pour } \{\text{chat}\}. \end{cases}$$

La moitié des données est extraite pour l'entraînement, tandis que l'autre moitié est utilisée pour tester les résultats. Le code suivant construit un réseau en utilisant la commande d'entraînement pour classer entre nos images.

Partie 3

Types de réseaux de neurones et Application

3.1 Réseaux de neurones feedforward

Les réseaux neuronaux à action directe (feedforward) traitent les données dans une seule direction, du nœud d'entrée au nœud de sortie. Chaque nœud d'une couche est connecté à chaque nœud de la couche suivante. Un réseau à action directe utilise un processus rétroactif pour améliorer les prédictions au fil du temps.

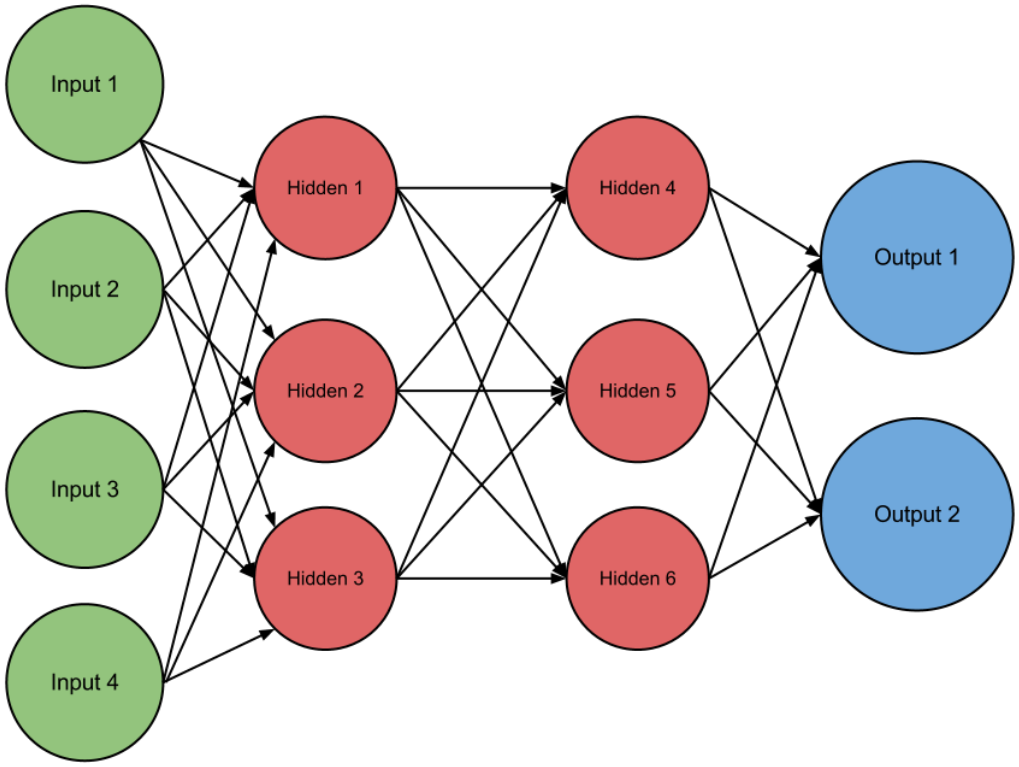


Image 3.1 – Réseau de neurones à action directe(feedforward)

3.2 Réseaux de neurones convolutifs (CNN)

Un réseau neuronal convolutif (ConvNet ou CNN) est un réseau neuronal artificiel (ANN) qui utilise des algorithmes d'apprentissage profond pour analyser des images, classer des éléments visuels et effectuer des tâches de vision par ordinateur.

Le CNN s'appuie sur des principes d'algèbre linéaire, tels que la multiplication des matrices, pour détecter des motifs dans une image. Comme ces processus impliquent des calculs complexes, ils nécessitent des unités de traitement graphique(GPU) pour l'entraînement des modèles.

En d'autres termes, le CNN utilise des algorithmes d'apprentissage profond pour prendre des données d'entrée telles que des images et attribuer de l'importance, sous forme de biais et de poids pouvant être appris, à différents aspects de cette image. De cette manière, le CNN peut différencier les images ou les classer

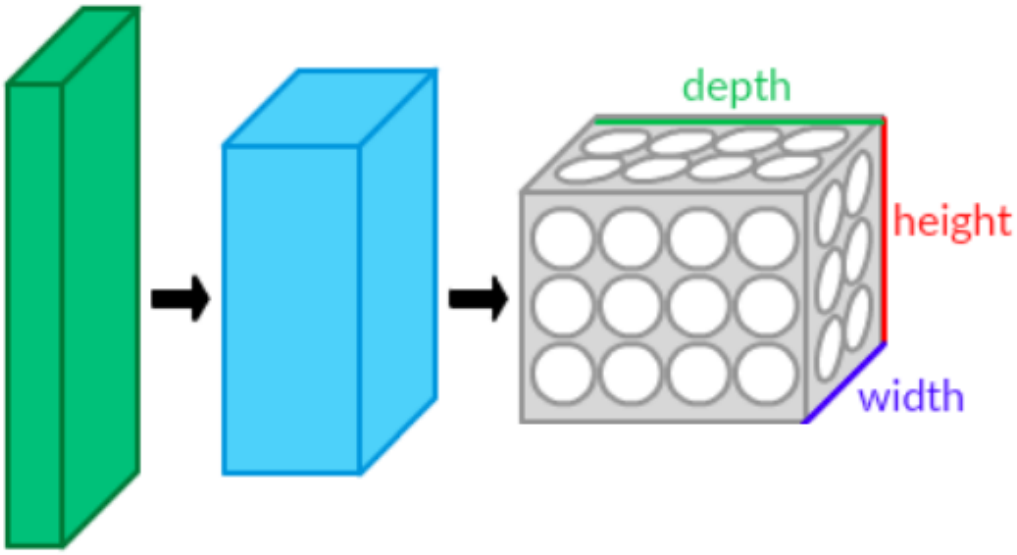


IMAGE 3.2 – Couches de CNN disposées en volumes tridimensionnels

3.3 Réseaux de neurones récurrents (RNN)

Un réseau de neurones récurrents (RNN pour recurrent neural network en anglais) est un réseau de neurones artificiels présentant des connexions récurrentes. Un réseau de neurones récurrents est constitué d'unités (neurones) interconnectées interagissant non-linéairement et pour lequel il existe au moins un cycle dans la structure. Les unités sont reliées par des arcs (synapses) qui possèdent un poids. La sortie d'un neurone est une combinaison non linéaire de ses entrées.

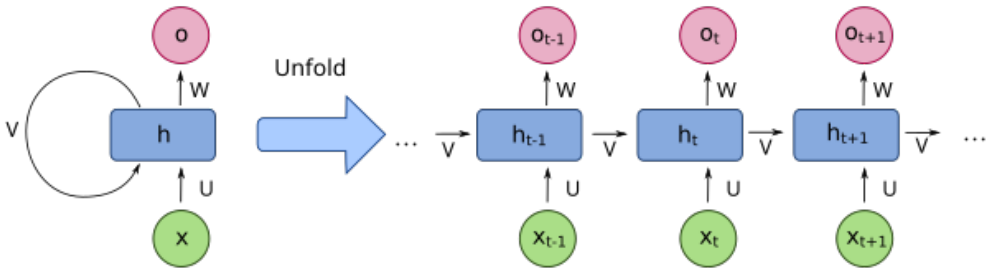


IMAGE 3.3 – Schéma d'un réseau de neurones récurrents à une unité reliant l'entrée et la sortie du réseau. A droite la version dépliée de la structure.

3.4 Réseaux adversariaux génératifs (GAN)

Les **GANs** (Generative Adversarial Networks) sont une approche de la modélisation générative utilisant des méthodes d'apprentissage en profondeur, telles que les réseaux de neurones convolutifs. Les GANs forment un modèle génératif composé de deux sous-modèles :

1. **Modèle générateur** : génère de nouveaux exemples qui ressemblent aux exemples du dataset d'apprentissage.
2. **Modèle discriminateur** : tente de classer les exemples comme réels (dataset d'apprentissage) ou faux (généré).

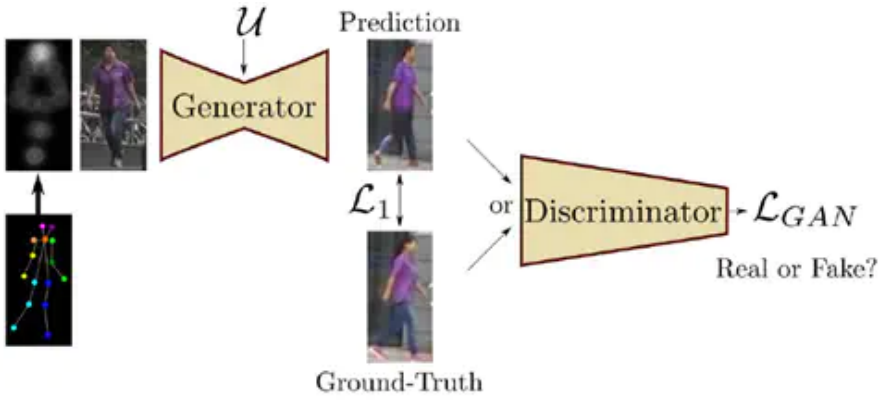


Image 3.4 – Architecture d'un GAN

Ces deux sous-modèles sont entraînés ensemble dans un jeu à somme nulle. Le discriminateur cherche à maximiser ses actions de classification tandis que le générateur tente de minimiser ses actions pour tromper le discriminateur. Ainsi le discriminateur est mis à jour avec la méthode du gradient ascendant (on cherche le maximum global) alors que le générateur est mis à jour avec la méthode du gradient descendant (on cherche le minimum global). L'apprentissage du modèle GAN converge lorsque le discriminateur et le générateur atteignent l'équilibre de Nash qui correspond au point optimal pour le jeu à somme nulle (le générateur ne changera pas son action indépendamment de la décision prise par le discriminateur).

Liste des autres types de réseaux de neurones :

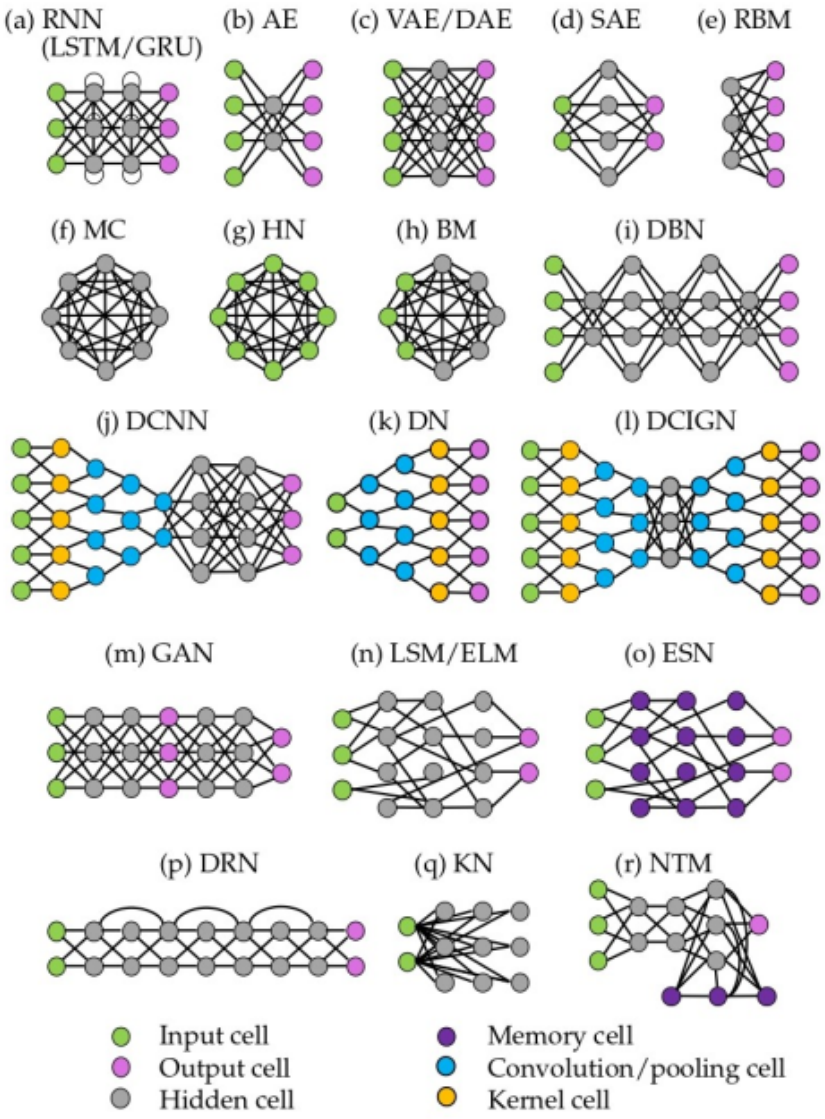


IMAGE 3.5 – Les architectures de réseaux neuronaux couramment étudiées dans la littérature incluent différents types de structures. Les réseaux de neurones (RN) se composent de nœuds d'entrée, de nœuds de sortie et de nœuds cachés. Ces nœuds peuvent également intégrer des fonctionnalités telles que la mémoire, la convolution, le regroupement (pooling) ou encore la transformation par noyau. Chaque type de réseau et son acronyme sont expliqués dans le texte.

3.5 Applications des réseaux de neurones

3.5.1 Reconnaissance d'images

La reconnaissance d'images est une tâche fondamentale en apprentissage automatique, qui consiste à attribuer une étiquette à une image en fonction de son

contenu visuel.

3.5.2 Traitement du langage naturel

Le traitement du langage naturel (TNL) permet à un ordinateur de comprendre, interpréter et générer du langage humain de manière significative. Il est utilisé dans diverses applications telles que les chatbots, la traduction automatique et l'analyse de sentiments.

3.5.3 Jeux et stratégies

Les jeux, qu'ils soient simples ou complexes, sont souvent utilisés pour tester des algorithmes d'intelligence artificielle. Les stratégies développées dans ces contextes permettent de simuler des processus décisionnels complexes.

3.5.4 Applications en santé

L'intelligence artificielle trouve de nombreuses applications dans le secteur de la santé, y compris le diagnostic médical, l'analyse d'images médicales et l'optimisation des traitements.

Partie 4

Entraînement des réseaux de neurones

4.1 Prétraitement des données

Le prétraitement des données est une étape essentielle pour garantir des résultats fiables et une convergence efficace lors de l'entraînement des réseaux de neurones. Il inclut plusieurs étapes telles que :

- La normalisation ou standardisation des données pour assurer que les caractéristiques aient des échelles similaires ;
- La gestion des valeurs manquantes et des outliers pour éviter les biais dans l'entraînement ;
- La transformation des données brutes en formats exploitables par les modèles, comme les vecteurs numériques ou les matrices pour les images ;
- L'application de techniques d'augmentation des données, lorsque nécessaire, pour enrichir le dataset et améliorer la généralisation.

Ces étapes permettent d'optimiser les performances du modèle et de réduire les risques de surapprentissage.

4.2 Algorithme de rétropropagation

En pratique, la fonction objective choisie pour l'optimisation n'est pas la véritable fonction objective désirée, mais plutôt un proxy pour celle-ci. Les proxies sont largement choisis en raison de leur capacité à différencier la fonction objective d'une manière calculatoirement tractable. Il existe également de nombreuses fonctions objectives différentes pour différentes tâches. Au lieu de cela, on considère souvent une fonction de perte convenablement choisie afin d'approximer l'objectif réel. Finalement, la tractabilité computationnelle est critique pour l'entraînement des réseaux de neurones (NN).

L'algorithme de rétropropagation (backprop) exploite la nature compositionnelle des NN afin de formuler un problème d'optimisation pour déterminer les poids

du réseau. Plus précisément, il produit une formulation propice à l'optimisation standard par descente de gradient (voir Section **??**). En particulier, *backprop* calcule le gradient de l'erreur, qui est ensuite utilisé pour la descente de gradient.

Backprop repose sur un principe mathématique simple : la règle de chaîne pour la différentiation. De plus, il peut être prouvé que le temps de calcul requis pour évaluer le gradient est dans un facteur de 5 du temps requis pour calculer la fonction elle-même. Ceci est connu sous le nom de théorème de Baur–Strassen.

La Figure **??** donne l'exemple le plus simple de *backprop* et comment la descente de gradient doit être effectuée. La relation entrée-sortie pour ce réseau à un seul nœud, avec une couche cachée, est donnée par :

$$y = g(z, b) = g(f(x, a), b). \tag{4.1}$$

Ainsi, étant données les fonctions $f(\cdot)$ et $g(\cdot)$ avec les constantes pondérales a et b, l'erreur de sortie produite par le réseau peut être calculée par rapport à la vérité terrain comme suit :

$$E = \frac{1}{2}(y_0 - y)^2 \tag{4.2}$$

où y_0 est la sortie correcte et y est l'approximation de la sortie par le réseau de neurones.

L'objectif est de trouver a et b pour minimiser l'erreur. La minimisation nécessite :

$$\frac{\partial E}{\partial a} = -(y_0 - y)\frac{\partial y}{\partial z}\frac{\partial z}{\partial a} = 0 \tag{4.3}$$

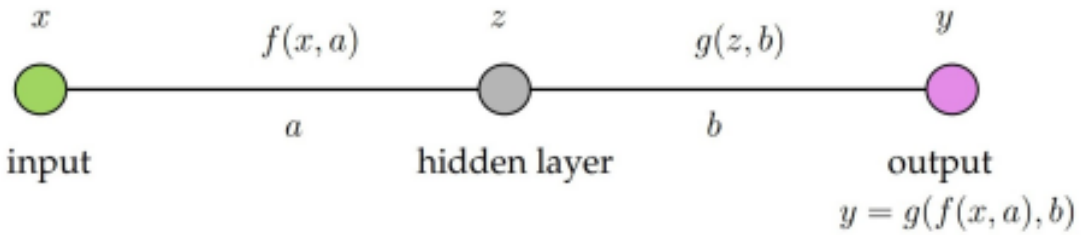


Image 4.1 – Illustration de l'algorithme de rétropropagation sur un réseau à une couche cachée et un seul nœud.

Une observation cruciale est que la nature compositionnelle du réseau, combinée à la règle de dérivation en chaîne, oblige l'optimisation à rétropropager l'erreur à travers le réseau. En particulier, les termes $(\partial y/\partial z)(\partial z/\partial a)$ illustrent comment cette rétropropagation se produit. Étant donné des fonctions $f(\cdot)$ et $g(\cdot)$, la règle de dérivation en chaîne peut être calculée explicitement.

La rétropropagation résulte en une règle de mise à jour itérative par descente

de gradient :

$$a_{k+1} = a_k - \delta \frac{\partial E}{\partial a_k}, \tag{4.4}$$

$$b_{k+1} = b_k - \delta \frac{\partial E}{\partial b_k}, \tag{4.5}$$

où δ représente le taux d'apprentissage et $\frac{\partial E}{\partial a}$ ainsi que $\frac{\partial E}{\partial b}$ peuvent être calculés explicitement en utilisant (3.4). L'algorithme itératif est exécuté jusqu'à convergence. Comme pour toute optimisation itérative, une bonne estimation initiale est cruciale pour obtenir une solution dans un temps de calcul raisonnable.

La rétropropagation se déroule comme suit :

1. Un réseau neuronal (NN) est spécifié avec un ensemble d'entraînement étiqueté.
2. Les poids initiaux du réseau sont définis à des valeurs aléatoires. Il est important de ne pas initialiser les poids à zéro, car cela pourrait entraîner des gradients identiques pour les neurones et limiter l'apprentissage.
3. Les données d'entraînement sont exécutées à travers le réseau pour produire une sortie y, dont la sortie idéale est y_0. Les dérivées par rapport à chaque poids sont ensuite calculées en utilisant les formules de rétropropagation (3.6).
4. Pour un taux d'apprentissage donné δ, les poids sont mis à jour comme dans (3.7).
5. On retourne à l'étape (iii) et on continue jusqu'à atteindre un nombre maximal d'itérations ou la convergence.

Comme exemple simple, considérons la fonction d'activation linéaire $f(\xi, \alpha) = g(\xi, \alpha) = \alpha\xi$. Dans ce cas, nous avons :

$$z = ax, \quad y = bz.$$

Nous pouvons maintenant calculer explicitement les gradients comme dans (3.6). Cela donne :

$$\frac{\partial E}{\partial a} = -(y_0 - y)\frac{\partial y}{\partial z}\frac{\partial z}{\partial a} = -(y_0 - y) \cdot b \cdot x, \tag{4.6}$$

$$\frac{\partial E}{\partial b} = -(y_0 - y)\frac{\partial y}{\partial b} = -(y_0 - y) \cdot z = -(y_0 - y) \cdot a \cdot x. \tag{4.7}$$

Avec les valeurs actuelles de a et b, ainsi que la paire entrée-sortie x et y, et la vérité cible y_0, chaque dérivée peut être évaluée pour effectuer la mise à jour (3.7).

Pour un réseau plus profond, considérons un réseau avec M couches cachées, étiquetées de z_1 à z_M. La généralisation est donnée par :

$$\frac{\partial E}{\partial a} = -(y_0 - y)\frac{\partial y}{\partial z_M}\frac{\partial z_M}{\partial z_{M-1}} \cdots \frac{\partial z_2}{\partial z_1}\frac{\partial z_1}{\partial a}.$$

La cascade de dérivées induite par la composition met en évidence la rétropropagation des erreurs. En désignant tous les poids à mettre à jour par le vecteur w, alors :

$$w_{k+1} = w_k - \delta \nabla E,$$

où ∇E est le gradient de l'erreur. Exprimé composante par composante :

$$w_j^{k+1} = w_j^k - \delta \frac{\partial E}{\partial w_j^k}.$$

La méthode la plus importante pour rendre ce calcul tractable est la descente de gradient stochastique, étudiée dans la section suivante.

4.3 Fonctions de perte et optimiseurs

L'optimisation dans le Machine Learning est l'une des étapes les plus importantes et peut-être aussi la plus difficile à apprendre. L'optimiseur est une fonction qui optimise les modèles d'apprentissage automatique à l'aide de données d'entraînement. Les optimiseurs utilisent une fonction de perte pour calculer la perte du modèle, puis, en fonction de celle-ci, essaient de l'optimiser. Donc, sans optimiseur, un modèle d'apprentissage automatique ne peut rien faire d'extraordinaire.

Dans ce Partie, mon objectif est d'expliquer le fonctionnement de l'optimisation, la logique qui la sous-tend et les mathématiques sous-jacentes.

Qu'est-ce que la fonction de perte ?

La fonction de perte (également appelée fonction d'erreur, fonction de coût ou fonction d'énergie) est une fonction qui calcule à quel point un modèle d'apprentissage automatique est bon ou mauvais. Si vous entraînez un modèle, vous pouvez utiliser une fonction de perte pour calculer le taux d'erreur. Si l'erreur est 0, votre modèle est parfait.

Dans les projets du monde réel, il est impossible d'obtenir une erreur de 0. L'objectif est donc d'obtenir un résultat aussi proche que possible de 0.

Comment la calculer ?

Il existe plusieurs façons de calculer la perte d'un modèle à l'aide de certaines fonctions de perte.

Erreur quadratique moyenne

Une façon populaire de calculer le taux d'erreur du modèle est appelée MSE (Mean Squared Error) ou erreur quadratique moyenne. Dans l'erreur quadratique moyenne, nous calculons la moyenne de la différence quadratique entre toutes les

valeurs prédites et les valeurs réelles pour toutes les entrées. La formule mathématique est donnée ci-dessous :

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

où y_i représente les valeurs réelles et $\hat{y}_i$ représente les valeurs prédites.

MSE produit essentiellement un nombre. La valeur minimale possible est 0, et la sortie est toujours égale ou supérieure à zéro.

L'optimisation commence

Une fois que nous avons la fonction de perte, nous disposons d'une mesure pour évaluer à quel point un modèle est bon ou mauvais. L'optimiseur peut maintenant réduire le taux d'erreur de la fonction de perte, et ainsi optimiser le modèle.

4.4 Surapprentissage et techniques de régularisation

4.4.1 Le surapprentissage

L'une des difficultés rencontrée lors de l'application de techniques d'apprentissage machine est le surapprentissage. Plus les techniques utilisées sont puissantes (grand nombre de paramètres libres), plus nous sommes susceptibles au surapprentissage.

Lors du surapprentissage, le modèle diverge du principe du rasoir d'Occam en augmentant si bien son niveau de complexité qu'il finit par essentiellement mémoriser chaque détails de l'ensemble d'entraînement. Un modèle surentrainé peut donc difficilement généraliser sur de nouveaux cas.

Heureusement, le surapprentissage peut être contrôlé à l'aide de différentes techniques de régularisation afin de générer des modèles plus parsimonieux.

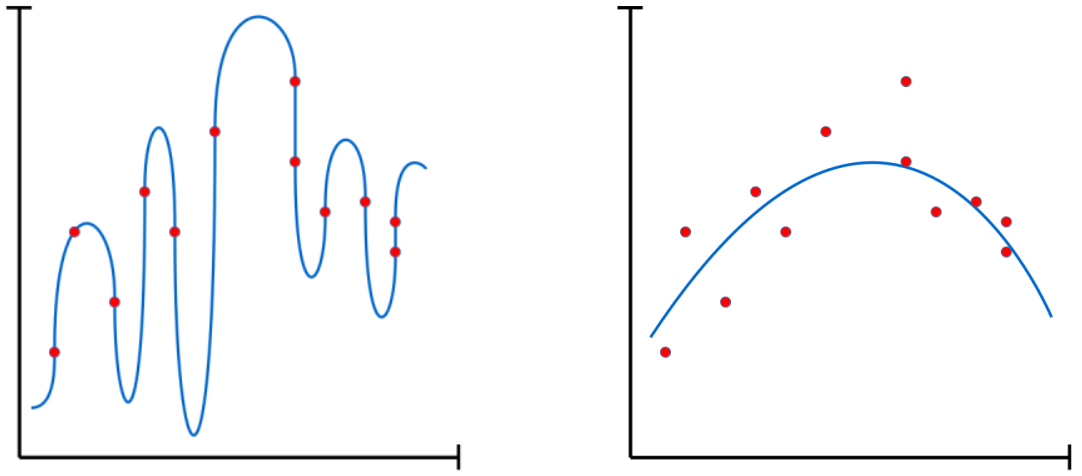


IMAGE 4.2 – Gauche : Over fiting - Droite : Proper fiting

4.4.2 La régularisation

La régularisation est une technique essentielle en apprentissage automatique, ou « machine learning ». Elle sert à éviter un problème très courant appelé "sur-apprentissage", sur-ajustement ou "overfitting" en anglais. Lorsqu'on entraîne un modèle sur des données, on souhaite que ce modèle puisse généraliser son apprentissage à de nouvelles données.

La régularisation est une technique qui empêche un modèle de "sur-ajuster" en lui ajoutant des informations supplémentaires. Il s'agit d'une forme de régression qui rétrécit les estimations des coefficients vers zéro . Cette technique apporte de légères modifications à l'algorithme d'apprentissage de sorte que le modèle se généralise mieux, améliorant les performances du modèle sur des données inédites ou "invisibles" .

Il existe plusieurs techniques de régularisation couramment utilisées pour contrôler la complexité des modèles d'apprentissage automatique.

La régularisation L1

La "régularisation L1", aussi appelée régularisation **Lasso**, est une technique d'optimisation qui vise à minimiser la complexité du modèle en ajoutant une pénalité égale à la somme absolue des coefficients du modèle (c'est-à-dire la norme L1 des coefficients) à la fonction de coût. Elle favorise des solutions parcimonieuses, c'est-à-dire avec beaucoup de coefficients nuls.

$$\text{Régularisation L1 : } \lambda \sum_{i=1}^{n} |\theta_i|$$

La régularisation L2

La "régularisation L2", ou régression crête ou Ridge, est une autre technique d'optimisation qui ajoute une pénalité égale à la somme des carrés des coefficients du modèle (c'est-à-dire la norme L2 des coefficients) à la fonction de coût. Elle a tendance à réduire les coefficients sans les rendre nuls. Ces deux techniques aident à prévenir le surapprentissage.

$$\text{Régularisation L2} : \lambda \sum_{i=1}^{n} \theta_i^2$$

Au final, la régression Lasso et la régression Ridge cherchent toutes deux à minimiser la somme des carrés des résidus, mais elles diffèrent dans la manière dont elles ajoutent une pénalité aux coefficients du modèle pour éviter le surapprentissage. En résumé, alors que la régression Ridge réduit la magnitude des coefficients, la régression Lasso peut les rendre nuls, conduisant à un modèle plus simple et plus interprétable.

Partie 5

Reconnaissance faciale avec les réseaux de neurones

5.1 Vue d'ensemble de la technologie de reconnaissance faciale

La reconnaissance faciale fascine et intrigue. Cette technologie révolutionnaire permet d'identifier une personne grâce à son visage. Elle repose sur un processus en trois étapes : détection, analyse, et reconnaissance. En utilisant des images numériques ou des vidéos, elle compare des traits faciaux avec des bases de données. Les applications abondent, de la sécurité aux réseaux sociaux, et elles transforment notre quotidien. Les avantages sont nombreux : rapidité, simplicité, et possibilités d'authentification. Toutefois, des enjeux éthiques et des inconvénients émergent, incitant à réfléchir à cette avancée technologique fascinante.

- **Détection :** La première étape de la reconnaissance faciale est la détection. Le système localise un visage dans une image ou une vidéo. Cette opération est facilitée grâce à des mécanismes sophistiqués qui scannent les pixels d'une image pour y repérer les caractéristiques du visage.
- **Analyse :** Une fois le visage détecté, l'étape suivante est l'analyse. Ici, le système examine divers traits faciaux, tels que la forme des yeux, la largeur du nez ou la distance entre les pommettes. Ces informations créent une sorte de carte faciale unique, propre à chaque individu.
- **Reconnaissance** Enfin, la dernière phase est la reconnaissance. Dans cette étape, la technologie compare la carte faciale créée à d'autres enregistrements existants dans une base de données. Si une correspondance est trouvée, l'identité de la personne est confirmée.

5.2 Application des CNN en reconnaissance faciale

Cette application permet la détection et la classification des émotions exprimées à travers des expressions faciales, appliquées sur des images et des séquences vidéo. Pour ce faire, nous utilisons

— des réseaux de neurones convolutifs (CNN).
— Langage utilisé : Python
— Jeu de données : fer2013
— Environnement de développement : Jupyter

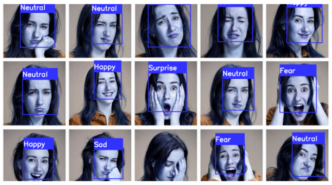
5.2.1 Résultats

Pour consulter le code Python, veuillez vous référer au fichier Jupyter joint.

— **Résultats pour les images :** Voici les résultats obtenus pour la détection et la classification des émotions à partir des images.

(a) Résultat Image 1

(b) Résultat Image 2



(c) Résultat Image 3

Image 5.1 – Résultats de la détection et classification des émotions à partir des images

— **Résultats pour les séquences vidéo :** Pour obtenir les résultats de la détection sur les séquences vidéo, il faut exécuter le programme "seq video detection".

Partie 6

Tendances futures et perspectives de recherche

Les réseaux de neurones continuent d'évoluer rapidement, et plusieurs tendances et perspectives de recherche prometteuses se dessinent pour l'avenir. Voici quelques-unes des plus importantes :

- **Apprentissage continu et adaptatif** : Les réseaux de neurones deviennent de plus en plus capables d'apprendre en continu à partir de nouvelles données, sans avoir besoin de re-entraînement complet. Cela permet une adaptation plus rapide aux changements et aux nouvelles informations.
- **Réseaux de neurones quantiques** : La recherche sur les réseaux de neurones quantiques explore comment les principes de la mécanique quantique peuvent être utilisés pour améliorer les performances des réseaux de neurones classiques. Cela pourrait potentiellement révolutionner le domaine de l'intelligence artificielle.
- **IA explicable et interprétable** : Il y a un effort croissant pour rendre les décisions des réseaux de neurones plus compréhensibles pour les humains. Cela inclut le développement de techniques pour visualiser et interpréter les processus internes des réseaux de neurones.
- **Fusion avec d'autres technologies émergentes** : Les réseaux de neurones sont de plus en plus intégrés avec d'autres technologies avancées, telles que l'intelligence artificielle conversationnelle, la réalité augmentée et la réalité virtuelle.
- **Réduction de l'empreinte carbone** : Avec la consommation énergétique élevée des réseaux de neurones profonds, la recherche se concentre sur le développement de modèles plus économes en énergie et sur l'utilisation de sources d'énergie renouvelables.
- **Éthique et régulation** : À mesure que les réseaux de neurones deviennent plus puissants, des questions éthiques et réglementaires se posent. La recherche sur les biais algorithmiques, la confidentialité des données et la transparence des modèles devient cruciale.

Ces tendances montrent que les réseaux de neurones continueront à repousser les limites de ce qui est possible en intelligence artificielle, tout en abordant les défis et les considérations éthiques associés.

Conclusion

En somme, les réseaux de neurones ont transformé notre manière d'aborder de nombreux défis technologiques et scientifiques. Grâce à des avancées significatives, tels que les réseaux de neurones profonds, convolutifs, récurrents et antagonistes génératifs, ainsi qu'à l'intégration croissante avec d'autres technologies émergentes, les possibilités semblent infinies. Les perspectives futures sont tout aussi prometteuses, avec des recherches en cours sur l'apprentissage continu, les réseaux de neurones quantiques, et l'explicabilité de l'IA. Toutefois, il est crucial de considérer les aspects éthiques et environnementaux pour garantir un développement responsable et durable. Les réseaux de neurones continueront à jouer un rôle central dans l'évolution de l'intelligence artificielle, offrant des solutions innovantes tout en repoussant les limites de ce qui est possible. En gardant à l'esprit les défis et les opportunités, nous pouvons espérer un avenir où l'IA contribuera positivement à notre société.

Références

[1] Bishop, C. M., *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] Goodfellow, I., Bengio, Y., Courville, A., *Deep Learning*. MIT Press, 2016.

[3] Hastie, T., Tibshirani, R., Friedman, J., *The Elements of Statistical Learning*. Springer, 2009.

[4] LeCun, Y., Bengio, Y., Hinton, G., *Deep learning*. Nature, 2015.

[5] Schmidhuber, J., *Deep Learning in Neural Networks : An Overview*. Neural Networks, 2015.

[6] Steven L. Brunton, Department of Mechanical Engineering, University of Washington.

[7] J. Nathan Kutz, Department of Applied Mathematics, University of Washington