الجمهورية الجزائرية الديمقراطية الشعبية
Republique Algerienne Democratique Et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
جامعة العربي التبسي – تبسة
Université Larbi Tébessi – Tébessa –
Faculté des Sciences et de la Technologie
Département de génie électrique

MEMOIRE

Présenté pour l'obtention du **diplôme** de **Master Académique**

En : Automatique

Spécialité : automatique et système

Par : **Maalem Nour el houda**
Zaidi Rayene

Sujet

Réseaux Contradictoires Génératifs : Concepts et applications du GAN

Présenté et soutenu publiquement, le / /, devant le jury composé de :

M. DJABRI RIADH	MAITRE DE CONFERENCE	PRESIDENT
M. CHERIET LAYLA	MAITRE DECONFERENCE	EXAMINATEUR
M. KHEMAISSIA SEDDIK	PHD	RAPPORTEUR
…	…	…

Promotion : 2021/2022

Table des matières :

CHAPITRE III : CONCEPTION DES GAN

Liste de figure :

Liste des tableaux :

Liste d'abréviation :

ANN : Artificiel Neural Networks

API : Application Program Ming Interface

CNN : Les réseaux de neurones convolution els

DL : Deep Learning

ENISA : l'Agence de la cybersécurité de l'Union européenne

FC: fully connected layer

FGSM: The Fast Gradient Sign Method

GPU : Processeur graphique

IA : Intelligence Artificielle

ML: machine learning

MLP: Multi-Layer Perceptron

NTIC : Technologies de l'information et de la communication

PMC : Le perceptron Multi Couches

Py: Python

RAM: Random Access Memories

Relu : Couche unité rectifié linéaire

ReLU : Rectified Linear Activation

RNN : Réseau de neurones récurrents

RVB : Couleur rouge, vert et bleu

USD: United State Dollar

W: abréviation de poids en anglais (Wight)

Introduction Générale

Introduction Générale :

La notion de savoir si les machines peuvent penser est plus ancienne que l'ordinateur lui-même. En 1950, le célèbre mathématicien, logicien et informaticien Alan Turing—
Peut-être mieux connu pour son rôle dans le décodage de la machine de chiffrement nazie en temps de guerre, Enigma - a rédigé un article qui immortaliserait son nom pendant des générations à venir, « Machines informatiques et intelligence ».

Dans l'article, Turing proposait un test qu'il appelait le jeu d'imitation, plus connu aujourd'hui comme le test de Turing. Dans ce scénario hypothétique, un observateur inconscient parle avec deux homologues derrière une porte close : l'un, un être humain ; l'autre, un ordinateur. Turing explique que si l'observateur est incapable de dire quelle est la personne et qui est la machine, l'ordinateur a réussi le test et doit être considéré comme intelligent. Les algorithmes d'apprentissage automatique sont excellents pour reconnaître les modèles dans les données existantes et utiliser ces informations pour des tâches telles que la classification (attribuer la bonne catégorie à un exemple) et la régression (estimation d'une valeur numérique basée sur une variété d'entrées). Lorsqu'on leur a demandé de générer de nouvelles données, cependant, les ordinateurs ont eu du mal. Un algorithme peut vaincre un grand maître d'échecs, estimer les mouvements du cours des actions et classer si une transaction par carte de crédit est susceptible d'être frauduleuse. En revanche, toute tentative de faire une petite conversation avec Alexa d'Amazon ou Siri d'Apple est vouée à l'échec.

Tout a changé en 2014 lorsque Ian Goodfellow, alors étudiant au doctorat à l'Université de Montréal, a inventé les réseaux antagonistes génératifs (GAN). Cette technique a permis aux ordinateurs de générer des données réalistes en utilisant non pas un, mais deux réseaux de neurones. Les GAN n'étaient pas le premier programme informatique utilisé pour générer des données, mais leurs résultats et leur polyvalence les distinguent de tous les autres. Les GAN ont atteint des résultats remarquables qui avaient longtemps été considérés comme pratiquement impossibles pour les systèmes artificiels, tels que la capacité de générer de fausses images avec une qualité semblable à celle du monde réel, se transforment en un griffonner en une image semblable à une photographie ou transformer une séquence vidéo d'un cheval en une course zebra, le tout sans avoir besoin de vastes trésors de données d'entraînement minutieusement étiquetées.

CHAPITRE I

L'APPRENTISSAGE PROFOND ET RÉSEAUX DE NEURONES ARTIFICIELS

I.1 Introduction :

Le cerveau humain est composé de 86 milliards de neurones, des cellules nerveuses connectées les unes aux autres. Les réseaux de neurones artificiels sont, eux aussi, composés de nombreuses unités, des fonctions mathématiques, assimilables à des neurones très simplifiés. Dans le cerveau, l'apprentissage modifie les connexions entre les neurones ; il en va de même dans les réseaux de neurones artificiels. Comme ces unités sont souvent organisées en couches multiples, on parle donc de réseaux et d'apprentissage « profond ». Le rôle de ces neurones artificiels est de calculer une somme pondérée de leurs signaux d'entrée, et de produire un signal de sortie si cette somme dépasse un certain seuil. Mais un neurone artificiel n'est ni plus ni moins qu'une fonction mathématique calculée par un programme d'ordinateur. Et si le champ lexical de l'intelligence artificielle est proche de celui du cerveau, ce n'est pas un hasard : les découvertes en neurosciences ont nourri la recherche en IA.

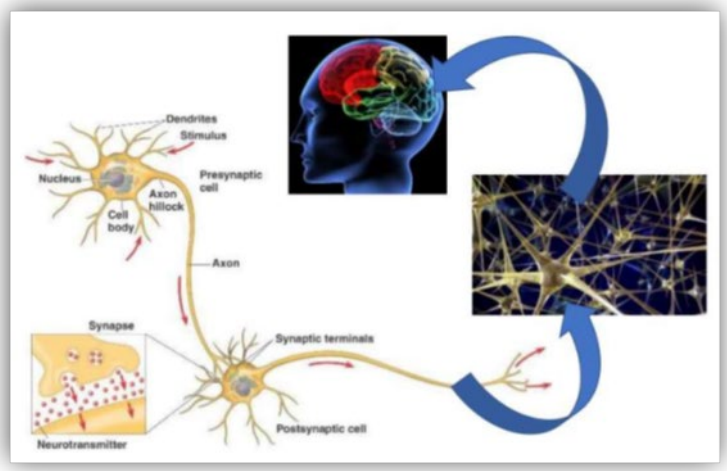


Figure (I.1) : Réseau de neurones et fonction du cerveau

Dans l'apprentissage profond, qui se compose de plusieurs couches de neurones artificiels connectés les uns aux autres, le même problème de blâme existe. En 1986, Hinton et ses collègues David Rumelhart et Ronald Williams ont constaté que lorsque l'information se déplace à travers différentes couches neuronales, en observant dans quelle mesure la sortie manque sa marque par rapport à celle désirée, il est possible de calculer mathématiquement un signal d'erreur.

Ce signal peut ensuite être renvoyé à travers les couches du réseau neuronal, chaque couche recevant individuellement un nouveau signal d'erreur sur la base de ses couches supérieures. D'où le nom de « rétropropagation »

C'est un peu comme cinq personnes se passant un ballon de basket en ligne, et le dernier lancer est raté. L'entraîneur - dans ce cas, la rétropropagation - partira du joueur final, jugera de la probabilité de son problème et redescendra la ligne pour déterminer qui a besoin d'ajustement. Dans un réseau de neurones artificiels, « ajustement » signifie changer le poids synaptique.

L'étape suivante consiste pour le réseau à calculer à nouveau la même sortie. Cette fois-ci, le ballon rentre. Cela signifie que les ajustements effectués par l'entraîneur back propagation ont bien fonctionné. Le réseau adoptera les nouveaux poids synaptiques et le cycle d'apprentissage se poursuit.

Cela ressemble à une façon logique d'apprentissage. Back propagation, en combinaison avec d'autres algorithmes, a fait de l'apprentissage en profondeur la technique dominante dans la reconnaissance faciale, la traduction de la langue et les victoires de l'IA contre les humains au Go et au poker.

La réalité est que dans les réseaux de neurones profonds, l'apprentissage en suivant le gradient d'une mesure de performance fonctionne très bien, ont déclaré les auteurs. Notre seul souci est de montrer qu'il y a une chance que les idées derrière le back propagation fonctionnent également pour le cerveau ?

I.2 L'intelligence artificiel, l'apprentissage automatique et l'apprentissage profond :

Le développement dans le domaine de la technologie s'est amélioré au fil des années. Avec le temps, nous obtenons des termes comme l'intelligence artificielle, l'apprentissage automatique et l'apprentissage profond en technologie. Nous confondons souvent ces termes et les définissons de manière similaire. Mais ce n'est pas une définition précise car ces termes sont différents les uns des autres. Dans ce chapitre, nous allons essayer de voir la différence entre ces trois termes AI, ML et Deep Learning.

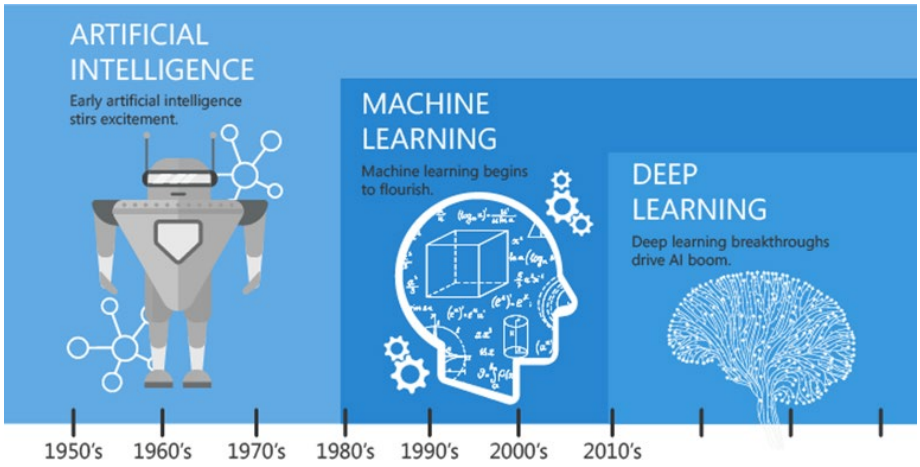


Figure (I.2) : différence entre AI, ML et Deep Learning

I.2.1 Intelligence artificielle :

L'intelligence artificielle, comme son nom l'indique c'est l'intelligence créée par les humains. En effet, ce sont des machines complexes utilisant des propriétés informatiques et effectuant diverses actions comme nous, les humains. En un mot, il intègre l'intelligence humaine dans des machines.

I.2.2 Apprentissage automatique :

L'apprentissage automatique fait partie de l'intelligence artificielle. La plupart des gens le considèrent comme une intelligence artificielle, mais ce n'est pas le cas. Les machines peuvent apprendre. Les robots apprennent d'eux-mêmes à partir des données qui leur sont fournies. Cela ressemble plus à une technique qui nous fait prendre conscience de la présence de l'Intelligence Artificielle. Cette technique utilise des algorithmes pour obtenir des données, apprendre, puis analyser les données. Les résultats sont obtenus sous forme de prédictions. Vous l'avez peut-être remarqué lorsque vous avez reçu des recommandations sur des sites commerciaux, Google ou Facebook. Vous obtenez des suggestions en fonction de vos intérêts. Cela se fait avec des algorithmes d'apprentissage automatique qui sont développés de manière à analyser les recherches récentes, l'historique et d'autres informations. Cette technique influence également les secteurs marketing et bancaire.

« L'apprentissage automatique est la tendance des machines à tirer des enseignements de l'analyse des données et à atteindre l'intelligence artificielle. »

Les nouveaux algorithmes d'apprentissage automatique étaient limités à l'IA de base, mais maintenant ils sont devenus une partie essentielle de ce système. De nombreux algorithmes complexes sont préparés pour offrir une meilleure expérience. Cela a révolutionné la façon de regarder des émissions et des films. L'industrie du divertissement utilise cet algorithme pour fournir des suggestions appropriées à ses téléspectateurs sur des chaînes Web comme Netflix et Amazon Prime. L'apprentissage automatique est le concept d'analyse des données et offre d'excellentes recommandations basées sur l'apprentissage à partir de ces points.[1]

I.2.3 L'apprentissage en profondeur :

C'est le sous-ensemble de l'apprentissage automatique, ou on peut dire l'intelligence artificielle, qui est la raison derrière les capacités de travail des machines. Cette technique est similaire à l'apprentissage automatique dans certains contextes. La différence entre ces deux éléments est que l'apprentissage automatique a besoin de conseils pour effectuer une tâche, tandis que l'apprentissage en profondeur du modèle le fera lui-même sans l'interférence du programmeur. Le Deep Learning a amélioré l'expertise des utilisateurs. Le meilleur exemple d'apprentissage en profondeur est une voiture automatique.

« La technique utilisée pour implémenter le machine learning est connue sous le nom de Deep learning. »

L'apprentissage en profondeur a permis aux machines de fonctionner et de penser comme des humains. Dans l'apprentissage automatique, les programmeurs doivent corriger l'algorithme si les résultats sont inappropriés. Mais les modèles d'apprentissage profond font ceux-là tout comme le cerveau humain.

I.3 Machine learning et le deep learning:

Le Deep learning et le machine learning sont deux concepts liés à l'intelligence artificielle. Les deux se sont combinés pour améliorer l'avenir de l'IA, mais ce n'est pas de l'intelligence artificielle. Ils sont différents à bien des égards qui doivent tenir compte pour créer l'AI qui est encore mieux que le cerveau humain. Lorsque les machines font preuve d'intelligence humaine, cela s'appelle l'intelligence artificielle. L'apprentissage automatique des perspectives pour obtenir l'intelligence artificielle et sa mise en œuvre est appelé apprentissage en profondeur. C'est la différence que chacun doit connaître au moment d'utiliser ces termes. Maintenant, c'estla différence fondamentale que ces trois termes ont entre eux.

Une des grandes différences entre le Deep Learning et les algorithmes de ML traditionnelles c'est qu'il s'adapte bien, plus la quantité de données fournie est grande plus les performances d'un algorithme de Deep Learning sont meilleures. Contrairement à plusieurs algorithmes de ML classiques qui est limite par la quantité de données qu'ils peuvent recevoir des fois appelée "plateau de performance", les modèles de Deep Learning n'ont pas de telles limitations (théoriquement) et ils sont même allés jusqu'à dépasser la performance humaine dans des domaines comme l'image processing.

Autre différence entre les algorithmes de ML traditionnelles et les algorithmes de Deep Learning c'est l'étape de l'extraction de caractéristiques. Dans les algorithmes de ML traditionnelles l'extraction de caractéristiques est faite manuellement, c'est une étape difficile et coûteuse en temps et requiert un spécialiste en la matière alors qu'en Deep Learning cette étape est exécutée automatiquement par l'algorithme.[2]

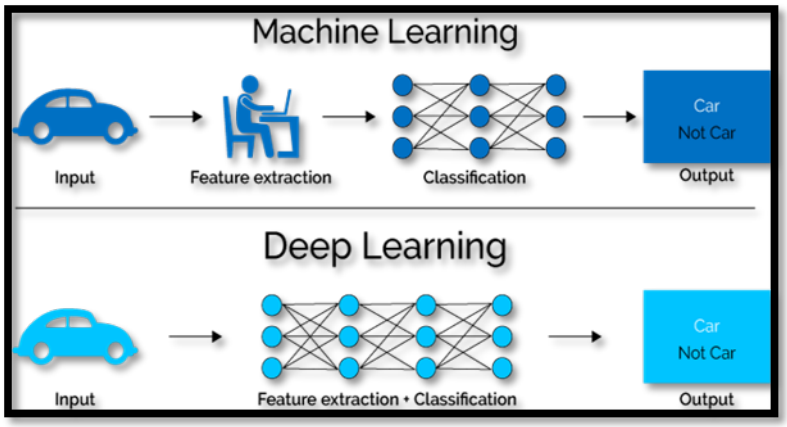


Figure (I.3) : différence entre L'apprentissage profond et L'apprentissage automatique

I.4 L'apprentissage profond (Deep learning) :

Au début des années 2000, la puissance de calcul s'est accrue de façon exponentielle et l'industrie a contribué à une explosion des techniques de calcul de ce fait, l'apprentissage en profondeur a émergé grâce à la croissance de l'informatique explosive durant cette décennie en tant que concurrent sérieux dans ce domaine, remportant de nombreux concours d'apprentissage automatique importants, et à partir de 2017, L'apprentissage profond devient l'un des domaines les plus en vogue de la science des données. De nombreuses études de cas ont donné des résultats étonnants dans les domaines de la robotique, de la reconnaissance des formes et de l'intelligence artificielle (IA).

Plusieurs publications ont énormément influencé le développement dans ce domaine :

- Apprentissage basé sur le gradient appliqué par la NYU à la reconnaissance de documents (1998), qui introduit le réseau de neurones convolution els dans le monde de l'apprentissage automatique.[3]
- Deep Boltzmann Machines de Toronto (2009), qui présente un nouvel algorithme d'apprentissage pour les machines Boltzmann, qui contient de nombreuses couches cachées.[4]
- Stanford & Google a créé des fonctionnalités de haut niveau à l'aide de l'apprentissage non supervisé à grande échelle (2012), qui résout le problème de la création de détecteurs de fonctionnalités de haut niveau et spécifiques à une classe à partir de données non étiquetées uniquement.[5]
- De CAF - Une fonctionnalité d'activation par convolution profonde de Berkeley pour la reconnaissance visuelle générique (2013), qui publie De CAF, une implémentation à code source ouvert des fonctionnalités d'activation par convolution profonde, ainsi que tous les paramètres de réseau associés permettant aux chercheurs en vision de mener des expériences avec représentations à travers une gamme de paradigmes d'apprentissage de concepts visuels.[6]
- Deep Ateliers de DeepMind avec Deep Rein forcement Learning (2016), qui présente le premier modèle d'apprentissage en profondeur permettant de maîtriser les politiques de contrôle directement à partir de données sensorielles de haute dimension utilisant l'apprentissage par renforcement.[7]

I.4.1 Le neurone biologique :

Le neurone biologique est une cellule nerveuse qui constitue l'unité fonctionnelle fondamentale du système nerveux de tous les animaux. Les neurones existent pour communiquer les uns avec les autres et transmettre des impulsions électrochimiques à travers les synapses, d'une cellule à l'autre, à condition que l'impulsion soit suffisamment puissante pour activer la libération de produits chimiques à travers une fente synaptique. La force de l'impulsion doit dépasser un seuil minimal, sinon les produits chimiques ne seront pas libérés.

La figure suivante présente les principales parties de la cellule nerveuse : le soma, les dendrites, les axones et les synapses.

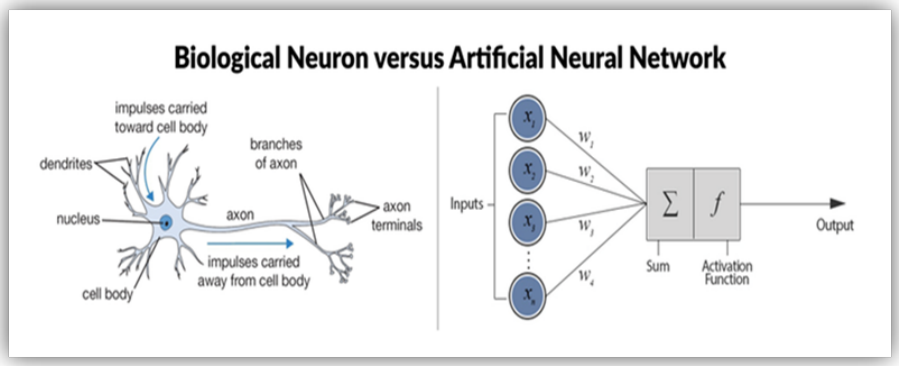


Figure (I.4) : neurone artificiel et biologique

Le neurone est constitué d'une cellule nerveuse constituée d'un soma (corps cellulaire) et de plusieurs dendrites mais un seul axone. L'axone unique peut cependant se ramifier des centaines de fois. Les dendrites sont des structures minces issues du corps cellulaire principal. Les axones sont des fibres nerveuses ayant une extension cellulaire particulière qui provient du corps de la cellule.

I.4.2 Flux d'informations à travers le neurone biologique :

Les synapses qui augmentent le signal sont considérées comme excitatrices, et celles qui le diminuent sont considérées comme inhibitrices. La plasticité fait référence aux changements à long terme de la force des connexions en réponse au stimulus d'entrée. Les neurones se sont également avérés créer de nouvelles connexions au fil du temps et même migrer. Ces mécanismes combinés au changement de connexion dirigent le processus d'apprentissage dans le cerveau biologique.

I.4.3 Le neurone artificiel :

Il a été démontré que le cerveau de l'animal est responsable des composants fondamentaux de l'esprit. Nous pouvons étudier les composants de base du cerveau et les comprendre.

La recherche a montré des moyens de cartographier les fonctionnalités du cerveau et de suivre les signaux lorsqu'ils se déplacent dans les neurones.

Les neurones artificiels peuvent être définis par le type d'entrée qu'ils sont capables de recevoir (binaire ou continue) et par le type de transformation (fonction d'activation) qu'ils utilisent pour générer une sortie.

Différents éléments caractéristiques importantes de ces neurones sont :

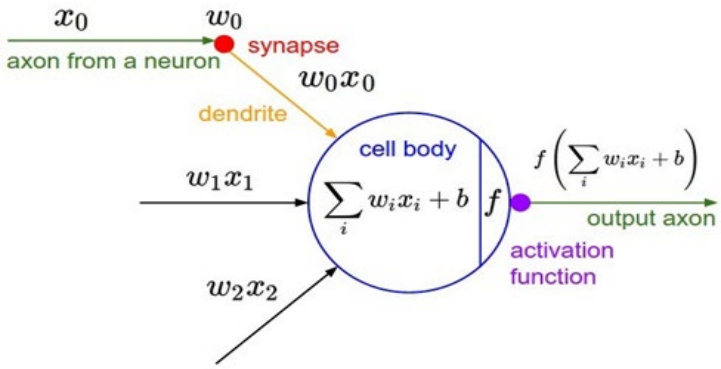


Figure (I.5) : Un neurone artificiel

I.4.3.1 Poids de connexion :

Les poids sur les connexions dans un réseau de neurones sont des coefficients qui modifient (amplifient ou minimisent) le signal entrant dans un neurone donné du réseau.

I.4.3.2 Les biais :

Les biais sont des valeurs scalaires ajoutées à l'entrée pour garantir qu'au moins quelques nœuds par couche sont activés, quelle que soit la puissance du signal. Les biais permettent l'apprentissage en donnant au réseau une action en cas de faible signal. Ils permettent au réseau d'essayer de nouvelles interprétations ou comportements.[8]

I.5 Présentation des réseaux de neurones artificiels :

Un réseau de neurones artificiels est une tentative de simulation du réseau de neurones

Constituant le cerveau humain, de sorte que l'ordinateur puisse apprendre des choses et prendre des décisions de manière humaine.

Les réseaux de neurones artificiels sont créés en programmant des ordinateurs ordinaires pour qu'ils se comportent comme des cellules cérébrales interconnectées.

Dans un réseau de neurones artificiels, nous avons des neurones artificiels disposés en groupes appelés couches comme suit :

- Une couche d'entrée
- Une ou plusieurs couches cachées, entièrement connectées
- Une seule couche de sortie

Comme le montre la figure 6, les neurones de chaque couche (représentés par les cercles) sont entièrement connectés à tous les neurones de toutes les couches adjacentes. Pour la couche d'entrée, l'entrée est l'entrée vectorielle brute. L'entrée des neurones des autres couches est la sortie (activation) des neurones de la couche précédente.

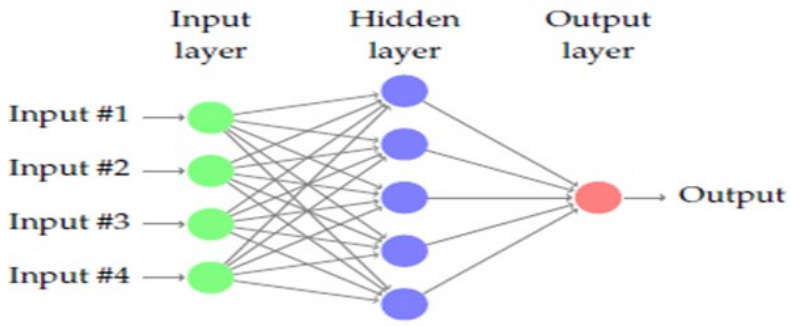


Figure (I.6) : Réseau de neurones artificiels

I.5.1 Couche d'entrée :

La couche d'entrée rassemble les données d'entrée dans le réseau neuronal. Les données d'entrée doivent être numériques. Le processus de manipulation des données avant leur entrée dans le réseau neuronal est appelé traitement des données.

I.5.2 Couche cachée :

Les couches cachées sont composées de la plupart des neurones du réseau neuronal et constituent le cœur de la manipulation des données pour obtenir la sortie souhaitée. Les données traverseront les couches cachées et seront manipulées par de nombreux poids et biais.

I.5.3 Couche de sortie :

La couche de sortie est le produit final de la manipulation des données dans le réseau neuronal. Souvent, la couche de sortie est constituée de neurones qui représentent chacun un objet et la valeur numérique attachée est la probabilité de cet objet spécifique.

L'idée principale est que les couches de sortie sont le résultat des données lors de leur passage à travers le réseau neuronal, et le but à atteindre.

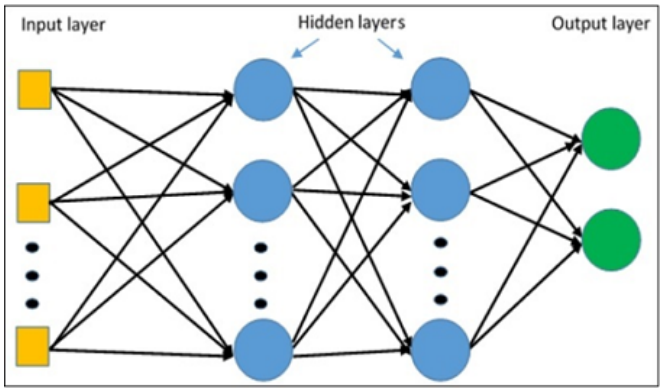


Figure (I.7) : La couche de sortie

I.5.4 Connexions inter-couches :

Dans un réseau entièrement connecté, les connexions entre les couches sont les connexions sortantes de tous les neurones de la couche précédente à tous les neurones de la couche suivante.

Les poids seront progressivement modifiés au fur et à mesure que notre algorithme trouve la meilleure solution possible durant la phase d'apprentissage de la rétropropagation.

Les poids peuvent être considérés comme une optimisation du vecteur de paramètres afin de minimiser les erreurs.

I.6 Fonctions d'activation :

La fonction d'activation est une composante essentielle durée seau neuronal. Cette fonction décide si le neurone sera activé ou non. C'est une transformation non linéaire de la valeur d'entrée.

La non-linéarité est si importante dans les réseaux de neurones. Il existe de nombreux types de ces fonctions, parmi lesquelles nous trouvons.[9]

Voici les principales fonctions d'activations que l'on peut trouver dans le domaine des réseaux de neurones :

I.6.1 La fonction Sigmoïde :

Cette fonction est l'une des plus couramment utilisées. Elle est bornée entre 0 et 1, et il peut être interprété stochastiquement comme la probabilité que le neurone s'active, et il est généralement appelé la fonction logistique ou le sigmoïde logistique. Fonction la plus

populaire depuis des décennies. Mais aujourd'hui, elle devient beaucoup moins efficace par rapport à d'autre pour une utilisation pour les couches cachées. Elle perd de l'information due à une saturation que cela soit pour la phase de feed forward ou de back propagation, en donnant des effets non linéaires au réseau due à un paramètre unique. Elle a aussi des soucis de gradient 0 avec des entrées étant très large, même si le souci est minimalisé avec le système utilisant des batch par lots (mini batch). Utilisé en couche de sortie pour de la classification binaire.

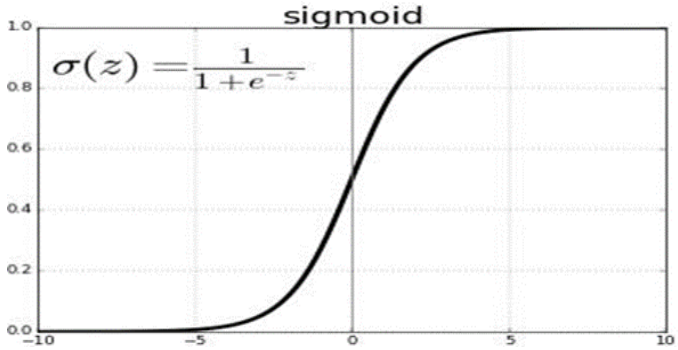


Figure (I.8) : Représentation graphique de la fonction Sigmoïde.

I.6.2 La fonction Tanh :

Utilisé pour des LSTM pour des données en continue. Intervalle de sortie : [-1,1]

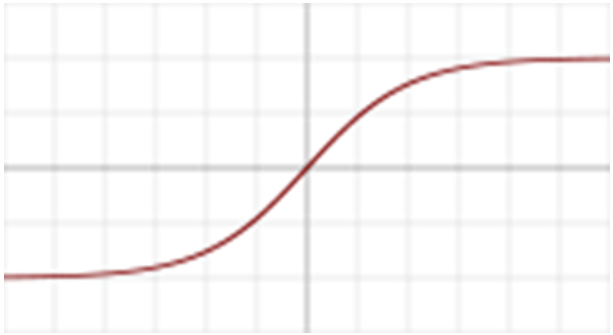


Figure (I.9) : La fonction Tanh

I.6.3 La fonction Relu (Rectifie Linéaire Unit) :

La fonction RELU est probablement la plus proche de sa correspondante biologique. Ce sont les fonctions les plus populaires de nos jours. Elles permettent un entrainement plus rapide comparé aux fonctions sigmoïde et tanh, étant plus légères. Attention au phénomène de 'Dying ReLU', auquel on préférera les variations de ReLU. Plus d'informations en fin d'article. Très utilisé pour les CNN, RBM, et les réseaux de multi perceptron. Intervalle de sortie [0 ; +∞ [.

Cette fonction est récemment devenue le choix de nombreuses tâches (notamment en computer vision).

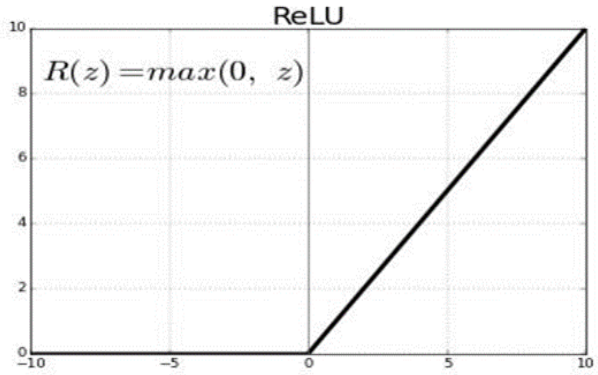


Figure (I.10) : Représentation graphique de la fonction Relu

I.6.4 La fonction Leaky ReLU :

La Leakey Relu permet d'ajouter une variante pour les nombres négatifs, ainsi les neurones ne meurent jamais. Ils entrent dans un long coma mais on toujours la chance de se réveiller à un moment donné. Intervalle de sortie] -∞ ; +∞ [.

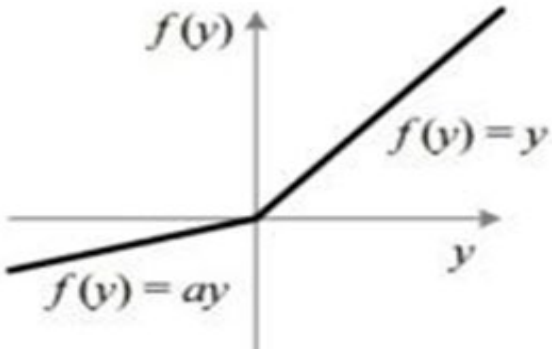


Figure (I.11) : La fonction Leaky ReLU

I.6.5 La fonction TReLU (Thresholded ReLU) :

Elle est identique à la simple ReLU. Mais la localisation de son seuil d'activation va être décalé, il n'est plus à 0, mais selon un paramètre thêta.

I.6.6 La fonction RRELU (Randomise Leakey Relu) :

La Randomise Leakey Relu permet de choisir le hyper paramètre ALPHA. Durant l'entrainement alpha est choisi aléatoirement. Puis durant les tests, il est calculé via une moyenne. Intervalle de sortie]-∞ ; +∞[.

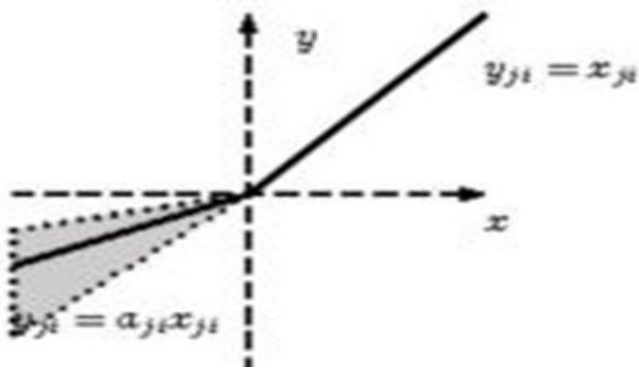


Figure (I.12) : LA fonction Randomise Leakey Relu

I.6.7 La fonction ELU (Exponentiel Linéaire Unit) :

Autre dérivé de la Relu. Celle-ci va approcher les valeurs moyenne proche de 0, ce qui va avoir comme impact d'améliorer les performances d'entrainements. Elle utilise exponentiel pour la partie négative et non plus une fonction linéaire. Elle parait plus performante en expérimentation que les autres Relu. Pas de soucis de neurone mort (dying Relu). Intervalle de sortie] -∞ ; +∞ [.

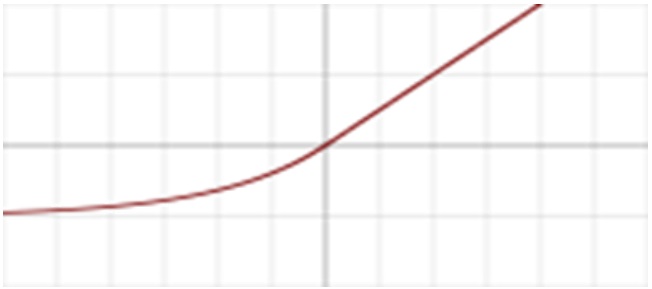


Figure (I.13) : Représentation graphique de La fonction Exponentiel Linéaire Unit

I.6.8 La fonction SeLU (Scaled ELU) :

C'est comme ELU en redimensionné mais avec en plus un paramètre ALPHA pré définit. Bon résultat, bonne vitesse, et évite les problèmes d'explosion et disparition de gradients en s'auto normalisant et gardant les mêmes variances pour les sorties de chaque couche, et ce tout au long de l'entrainement.

I.6.9 La fonction Soft max :

Régression Soft max (synonymes : Logistique multinomiale, Classificateur d'entropie maximale, ou simplement Régression logistique multi-classe) est une généralisation de la régression logistique que nous pouvons utiliser pour la classification multi-classes.[10]

Utilisé pour de la multi classification en couche de sortie. Intervalle de sortie] -∞ ; +∞ [.

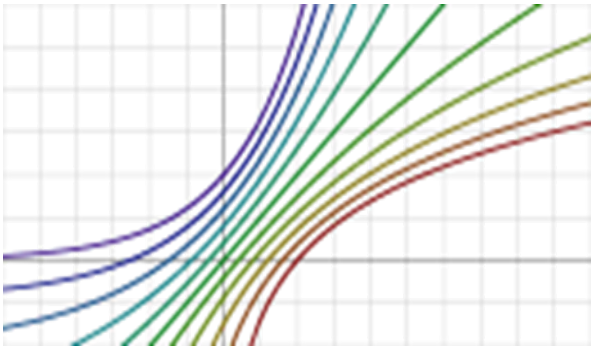


Figure (I.14) : Représentation graphique DE Régression Soft max

Contrairement à d'autres types de fonctions, la sortie d'un neurone d'une couche utilisant la fonction soft max dépend des sorties de tous les autres neurones des autres couches.

Cela s'explique par le fait qu'il nécessite que la somme de toutes les sorties soit égale à1.

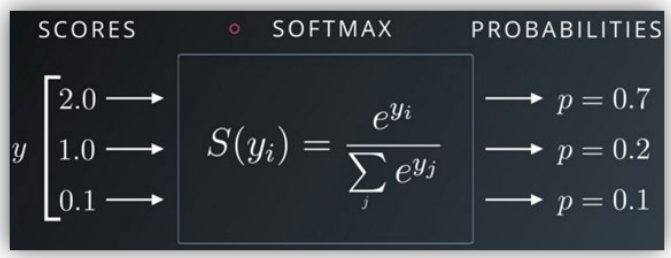


Figure (I.15) : fonction soft max

I.7 Les architectures des réseaux de neurones :

La plupart des architectures profondes sont réalisées en combinant et recombinant un ensemble limité de primitives architecturales (des couches de réseaux neuronaux).[11]

Selon Hinton, dans, nous disons que notre réseau de neurones est profond lorsque le nombre des couches cachées est supérieur à 1. Dans cette section, nous présenterons un bref aperçu des structures communes que l'on retrouve dans de nombreux réseaux profonds.[12]

I.7.1 Les réseaux entièrement connectés :

Un réseau entièrement connecté (Fully Connected en Anglais) permet de transformer une liste d'entrées en une liste de sorties. Couche d'un réseau neuronal artificiel dans laquelle tous les nœuds contenus se connectent à tous les nœuds de la couche suivante. Les couches entièrement connectées sont couramment utilisées dans les réseaux de neurones convolutifs et les réseaux de neurones récurrents.

La transformation est appelée totalement connectée car toute valeur d'entrée peut affecter toute valeur de sortie. Dans ce type de réseaux, les activités des neurones de chaque couche sont une fonction non-linéaire des activités de la couche inférieure.

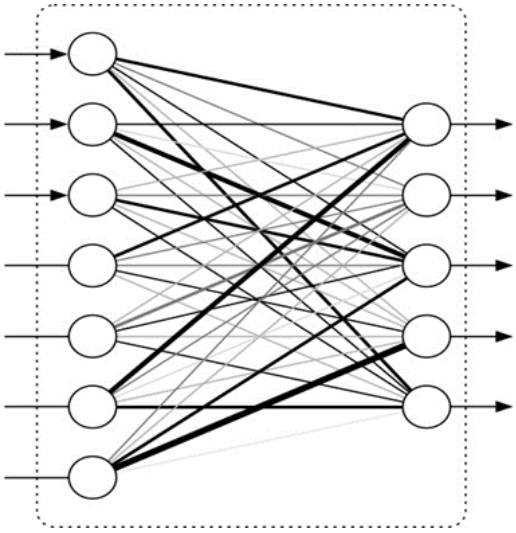


Figure (I.16) : LES réseau entièrement connecté

I.7.2 Les réseaux convolution els :

Un réseau convolution els (CNN) suppose une structure spatiale particulière dans son entrée. En particulier, il suppose que les entrées qui sont proches les unes des autres dans l'entrée originale sont sémantiquement liées. CNN est une séquence de couches, et chaque couche transforme un volume d'activations en un autre par une fonction différentiable. Les trois H principaux types de couches pour construire ce type de réseau sont : couche convolutive, couche de pooling et couche entièrement connectée.

I.7.3 Les réseaux neuronaux récurrents et LSTM :

Les couches de réseaux neuronaux récurrentes (RNN : Récurrent Neural Networks) sont des entités primitives qui permettent aux réseaux neuronaux d'apprendre à partir de séquences d'entrées.

La figure représente les types de séquences d'entrée que le réseau traite :[13]

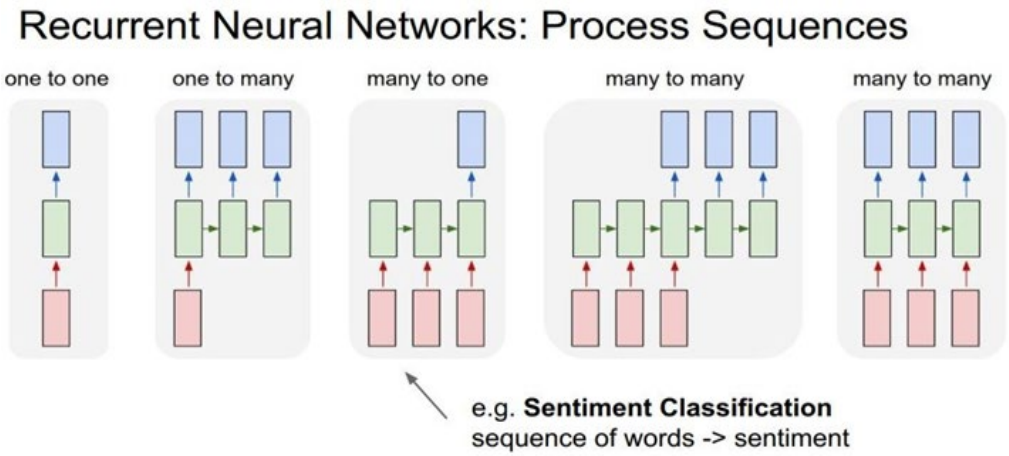


Figure (I.17) : Les types de séquences d'entrée pour un réseau récurrent

I.8 Apprentissage :

Un réseau neuronal artificiel bien entraîné a des poids qui amplifient le signal et atténuent le bruit. Un poids plus important signifie une corrélation plus étroite entre un signal et le résultat du réseau. Les entrées associées à des poids élevés affecteront davantage l'interprétation des données par le réseau que les entrées associées à des poids plus faibles.

Le processus d'apprentissage pour tout algorithme d'apprentissage utilisant des poids est le processus de réajustement des poids et des biais, en rendant certains plus petits et d'autres plus grands, attribuant ainsi une signification à certaines informations et en minimisant d'autres.

Cela aide notre modèle à déterminer quelles variables de prédiction (ou caractéristiques) sont liés à quels résultats et ajuste les poids et les biais en conséquence. Dans la plupart des ensembles de données, certaines caractéristiques sont fortement corrélées avec certaines étiquettes (par exemple, la superficie en pieds carrés correspond au prix de vente d'une maison). Les réseaux de neurones apprennent aveuglément ces relations en faisant une supposition basée sur les entrées et les poids, puis en mesurant la précision des résultats. Les

fonctions de perte dans les algorithmes d'optimisation, tels que la descente de gradient stochastique (Stochastique gradient descentes (SGD)), récompensent le réseau pour les bonnes suppositions et le pénalisent pour les mauvaises. SGD déplace les paramètres du réseau vers de bonnes prévisions et s'éloigne des mauvaises.

I .9 Conclusion :

L'apprentissage profond est un domaine de recherche en apprentissage automatique basé sur

Un type spécifique de mécanisme d'apprentissage. Il se caractérise par des efforts pour créer un modèle d'apprentissage à plusieurs niveaux, où les niveaux plus profonds prennent en compte les résultats des niveaux précédents, les transforment et les rendent plus abstraits que jamais. Cet aperçu des niveaux d'apprentissage s'inspire de la façon dont le cerveau traite l'information et apprend en interagissant avec des stimuli externes. Chaque niveau d'apprentissage correspond, par hypothèse, à l'une des différentes aires qui composent le cortex cérébral.

L'apprentissage en profondeur est basé sur l'idée de réseaux de neurones artificiels et est conçu pour gérer de grandes quantités de données en ajoutant des couches au réseau.

Le modèle d'apprentissage en profondeur a la capacité d'extraire des caractéristiques à partir de données brutes à travers plusieurs couches de traitement constituées de plusieurs transformations linéaires et non linéaires et d'apprendre ces caractéristiques étape par étape à travers chaque couche avec une intervention humaine minimale.

L'apprentissage en profondeur est la méthode Dominante dans la reconnaissance faciale, la traduction linguistique et les triomphes de l'intelligence artificielle.

CHAPITRE II

LES RÉSEAUX DE NEURONES CONVOLUTIFS (CNN)

II.1 Introductions :

Les réseaux de neurones convolutifs sont à ce jour les modèles les plus performants pour classer des images. Désignés par l'acronyme CNN, de l'anglais Convolution al Neural Network, ils comportent deux parties bien distinctes. En entrée, une image est fournie sous la forme d'une matrice de pixels. Elle a deux dimensions pour une image aux 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 d'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. En fin, les cartes de convolutions sont mises à plat et concaténées en un vecteur de caractéristiques, appelé code CNN.

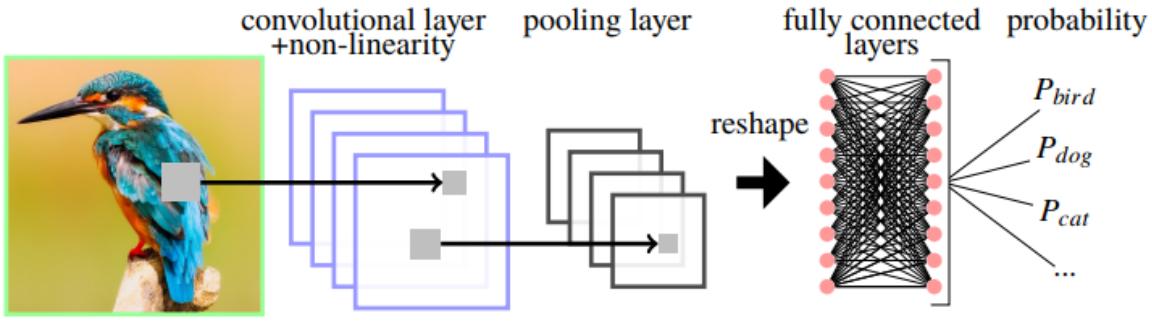


Figure (II.1) : Les réseaux de neurones convolutifs

Ce code CNN en sortie de la partie convolutive est ensuite branché en entrée d'une deuxième partie, constituée de couches entièrement connectées. 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.[14]

Toute l'idée derrière CNN a commencé avec notre cerveau. Le cerveau humain traite facilement l'image, où elle passe à travers la rétine sous forme de signal électrique vers le cortex visuel primaire rempli de grandes couches denses de cellules. Il extrait diverses informations de l'image comme le bord, des parties de l'image, pertinemment CNN utilise divers filtres pour extraire des informations de l'image d'entrée.

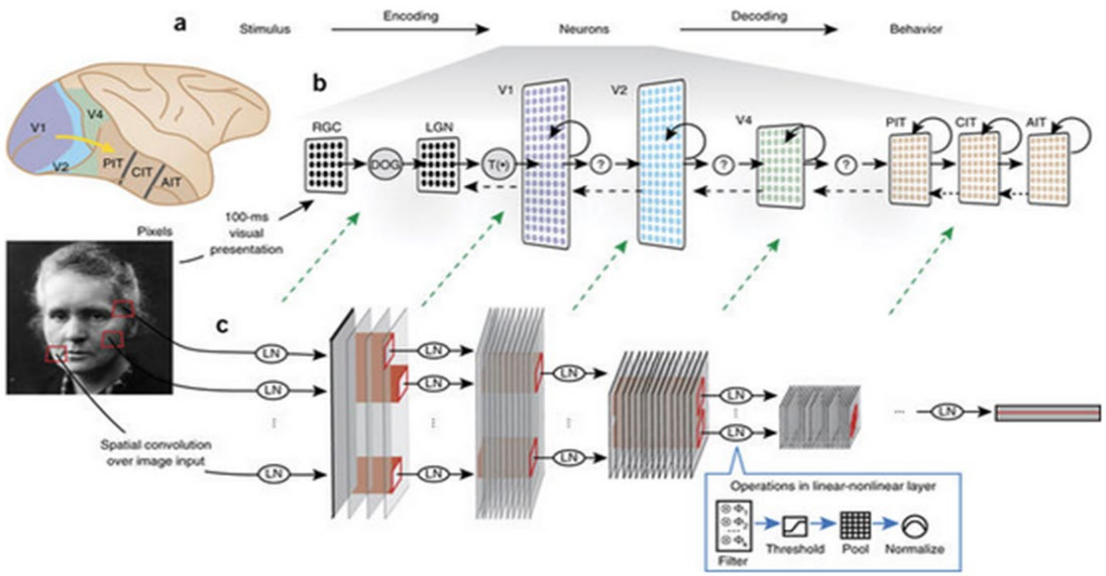


Figure (II.2) : Réseau de neurones convolutifs vs cerveau humain

II.2 CNN et vision par ordinateur :

La vision par ordinateur est un domaine interdisciplinaire de l'informatique qui vise à faire en sorte que les ordinateurs traitent et analysent des images ou des vidéos et en extraient des détails de la même manière qu'un esprit humain (achats en ligne aux diagnostics médicaux).

Un CNN (réseau de neurones convolution els) est un modèle avancé de réseau de neurones artificiels utilisé dans l'apprentissage automatique pour extraire des caractéristiques, telles que la texture et les bords, à partir de données spatiales.

En général, un ANN (réseau de neurones artificiels) apprend de l'ensemble de données (entrées), ajustant leurs paramètres (poids et biais) pour faire des prédictions précises.

Les CNN ont une capacité spéciale : extraire les caractéristiques des images. En effet un CNN est un modèle qui s'attend à recevoir une image en entrée.

II.3 Image sous forme de nombres :

D'un point de vue mathématique, une image en noir et blanc est un tenseur d'ordre 2 (matrice 2D), tandis qu'une image couleur est un tenseur d'ordre 3 (un vecteur de matrices) car il y a trois canaux (RGB).

En mathématiques, un scalaire est un tenseur d'ordre zéro ; un vecteur (ou tableau) est un tenseur du premier ordre ; une matrice est un tenseur du second ordre, etc.

Voyons quelques exemples :

Scalaire 5 → (température par exemple).

Vecteur [3, 5, 2, 0.8] → (vitesse).

Matrice [1, -3, 0.7], [13, 9, -15], [-0.1, 8, 4]] → (image en noir et blanc).

Tenseur du troisième ordre :

[[[1, -3, 0.7], [13, 0.35, -1.5], [-0.1, 59, 4]], [[2.2, -25, 0.9], [-23, 19, 6.5], [-0.1, 81, 4]], [[11, 18, 7.7], [13, 9, -1.5], [-0.1, 44, 17]]] → (une image en couleur).

Tenseur du quatrième ordre : […, [[[1, -3, 0.7], [13, 0.35, -1.5], [-0.1, 59, 4]], [[2.2, -25, 0.9], [-23, 19, 6.5], [-0.1, 81, 4]], [[11, 18, 7.7], [13, 9, -1.5], [-0.1, 44, 17]]], …] → un vecteur de tenseurs du troisième ordre (une vidéo).

II.4 Les architectures de CNN :

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

II.4.1 Couche convolutive :

La convolution est une opération mathématique de deux fonctions qui produit une troisième fonction qui exprime comment la forme de l'une est modifiée par rapport à l'autre.

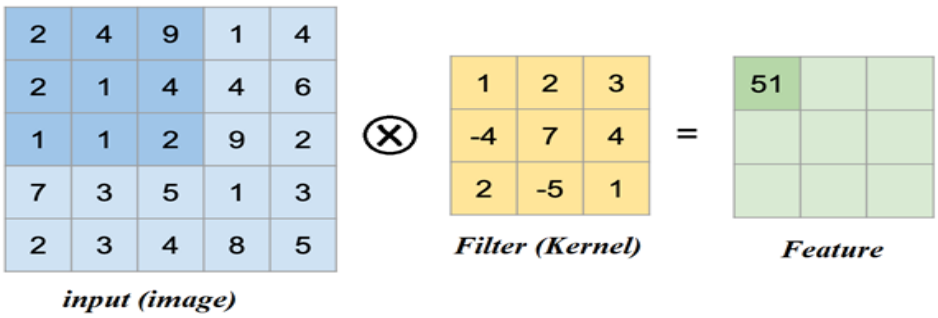


Figure (II.3) : Opération mathématique de la convolution

En vision par ordinateur, la deuxième fonction est appelée "filtre" (ou noyau), qui est une matrice plus petite. Ce filtre se déplace sur toute la matrice de l'image en multipliant ses valeurs par les valeurs de pixels d'origine. Toutes ces multiplications sont additionnées à un nombre à la fin (comme un point scalaire).[15]

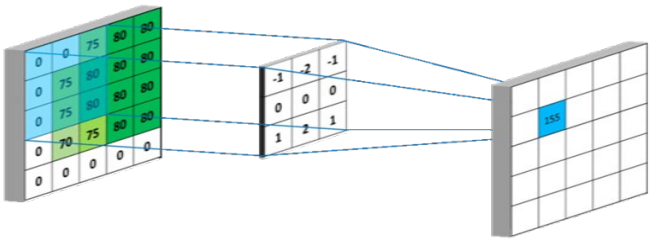


Figure (II.4) : opération convolutive

Convolution Layer est une couche permettant d'extraire les caractéristiques contenues dans une image d'entrée. Également une couche pour calculer l'équation mathématique des entités complexes.

Comme exemple nous filtrons certaines caractéristiques d'une image 5x5 à l'aide d'un filtre 3x3 puis multiplier les fonctionnalités et les filtrer comme indique ci-dessous.

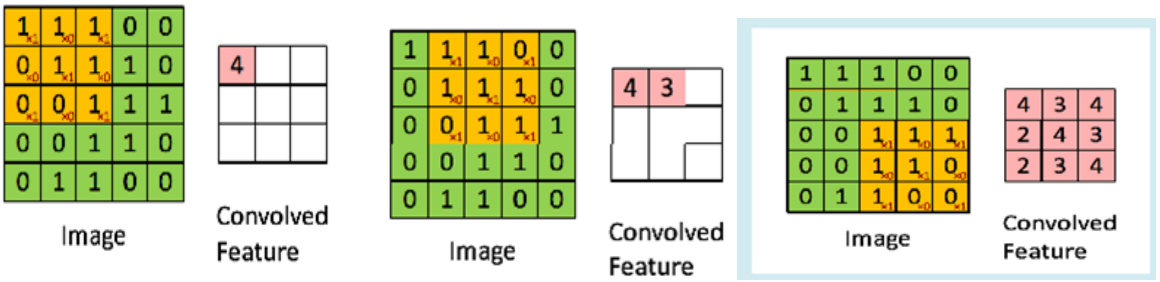


Figure (II.5) : fonction convoluée

II.4.2 Noyaux d'image/Filtres :

Les filtres sont essentiellement des noyaux d'image, qui sont une petite matrice appliquée à une image entière. Les filtres nous permettent de transformer des images en extrayant des informations des images en identifiant les bords, où plusieurs noyaux sont utilisés dans les réseaux de neurones pour identifier les bords. Le noyau est appliqué par élément à la manière d'une fenêtre coulissante.

Nous glissons essentiellement le noyau sur la matrice d'entrée, c'est-à-dire l'image qui est multipliée par les poids du filtre et la somme de ces résultats pour obtenir la fonctionnalité finale. Remarquez comment la résolution diminue parce que nous prenons 9 valeurs d'entrée et sortons à une valeur

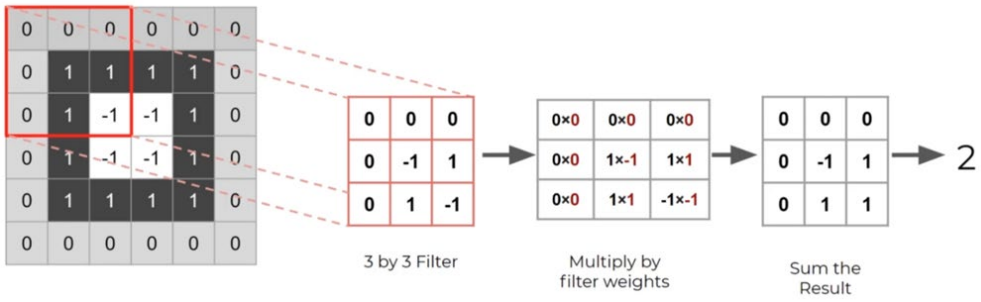


Figure (II.6) : Noyaux d'image/Filtre

II.4.3 stride (foulée) :

Nous pouvons dire foulée comme valeur de décalage de pixel. Si la valeur de foulée est de 1, nous décalons le pixel une fois par fois. Cela peut prendre beaucoup de temps pour détecter les photos à grands pixels. Lorsque nous augmentons la valeur de la foulée, les performances de notre modèle seront meilleures et diminueront également notre précision. Nous devons donc décider quelle valeur de foulée est la meilleure pour notre modèle.[16]

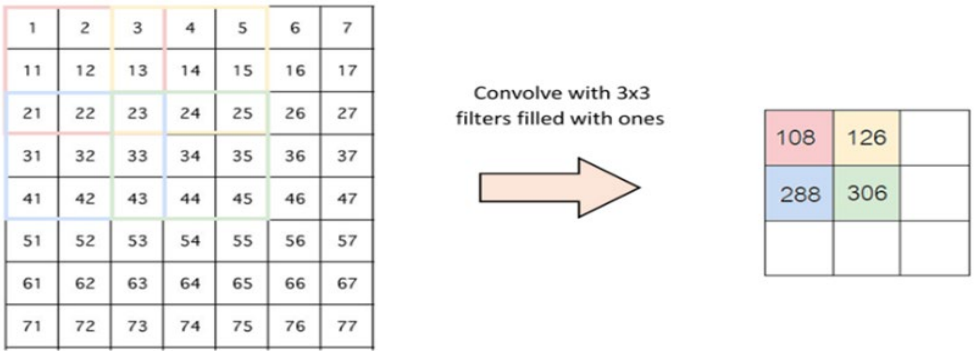


Figure (II.7) : décalage de pixels

La formule générale pour connaître la taille de la matrice de convolution sans remplissage est

(N x N) * (F x F) = (N-F+1) x(N-F+1). Ceci peut être réalisé en donnant 'valide' pour le remplissage de paramètre.

La formule de la taille de la matrice de convolution avec remplissage est (N+2p-F+1) x(N+2p-F+1). Si la taille de remplissage est une, la taille de la matrice résultante est la même que celle de la matrice d'entrée. Ceci peut être réalisé en donnant 'sème' pour le remplissage de paramètre.

Pour décrire quelle serait la taille de rembourrage correcte, cette formule est utilisée p = (F-1) /2.

II.4.4 Rembourrage (Remplissage) :

Lorsque nous mappons nos fonctionnalités à l'aide d'un filtre, notre pixel d'image ne s'adaptera pas comme toujours. Notre solution efficace est devenue de donner du rembourrage aux pixels de notre

image. Nous donnons un rembourrage à notre image avec des valeurs de zéros, également appelées rembourrage à zéro.

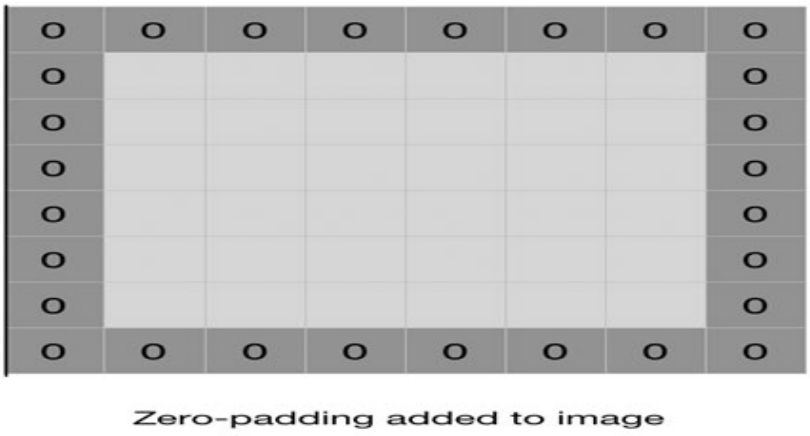


Figure (II.8) : Remplissage d'une image

Si vous utilisez un rembourrage, votre filtre sera adapté à votre image. Déposez la partie de l'image où le filtre ne rentre pas. C'est ce qu'on appelle un rembourrage valide qui ne conserve qu'une partie valide de l'image. Vous avez utilisé le rembourrage mais votre filtre correspond à votre image. C'est ce qu'on appelle le même rembourrage.

Regardez cette photo ! nous utilisons le filtre 3x3 et la foulée 2. C'est l'exemple du même rembourrage.[16]

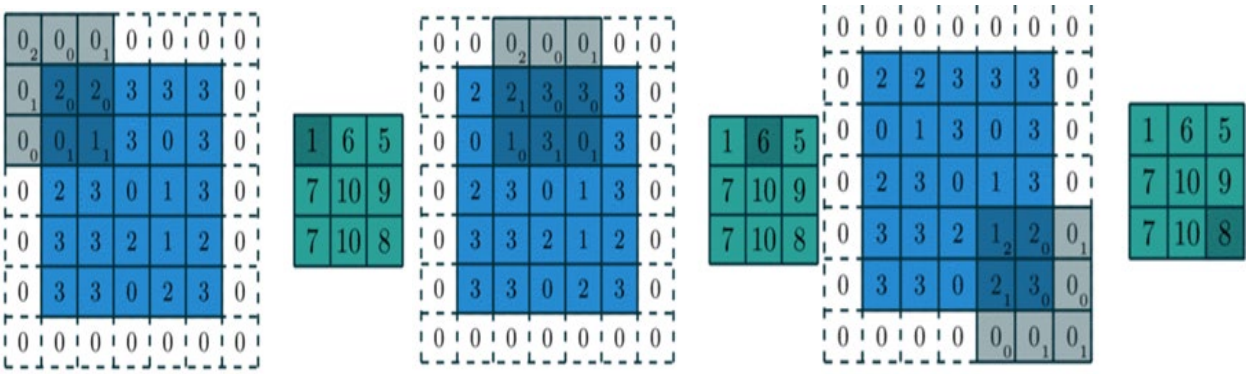


Figure (II.9) : exemple de rembourrage

II.4.5 Couche non linéaire :

Cette couche est ajoutée après chaque couche dense ou de convolution. Il utilise une fonction d'activation pour détecter la non-linéarité des données. La non-linéarité signifie que la variation de la sortie n'est pas proportionnelle à la variation de l'entrée. Nous avons besoin de cette non-linéarité

comme si le réseau était linéaire, il n'y aurait aucun sens à ajouter plusieurs couches (plusieurs couches linéaires sont équivalentes à une seule couche).

Plus il y a de couches cachées (moyennes), plus le processus d'apprentissage est profond et plus il devient facile de capturer les non-linéarités (modèles complexes). C'est pourquoi on l'appelle Deep Learning.

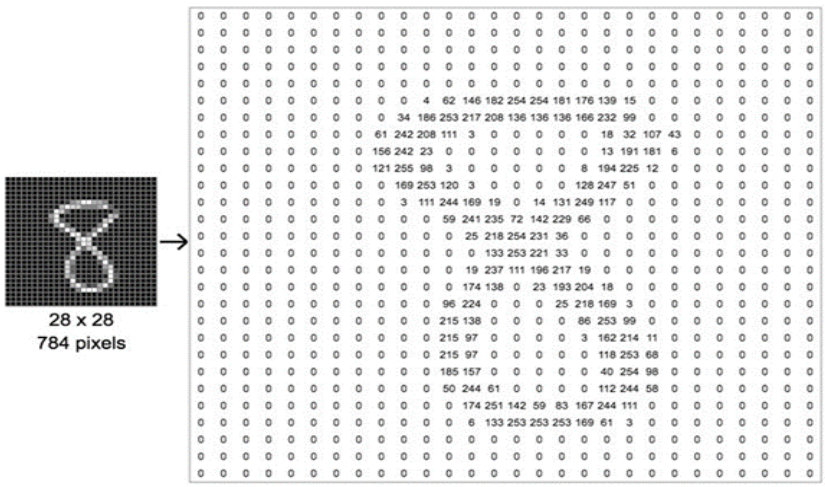


Figure (II.10) : Couche non linéaire

II.4.6 Motif non linéaire :

Un modèle de réseau approprié devrait être capable de trouver de nouveaux motifs complexes dans les images. La fonction d'activation la plus utilisée est l'unité linéaire rectifiée (Relu).

Relu signifie Rectifie Linéaire Unit pour un fonctionnement non linéaire. Le but de Relu est d'introduire la non-linéarité dans notre ConvNet. Puisque les données du monde réel voudraient que notre ConvNet apprenne, ce seraient des valeurs linéaires non négatives. Nous utilisons ReLu pour obtenir une valeur positive à partir de la sortie de notre modèle. Si les valeurs de sortie sont inférieures à 0, il suppose que 0. Ƒ(x) = max (0, x) ← c'est l'équation de ReLu.

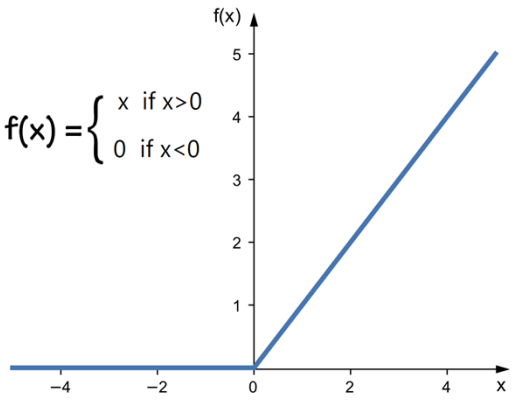


Figure (II.11) : La fonction d'activation

II.4.7: couche de regroupement :

La couche de mise en commun est utilisée pour réduire davantage la taille de la matrice. La forme la plus courante d'une couche de regroupement est un filtre de taille 2 × 2 ; appliqué avec une foulée de deux échantillons sur la largeur et la hauteur, rejetant 75% des activations (il réduit de moitié les dimensions).

La couche de regroupement est généralement utilisée pour sélectionner les pixels les plus importants en utilisant la fonction de regroupement Max qui sélectionne uniquement le pixel de valeur la plus élevée présent dans le filtre. Cela réduit la quantité de calcul requise pour la formation, réduisant ainsi considérablement le temps nécessaire à la formation du réseau neuronal.[15]

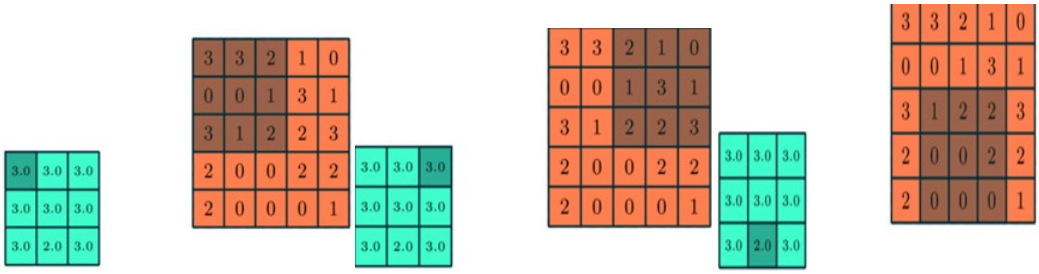


Figure (II.12) : couche de regroupement

La mise en commun spatiale également appelée sous-échantillonnage ou sous-échantillonnage qui réduit la dimensionnalité de chaque carte mais conserve des informations importantes. Dans un algorithme ultérieur comme U-Net, nous pouvons voir un sur échantillonnage.

- Types de sous-échantillonnage sont :

• Max Pooling (prendre la plus grande valeur de pixel)

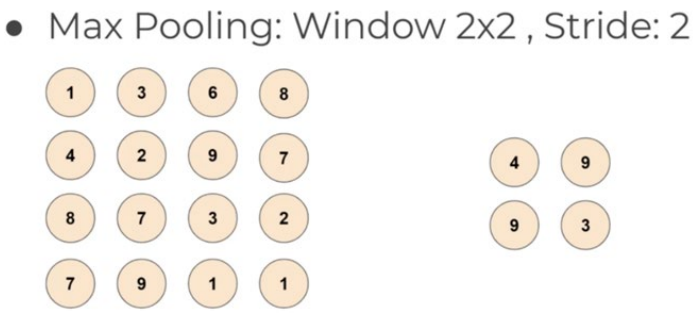


Figure (II.13) : Max Pooling

• Pooling moyen (prenez la moyenne — somme des valeurs/nombre de tailles de filtre)

• Sum Pooling (prendre la valeur somme du pixel).[16]

II.4.8 Couche entièrement connectée :

La couche dense est une couche de réseau de neurones profondément connectée, ce qui signifie que chaque neurone de la couche dense reçoit une entrée de tous les neurones de sa couche précédente. La couche dense s'avère être la couche la plus couramment utilisée dans les modèles d'apprentissage en profondeur.

Une couche dense, également appelée couche entièrement connectée, est une couche qui aide à modifier la dimensionnalité de la sortie de la couche précédente afin que le modèle puisse facilement définir la relation entre les valeurs des données dans lesquelles le modèle fonctionne.

En arrière-plan, la couche dense effectue une multiplication matrice-vecteur. Les valeurs utilisées dans la matrice sont en fait les paramètres qui sont mis à jour pendant le processus de formation à l'aide de la technique de rétropropagation à travers les équations d'apprentissage.

- Équations d'apprentissage :

$$\omega_i \mathrel{+}= -LearningRate * \frac{\partial Loss}{\partial \omega_i} \quad Weight \qquad \text{(II .1)}$$

$$b_i \mathrel{+}= -LearningRate * \frac{\partial Loss}{\partial b_i} \quad Bais \qquad \text{(II .2)}$$

$$f_i \mathrel{+}= -LearningRate * \frac{\partial Loss}{\partial f_i} \quad Filter \qquad \text{(II.3)}$$

La couche que nous appelons couche FC, nous avons aplati notre matrice en vecteur et l'avons introduite dans une couche entièrement connectée comme un réseau de neurones. Nous voyons généralement la couche FC dans la dernière partie d'un bloc de code. Et aussi un dernier processus de modèle.[15]

Soft max est une fonction mathématique qui convertit un vecteur de nombres [5.2, 16.9, 0.3, 1.7] en un vecteur de probabilités [0.13, 0.81, 0.01, 0.05]

Où chaque valeur nous indique si l'entrée (l'image) est un cercle, un triangle, un carré ou un hexagone. Notez que la probabilité totale est égale à un.

$$s\left(x_i\right) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

Après l'activation de la fonction soft max, le résultat final est :

1) Environ 13 %

2) Triangle 81 %

3) Carré 1%

4) Hexagone 5%

Cela signifie qu'un triangle est prédit !

Maintenant, étant donné une image, si nous voulons prédire si une personne sourit ou non, la tâche devient beaucoup plus complexe, et un simple ANN échouerait, à moins que nous ayons un très grand ensemble de données.

Heureusement, les chercheurs ont mis au point une technique avancée appelée CNN (réseau neuronal convolutif) qui fonctionne très bien, extrayant des modèles complexes et faisant ainsi des prédictions précises avec un ensemble de données abordable.

Les CNN sont capables de faire la reconnaissance de formes au point de distinguer une personne qui sourit d'une personne qui est sérieuse.

Avec CNN, votre caméra devient pratiquement un œil humain.

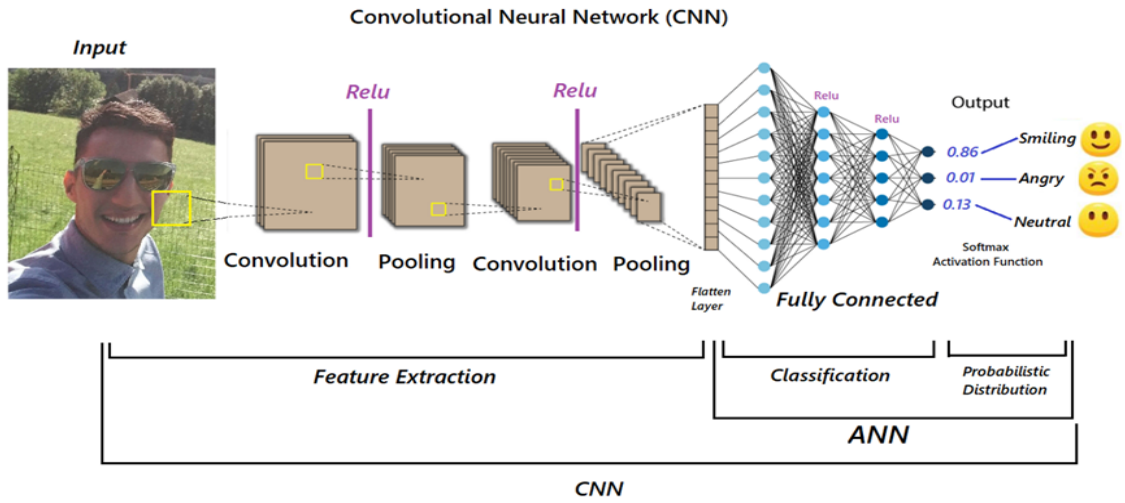


Figure (II.16) : OPERATION PRINCIPALE DANS LE CNN.[15]

II.5 Avantages de CNN :

Un avantage majeur des réseaux convolutifs est l'utilisation d'un poids unique associé aux signaux entrant dans tous les neurones d'un même noyau de convolution. Cette méthode réduit l'empreinte mémoire, améliore les performances et permet une invariance du traitement par translation. C'est le principal avantage du CNN par rapport au MLP, qui lui considère chaque neurone indépendant et donc affecte un poids différent à chaque signal entrant. Lorsque le volume d'entrée varie dans le temps (vidéo ou son), il devient intéressant de rajouter un paramètre de temporisation (delay) dans le paramétrage des neurones. On parlera dans ce cas de réseau neuronal à retard temporel (TDNN).[17]

II.6 Plateforme :

Afin d'implémenter notre projet, le système doit être configuré avec un environnement dans lequel les packages keras, tensorflow, pandas, scikit-learn et matplotlib sont installés et télécharger le jeu de données et le placer dans le même répertoire que la plateforme Anaconda.

II.6.1 Anaconda :

Anaconda est une distribution gratuite et à code source ouvert des langages de programmation Python et R pour les applications liées à la science des données et à l'apprentissage automatique, qui vise à simplifier la gestion et le déploiement des packages. Vous pouvez le télécharger à partir du lien ci-dessous, en fonction de votre système.[18]

II.6 .2 Spyder :

Spyder est un environnement de développement intégré comme IDLE créé pour la programmation scientifique. Il est codé en Python et pour Python ! Spyder est livré avec plusieurs fonctionnalités telles qu'un éditeur de code, une console interactive et finalement un explorateur de variables.

II.6.3 Tensorflow : TensorFlow est une bibliothèque de logiciels open-source permettant de programmer des flux de données pour diverses tâches**.**[19]

II.6.4 Keras :

Keras est une bibliothèque de réseau de neurones open source écrite en Python. Activez Tensorflow environnement et installez keras à l'aide de «pip install keras ».

II.7 Conclusion :

Dans ce chapitre, nous avons présenté les réseaux de neurones convolution els qui 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 » (réceptive 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 convolution els 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).

CHAPITRE III

CONCEPTION DES GAN

III.1 Introduction :

Les réseaux antagonistes génératifs appartiennent à l'ensemble des modèles génératifs. Cela signifie qu'ils sont capables de produire/générer de nouveaux contenus. Ce n'est un secret pour personne que les réseaux antagonistes génératifs (GAN) sont devenus un énorme succès dans le monde de la vision par ordinateur pour générer des images hyperréalistes. Certains des échantillons produits par les variantes GAN les plus récentes sont étonnants. En voici quelques-unes extraites d'un article récent :

Figure (III.1) : Illustration des capacités des GAN par Ian Goodfellow et co-auteurs.

S'appuyant sur leur succès en matière de génération, les GAN d'images ont également été utilisés pour des tâches telles que l'augmentation des données, le suréchantillonnage d'images, la synthèse texte-image et, plus récemment, la génération basée sur le style, qui permet de contrôler les fonctionnalités fines et grossières dans les éléments générés.

Naturellement, cette capacité à générer de nouveaux contenus donne aux GAN un aspect un peu "magique", du moins à première vue. Dans les parties suivantes, nous allons surmonter la magie apparente des GAN afin de plonger dans les idées, les mathématiques et la modélisation derrière ces modèles. Non seulement nous discuterons des notions fondamentales sur lesquelles s'appuient les réseaux antagonistes génératifs mais, de plus, nous construirons étape par étape et en commençant par le tout début le raisonnement qui mène à ces idées.[20]

III.2 Présentation de la structure du GAN :

Un réseau antagoniste génératif (GAN) comporte deux parties :

• Le générateur apprend à générer des données plausibles. Les instances générées deviennent des exemples d'apprentissage négatifs pour le discriminateur.

• Le discriminateur apprend à distinguer les fausses données du générateur des vraies données. Le discriminateur pénalise le générateur pour avoir produit des résultats non plausibles.

Lorsque l'entraînement commence, le générateur produit des données manifestement fausses, et le discriminateur apprend rapidement à dire que c'est faux :

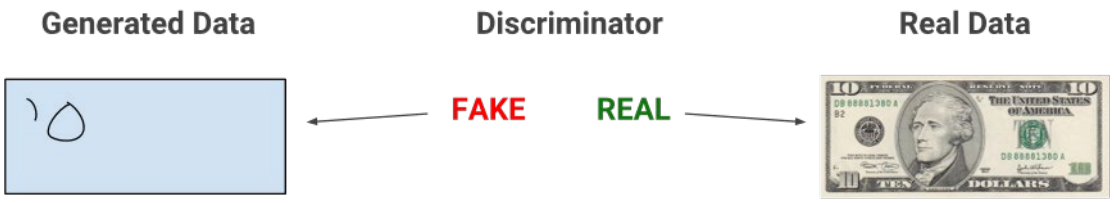


Figure (III.2) : Distinguez les générateurs de données pseudos réels.

Au fur et à mesure que l'entraînement progresse, le générateur se rapproche de la production d'une sortie qui peut tromper le discriminateur :

Figure (III.3) : Modèle de déception du générateur pour le discriminateur

Enfin, si la formation du générateur se passe bien, le discriminateur devient moins efficace pour faire la différence entre le vrai et le faux. Il commence à classer les fausses données comme réelles et leur précision diminue.

Figure (III.4) : Classer les fausses données comme réelles.[21]

III.2 .1 Le discriminateur :

Le discriminateur dans un GAN est simplement un classifieur. Il essaie de distinguer les données réelles des données créées par le générateur. Il peut utiliser n'importe quelle architecture de réseau appropriée au type de données qu'il classe.

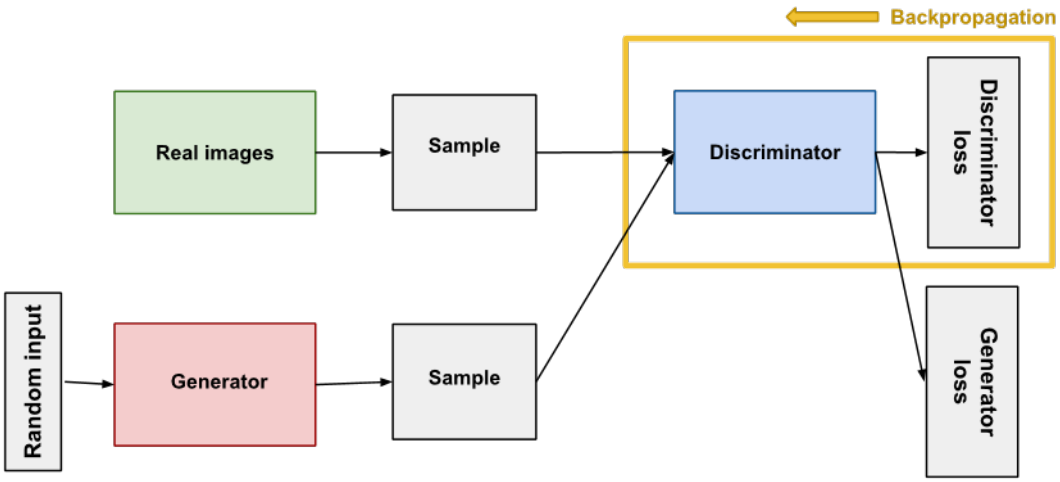


Figure (III.5) : Rétropropagation dans la formation des discriminateurs.

Dans la figure 5, les deux cases "Sample" représentent ces deux sources de données alimentant le discriminateur. Pendant l'entraînement du discriminateur, le générateur ne s'entraîne pas. Ses poids restent constants pendant qu'il produit des exemples sur lesquels le discriminateur peut s'entraîner.[22]

III.2.2 Le générateur :

La partie génératrice d'un GAN apprend à créer de fausses données en incorporant les commentaires du discriminateur. Il apprend à faire en sorte que le discriminateur classe sa sortie comme réelle.

L'entraînement du générateur nécessite une intégration plus étroite entre le générateur et le discriminateur que ne l'exige l'entraînement du discriminateur. La partie du GAN qui entraîne le générateur comprend :

• entrée aléatoire

• réseau générateur, qui transforme l'entrée aléatoire en une instance de données

• réseau discriminateur, qui classe les données générées

• sortie du discriminateur

• perte du générateur, qui pénalise le générateur pour ne pas avoir trompé le discriminateur

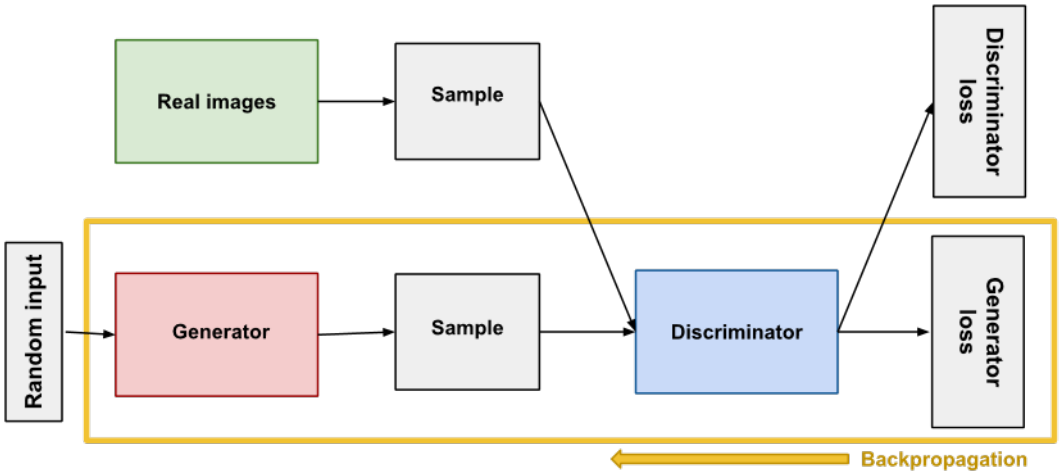


Figure (III.6) : Rétropropagation dans la formation des générateurs.[23]

III.3 Classification des 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 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.

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 rétro propagation, pratiquée depuis le milieu des années1980.[24]

III.4 Comment fonctionnent les GAN :

Réseaux antagonistes génératifs (GAN) sont un modèle génératif avec estimation de densité implicite, faisant partie de l'apprentissage non supervisé et utilisant deux réseaux de neurones. Ainsi, on entend les termes « génératifs » et « réseaux » dans « réseaux génératifs antagonistes.

III.4.1 Le principe : générateur vs discriminateur :

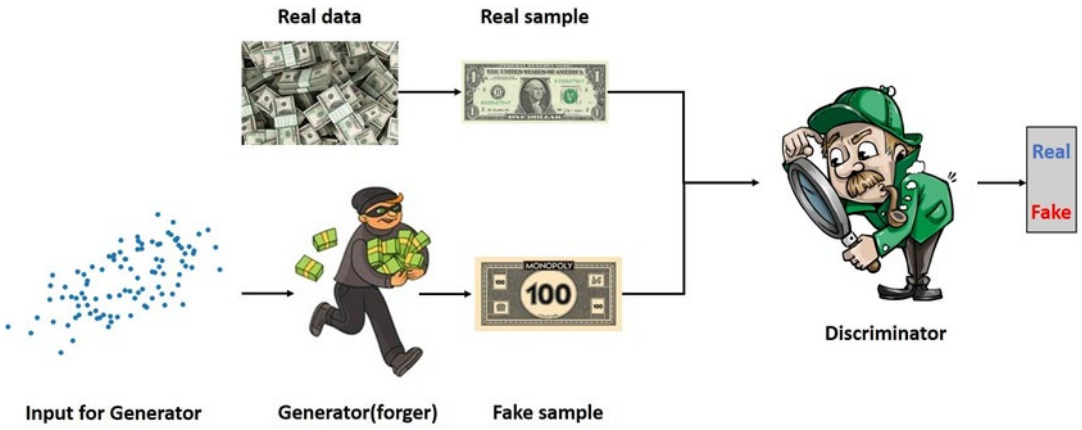


Figure (III.7) : Rôles du générateur et du discriminateur.[25]

III.4.2 Entraînement générateur et discriminateur :

Au cours de l'entraînement, le générateur s'améliore progressivement pour créer des images qui semblent réelles, tandis que le discriminateur s'améliore pour les différencier. Le processus atteint l'équilibre lorsque le discriminateur ne peut plus distinguer les vraies des fausses images. Voir Figure 8. Ainsi, si le discriminateur est bien entraîné et que le générateur parvient à générer des images réalistes qui trompent le discriminateur, alors nous avons un bon modèle génératif : nous générons des images qui ressemblent à l'ensemble d'entraînement.

Après cette phase d'apprentissage, nous n'avons besoin que du générateur pour échantillonner de nouvelles (fausses) données réalistes. Nous n'avons plus besoin du discriminateur. A noter que le bruit aléatoire garantit que le générateur ne produit pas toujours la même image (ce qui peut tromper le discriminateur).

Notez qu'au début de l'apprentissage de la figure 8, le générateur ne génère qu'un bruit aléatoire qui ne ressemble pas aux données d'apprentissage.

Figure (III.8) : Entraînement générateur et discriminateur.

- Nous formons donc le générateur selon les étapes suivantes :

1. Entrer un exemple de bruit aléatoire. Cette phase de formation consiste à alimenter le générateur en données aléatoires (sous forme de distribution) pour simuler le bruit

2. obtenir une sortie de générateur à partir d'un bruit aléatoire échantillonné. Au cours de cette étape, le générateur génère des images aléatoires que le discriminateur utilise pour identifier les motifs.

3. classification "Real" ou "Fake" du discriminateur pour la sortie du générateur. Le discriminateur apprend ou évalue des caractéristiques à partir de ses entrées afin de distinguer les vraies données des fausses données.

4. Calculer la perte à partir de la classification du discriminateur.

5. Rétropropagation à travers le discriminateur et le générateur pour obtenir des gradients. À travers le réseau, le discriminateur produit une certaine probabilité et la différence entre les résultats prédits et les résultats réels est rétro propagée.

6. Utilisation des gradients pour modifier uniquement les poids du générateur. Les poids du discriminateur sont mis à jour. Dans cette phase, la rétropropagation est stoppée à la fin du discriminateur, et le générateur n'est pas mis à jour.[23]

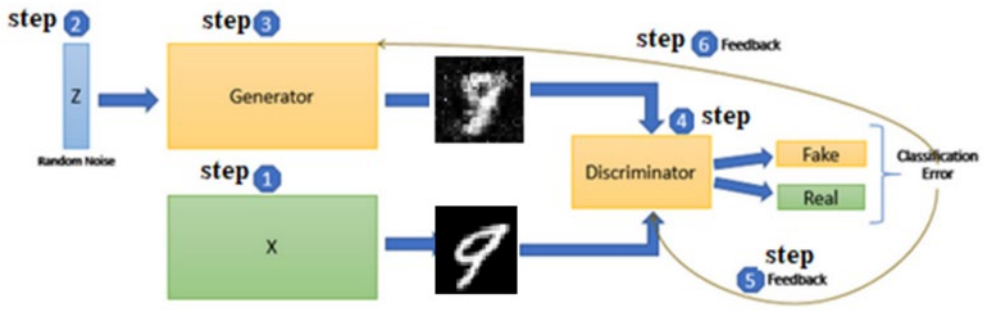


Figure (III.9) : La procédure de formation des GAN.

III.5 Mathématiques impliquées dans la modélisation GAN :

Les réseaux antagonistes génératifs font référence à une famille de modèles génératifs qui cherchent à découvrir la distribution sous-jacente derrière un certain processus de génération de données. Cette distribution est découverte à travers une compétition contradictoire entre un générateur et un discriminateur. Les deux modèles sont formés de telle sorte que le discriminateur s'efforce de faire la distinction entre les exemples générés et vrais, tandis que le générateur cherche à confondre le discriminateur en produisant des données aussi réalistes et convaincantes que possible.

III.5 .1 La fonction de perte :

Le GAN peut être vu comme une interaction entre deux modèles différents : le générateur et le discriminateur. Par conséquent, chaque modèle aura sa propre fonction de perte x représente les données réelles, z le vecteur latent, G(z) les fausses données, D(x) l'évaluation du discriminateur des données réelles, D(G(z)) l'évaluation du discriminateur des fausses données.

III.5 .2 Le discriminateur :

L'objectif du discriminateur est d'étiqueter correctement les images générées comme fausses et les points de données empiriques comme vrais. Par conséquent, nous pourrions considérer ce qui suit comme la fonction de perte du discriminateur

$$L_D = L(D(x), 1)) + L(D\big(G(z)\big), 0) \qquad \text{(III.1)}$$

III.5 .3 Le générateur :

Nous pouvons continuer et faire la même chose pour le générateur. Le but du générateur est de confondre le plus possible le discriminateur de sorte qu'il étiquette à tort les images générées comme étant vraies

$$L_G = L\big(D\big(G(z)\big), 1\big) \qquad \text{(III.2)}$$

La clé ici est de se rappeler qu'une fonction de perte est quelque chose que nous souhaitons minimiser. Dans le cas du générateur, il doit s'efforcer de minimiser la différence entre 1, l'étiquette des vraies données, et l'évaluation par le discriminateur des fausses données générées.

III.5 .4 Entropie croisée binaire :

Une fonction de perte courante utilisée dans les problèmes de classification binaire est l'entropie croisée binaire. La formule de l'entropie croisée ressemble à :

$$H_{(p,q)} = E_{x\sim p(x)}[\_\log q(x)] \qquad \text{(III.3)}$$

Dans les tâches de classification, la variable aléatoire est discrète. Par conséquent, l'espérance peut être exprimée comme une sommation.

$$H_{(p,q)} = \_\sum_{x\epsilon X}[p(x)\log q(x)] \qquad \text{(III.4}$$

Nous pouvons encore simplifier l'expression dans le cas de l'entropie croisée binaire, puisqu'il n'y a que deux étiquettes : zéro et un.

$$H_{(y,\hat{y})} = \_\sum y\log(\hat{y}) + (1\ \_y)\log(1\_\hat{y}) \qquad \text{(III.5)}$$

Il s'agit de la fonction de perte que nous avons vaguement utilisée dans la section ci-dessus. L'entropie croisée binaire remplit notre objectif en ce qu'elle mesure à quel point deux distributions sont différentes dans le contexte de la classification binaire déterminant si un point de données d'entrée est vrai ou faux. En appliquant cela à chacune des fonctions de perte, nous obtenons

$$L_D = \_\sum_{x\epsilon X, z\epsilon\varsigma}\log\big(D(x)\big) + \log 1\_D(G(z)) \qquad \text{(III.6)}$$

$$L_G = \_\sum_{z\epsilon\varsigma}\log(D(G(z)) \qquad \text{(III.7)}$$

Nous avons maintenant deux fonctions de perte avec lesquelles on peut entraîner le générateur et le discriminateur. Notez que, pour la fonction de perte du générateur, la perte est faible si D(G(z)) est proche de 1, puisque log (1) = 1. C'est exactement le type de comportement que nous attendons d'une fonction de perte du générateur.

III.5 .5 Remarque Importante :

L'article original de Goodfellow présente une version légèrement différente des deux fonctions de perte dérivées ci-dessus. Goodfellow procède en le présentant comme un jeu min-max, où le discriminateur cherche à maximiser la quantité donnée tandis que le générateur cherche à réaliser l'inverse.

$$\min_G \max_D \log\big(D(x)\big) + \log(1\_D(G(z))) \qquad \text{(III.8)}$$

La formulation originale est une ligne concise qui démontre intuitivement la nature contradictoire de la composition entre le générateur et le discriminateur. Cependant, en pratique, nous définissons des fonctions de perte séparées pour le générateur et le discriminateur comme nous l'avons fait ci-dessus. En effet, le gradient de la fonction y = log(x) est plus raide près de x=0 que celui de la fonction y = log(1-x), ce qui signifie qu'essayer de maximiser log(D(G(z)) (ou de manière équivalente, minimiser - log(D(G(z)))).[26]

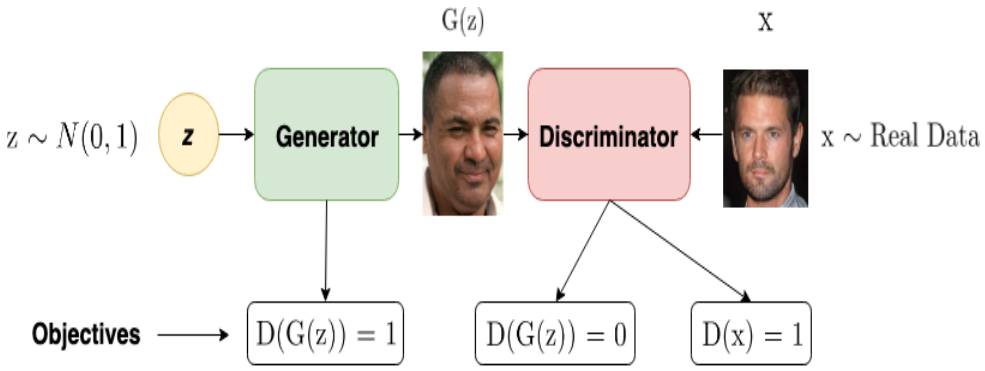


Figure (III.10) : Représentation mathématique

III.6 GAN Conditionnel ou infos GAN :

Le réseau peut apprendre à produire des images avec des caractéristiques catégorielles spécifiques (telles que les chiffres de 0 à 9) et des caractéristiques continues (telles que l'angle

de rotation des chiffres), de manière non supervisée. De plus, comme l'apprentissage n'est pas supervisé, il est capable de trouver les motifs cachés parmi les images et de générer des images qui suivent ces motifs cachés.

III.6.1 Concept Info GAN :

Dans InfoGAN, pour contrôler les types d'images produites, nous devons fournir des informations supplémentaires sur les bruits aléatoires au générateur et le forcer à utiliser les informations lors de la création de fausses images. Les informations supplémentaires que nous alimentons doivent concerner les types de fonctionnalités que nous voulons que les images aient. Par exemple, si nous voulons produire des chiffres MNIST spécifiques, nous devons alimenter un vecteur catégoriel contenant des entiers de 0 à 9 ; si nous voulons produire des chiffres MNIST avec différents angles de rotation, nous pouvons vouloir alimenter des nombres flottants sélectionnés au hasard entre -1 et 1.

Il est facile de fournir des informations supplémentaires, car nous avons juste besoin d'ajouter des entrées supplémentaires au modèle de générateur. Mais comment s'assurer que le générateur utilisera l'information au lieu de l'ignorer complètement ? Si nous formons toujours le générateur simplement en fonction de la réponse du discriminateur, le générateur n'utilisera pas les informations supplémentaires car les informations supplémentaires n'aideront pas le générateur à créer des images plus réalistes (il est seulement utile de générer des caractéristiques spécifiques des images). Ainsi, nous devons appliquer des "pénalités" supplémentaires sur le générateur s'il n'utilise pas les informations supplémentaires. Une façon consiste à ajouter un réseau supplémentaire (souvent appelé réseau auxiliaire et noté Q) qui prend de fausses images et reproduit les informations supplémentaires que nous avons introduites dans le générateur. De cette façon, le générateur est obligé d'utiliser l'information supplémentaire comme s'il ne le faisait pas, il n'y a aucun moyen pour que le réseau auxiliaire puisse reproduire correctement l'information supplémentaire, et le générateur sera « pénalisé ». L'image ci-dessous résume les structures du GAN (à gauche) et d'InfoGAN (à droite).

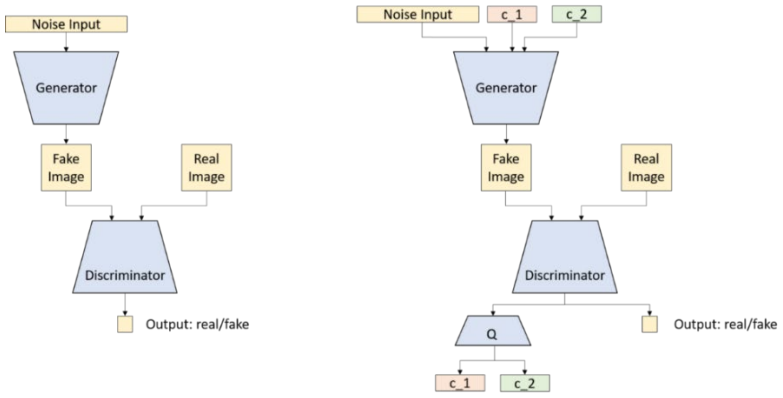


Figure (III.11) : Structures du GAN (à gauche) et d'InfoGAN (à droite). Image de l'auteur

III.6.1.1 Jeu Minimax utilisant des informations mutuelles :

Dans la théorie de l'information, l'information mutuelle entre X et Y, I (X, Y), mesure la "quantité d'informations" tirée de la connaissance de la variable aléatoire Y sur l'autre variable aléatoire X.

- L'information mutuelle peut être exprimée comme la différence de deux termes d'entropie :

$$I(X;Y) = H(X) - H(X/Y) = H(Y) - H(Y/X) \qquad \text{(III.10)}$$

- I (X ; Y) est la réduction de l'incertitude sur X lorsque Y est observé.
- Si X et Y sont indépendants, alors I (X ; Y) = 0, car connaître une variable ne révèle rien sur l'autre.

En revanche, si X et Y sont liés par une fonction déterministe et inversible, alors l'information mutuelle maximale est atteinte.

- Des objectifs similaires inspirés par l'information mutuelle ont déjà été envisag és dans le contexte du clustering [23-25].
- Cette interprétation permet de formuler facilement un coût :

Étant donné tout x ~ P_G (x), nous voulons que P_G (c | x) ait une petite entropie. En d'autres termes, les informations contenues dans le code latent c ne doivent pas être perdues dans le processus de génération.

- Dans le GAN normal, le jeu minimax est :

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}}[\log D(x)] + E_{z\sim noise}[\log(1-D(G(Z)))] \quad \text{(III.11)}$$

- Maintenant, le jeu minimax à informations régularisées suivant est résolu :

$$\min_G \max_D V_I(D,G) = V(D,G) - \lambda I(c; G(z,c)) \quad \text{(III.12)}$$

III.6.1.2 Maximisation de l'information mutuelle variation elle :

- En pratique, le terme d'information mutuelle I (c ; G (z, c)) est difficile à maximiser directement car il nécessite l'accès au postérieur P (c | x).
- LI (G, Q) est ajouté aux objectifs du GAN sans modification de la procédure de formation du GAN, ce qui en résulte Information Maximizing Générative Adversarial Networks (Info GAN).
- Info GAN est défini comme le jeu minimax suivant avec une régularisation variation elle des informations mutuelles LI (G, Q) et un hyperparamètre λ :

$$\min_{G,Q} \max_D V_{InfoGAN}(D,G,Q) = V(D,G) - \lambda L_I(G,Q) \quad \text{(III.13)}$$

III.6.2 Cadre Info GAN :

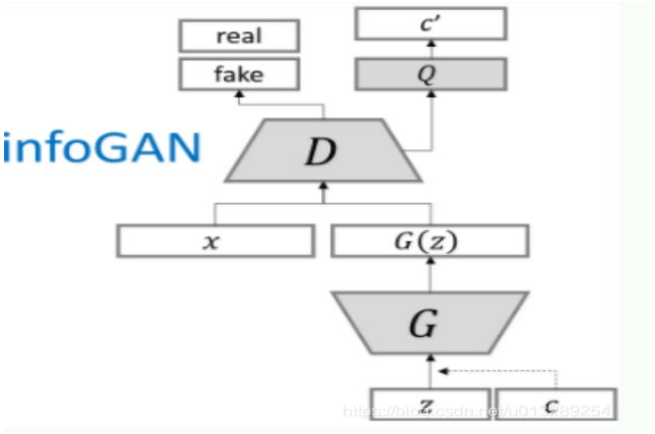


Figure (III.12) : structure de de l'Info-GAN

- La distribution auxiliaire Q est paramétrée comme un réseau de neurones.
- Dans la plupart des expériences, Q et D partagent toutes les couches convolutives et il existe une couche finale entièrement connectée aux paramètres de sortie pour la distribution conditionnelle Q (c | x), ce qui signifie qu'Info GAN n'ajoute qu'un coût de calcul négligeable à GAN .

- On observe que *LI* (G, *Q*) converge toujours plus rapidement que les objectifs GAN normaux et donc Info GAN est essentiellement gratuit avec GAN.
- Pour le code latent catégoriel *ci*, nous utilisons le choix naturel de la non- linéarité soft max pour représenter *Q* (ci | *x*).
- Pour le code latent continu *cj*, il y a plus d'options en fonction de ce qui est le vrai *Q* postérieur (cj | *x*). Dans les expériences, il suffit de traiter *Q* (cj | *x*) comme une gaussienne factorisée.
- Les expériences sont basées sur des techniques existantes introduites par DCGAN.
- Il suffit de régler λ sur 1 pour les codes latents discrets. Lorsque le code latent contient des variables continues, un λ plus petit est généralement utilisé. [27]

III.9 Conclusion :

Le GAN est devenu une technique très populaire et répandue dans diverses industries pour résoudre une grande variété de problèmes. Il peut sembler plus facile de les former, mais cela ne fonctionne pas de cette façon car ils ont besoin de deux réseaux pour le faire, ce qui les rend instables. Qu'ils soient utilisés pour le bien ou pour le mal, les GAN sont capables d'imiter n'importe quelle distribution d'informations. On apprend souvent à faire en sorte que les GAN créent des mondes qui reflètent presque le nôtre dans n'importe quel domaine : images, musique, discours ou prose. Ce sont des robots artistes dans un sens, et leur production est impressionnante, voire poignante. Mais ils voudront même générer de faux contenus médiatiques et sont la technologie qui sous-tend Deep Fakes. Dans ce chapitre, nous avions discuté du concept des GAN et des cas d'utilisation et de la mise en œuvre du GAN. Plus les GAN deviennent précis et avancés, plus la valeur qu'ils peuvent apporter aux entreprises est importante. Les applications des GAN se sont rapidement développées, notamment pour les images. En outre, Les GAN peuvent être utilisés pour augmentation des données quand on n'a qu'une centaine d'images et qu'on souhaite en avoir plus.

Des GAN ont également été développés pour des sorties binaires (malade ou non) ou des sorties discrètes (tension artérielle arrondie, poids arrondi…) [28]. Les bénéfices de ces nouvelles recherches sur les données tabulaires sont nombreux, en particulier pour intimité fins. Par exemple, au lieu d'envoyer des données confidentielles à partir de feuilles Excel, les hôpitaux

peuvent envoyer de fausses données réalistes (qui maintiennent la corrélation entre les caractéristiques) à leurs partenaires.

CHAPITRE IV

APPLICATION GENERATION DE VISAGE AVEC LES GAN

IV.1. Introduction :

Ce chapitre donnera une introduction aux GAN à travers un exemple. Nous formerons un réseau contradictoire génératif (GAN) pour générer de nouveaux visages célébrités après lui avoir montré des photos de nombreuses célébrités réelles. Ce document donnera une explication détaillée de l'implémentation et éclairera comment et pourquoi ce modèle fonctionne. De plus, pour gagner du temps, il sera utile d'avoir un GPU, ou deux.

Les GAN sont un cadre pour apprendre à un modèle DL à capturer la distribution des données de formation afin que nous puissions générer de nouvelles données à partir de cette même distribution. Ils sont constitués de deux modèles distincts, un générateur et un discriminateur. Le travail du générateur est de créer de "fausses" images qui ressemblent aux images d'entraînement. Le travail du discriminateur consiste à regarder une image et à indiquer s'il s'agit ou non d'une image d'entraînement réelle ou d'une fausse image fausse du générateur.

Pendant la formation (entrainement), le générateur essaie constamment de déjouer le discriminateur en générant de mieux en mieux des faux, tandis que le discriminateur s'efforce de devenir un meilleur détective et de classer correctement les images réelles et fausses. L'équilibre de ce jeu est lorsque le générateur génère des faux parfaits qui semblent provenir directement des données d'apprentissage, et le discriminateur doit toujours deviner à 50 % de confiance que la sortie du générateur est réelle ou fausse. Les GAN sont des réseaux de neurones capables de générer des images.

Cependant, ces réseaux ont besoin d’un ensemble de données d’entraînement important ainsi que d’une forte puissance de calcul en parallèle afin de produire des images pouvant être confondues avec les images d’entraînement.

Le discriminateur et le générateur ne fonctionnent pas simultanément comme indique dans la figure suivante :

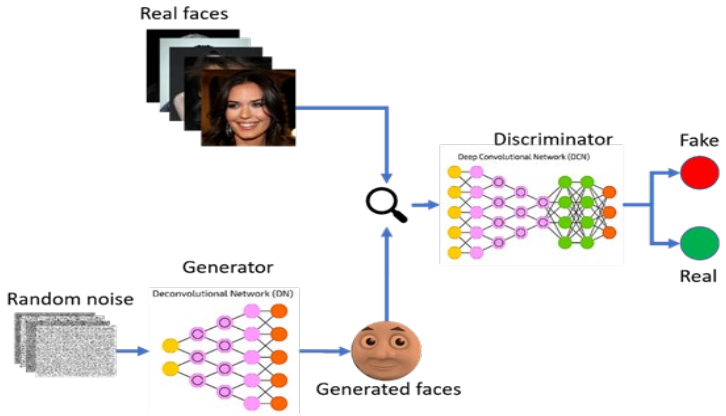


Figure (IV.1) : Schéma de l'entrainement d'un gan

IV.2 Implémentations :

Dans cette section de notre mémoire, nous allons essayer d'implémenter un modèle de génération de visage à partir de rien et obtenir des résultats de haute qualité similaires à certaines des méthodes de pointe.

La réalisation de ce projet s'est faite en plusieurs étapes, définies par l'état de nos recherches en matière de GAN et des résultats obtenus. Ces différentes étapes vont être brièvement expliquées dans le reste de ce chapitre.

- Implémentation du GAN simple (couche entièrement connectées) qui génère des chiffres manuscrits MNIST.
- Implémentation du GAN conditionnelle à convolution profonde qui génère des visages personnalisés des acteurs du cinéma indien Bollywood.
- Implémentation du GAN conditionnelle à convolution profonde qui génère des visages personnalisés en fonction de la saisie textuelle.

IV.2.1 Préparation de l'environnement :

Notre environnement est composé de :

L'IDE Pycharm (V22) ; Anaconda (Conda18) pour les scientifiques et Python (Version 3.10) avec des packages nécessaire pour le code GAN

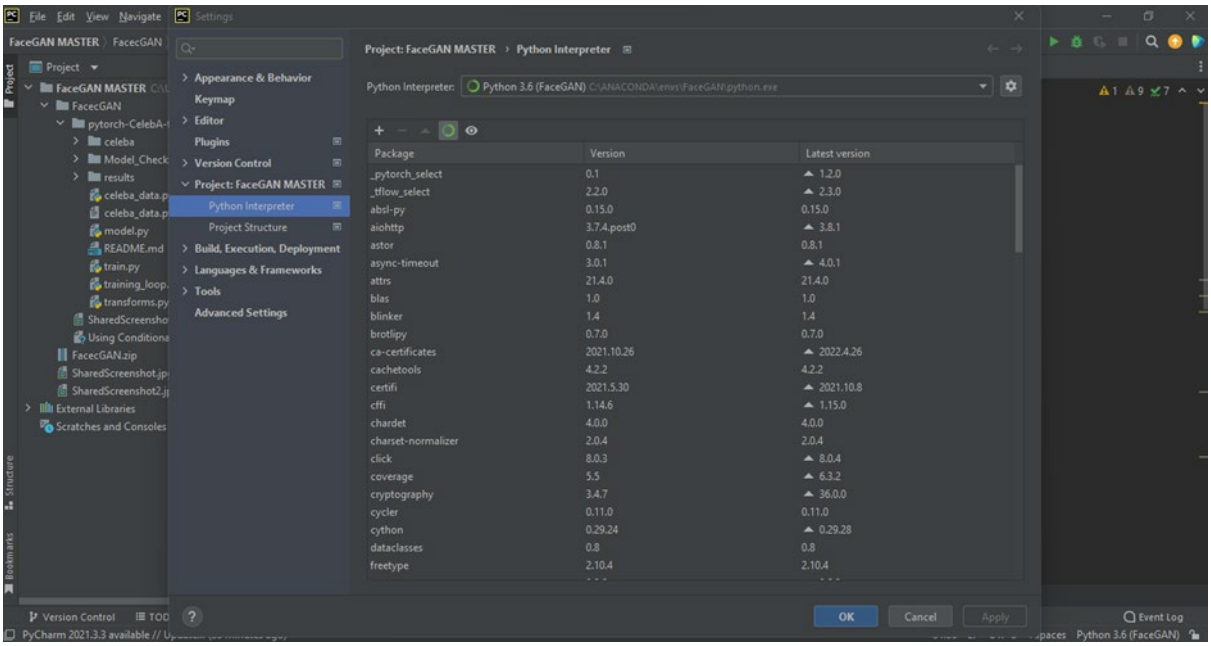


Figure (IV.2) : Composants de l'environnement python

IV.2.2 Présentation des packages pour Python GAN :

Notre programme est base sur Pytorch comme indique sur la figure suivante :

```
import torch

import torch.optim as optim

from model import *

from transforms import *

from training loop import *

from celeba_data import *

from torch.utils.data import Data Loader
```

Figure (IV.3) : programme de base sur Pytorch

IV.3 Génération des chiffres MNIST :

Dans cette première partie nous allons essayer une implémentation simple qui utilise un GAN à créer des images de chiffres de l'ensemble de données MNIST comme indique ci-dessous.

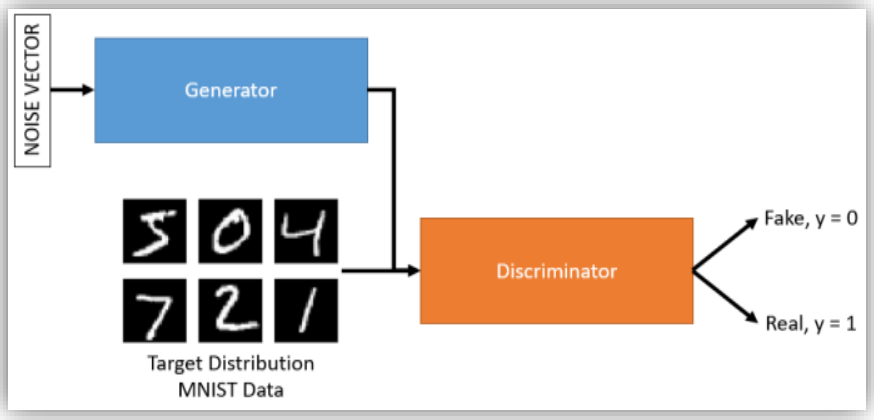
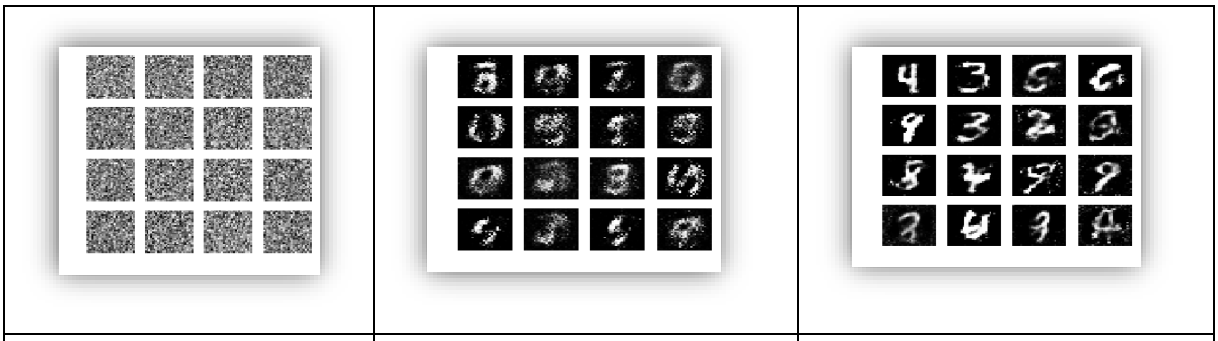


Figure (IV.4) : GAN pour la génération des chiffres MNIST

IV.3.1 Résultats :



1 époque	20 époques	80 époques

Figure (IV.5) : génération des chiffres MNIST durant training

Après environ 5 époques, les chiffres sont flous et satisfaisants après 80 époques.

On constate que celui qui a plus d'époques est beaucoup plus clair. Car au début de l'entrainement, deux situations peuvent se produire. Le générateur ne sait pas comment créer des images qui ressemblent à celles du jeu d'entraînement. Le discriminateur ne sait pas comment catégoriser les images reçues comme vraies ou fausses.

IV.4 Génération des visages de Bollywood avec le DCGAN :

Nous utiliserons le DC-GAN qui est une extension directe du GAN, sauf que les couches de convolution et de transposition de convolution sont explicitement utilisées dans le discriminateur et le générateur, respectivement comme indique ci-dessous.

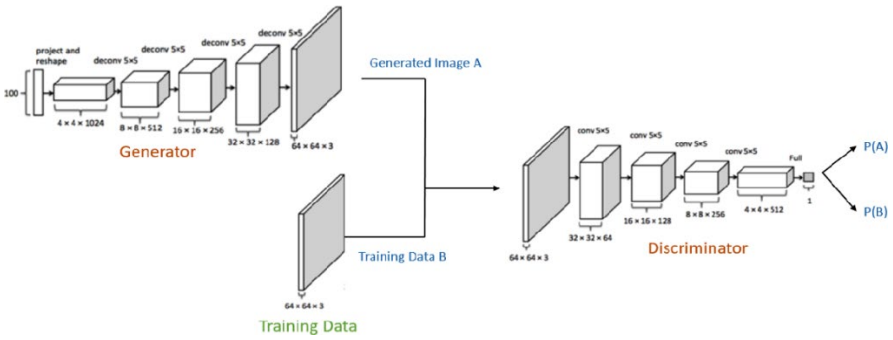


Figure (IV .6) : DC-GAN architecture

IV.4.1 Base de données :

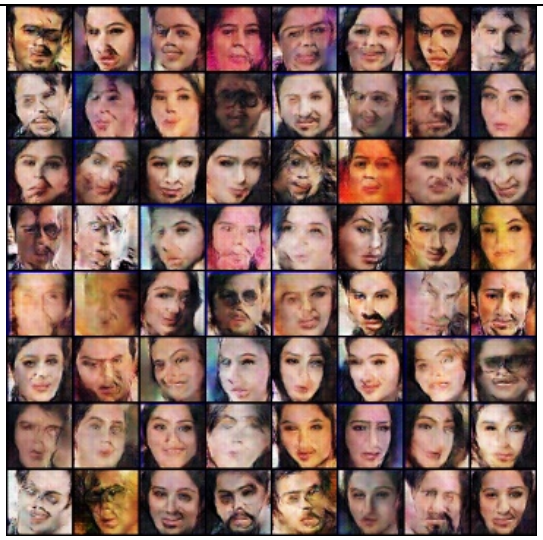
Le jeu de données Bollywood Celeb faces Dataset disponible sur Kaggle, qui comprend plus de 8664 visages de célébrités Bollywood recadrées. Notez que la modélisation générative est une tâche d'apprentissage non supervisée, les images n'ont donc pas besoin d'avoir d'étiquettes.

Pour charger cet ensemble de données sur PyTorch, j'ai utilisé l'Image Folderclasse de torchvision. J'ai également redimensionné et recadré les images à 64x64 px, et normalisé les valeurs de pixels avec une moyenne et un écart type de 0,5 pour chaque canal. Cela garantira que les valeurs de pixel sont dans la plage, ce qui est plus pratique pour entraîner le discriminateur. Nous allons également créer un chargeur de données pour charger les données par lots.

L'entrée du générateur est généralement un vecteur ou une matrice de nombres aléatoires (appelé tenseur latent) qui est utilisé comme bruit pour générer une image. Le générateur convertira un tenseur latent de forme (64, 1, 1) en un tenseur d'image de forme (3 x 28 x 28). Pour y parvenir, nous utiliserons la couche ConvTranspose2d de PyTorch, qui est exécutée comme une convolution transposée (également appelée déconvolution).

IV.4.2 Résultats :

5 époques



10 époques

15 époques

20 époques

Figure (IV.7) : génération des images durant training

Étant donné que les sorties du générateur sont des images, il n'est pas évident de savoir comment entraîner le générateur. C'est là que nous employons une astuce plutôt élégante, qui consiste à utiliser le discriminateur comme une partie de la fonction de perte.

• On génère un lot d'images à l'aide du générateur, on le passe dans le discriminateur.

• Nous calculons la perte en définissant les étiquettes cibles sur 1, c'est-à-dire réel. Nous faisons cela parce que l'objectif du générateur est de "tromper" le discriminateur.

• Nous utilisons la perte pour effectuer une descente de gradient, c'est-à-dire modifier les poids du générateur, de sorte qu'il améliore la génération d'images réelles pour "tromper" le discriminateur.

IV.5 Information Maximizing Generative Adversarial Nets (cGAN):

Dans cette partie nous allons implémenter un GAN à convolution profonde conditionnelle (cGAN) dans PyTorch qui utilise le texte anglais comme étiquettes au lieu de nombres uniques. Notre modèle est un GAN à convolution profonde (DCGAN), c'est-à-dire qu'il utilise des couches à convolution profonde dans son architecture au lieu de couches entièrement connectées comme dans l'implémentation précédente.

Info GAN est défini comme le jeu minimax suivant avec une régularisation variationnelle des informations mutuelles LI (G, Q) et un hyperparamètre λ :

$$\min_{G,Q} \max_{D} V_{InfoGAN}\ (D, G, Q) = V(D, G) - \lambda\, L_I(G, Q) \qquad \text{(IV. 1)}$$

- Pour le code latent catégoriel c_i, nous utilisons le choix naturel de la non- linéarité soft max pour représenter Q (ci | x).
- Pour le code latent continu c, il y a plus d'options en fonction de ce qui est le vrai Q postérieur (cj | x). Dans les expériences, il suffit de traiter Q (cj | x) comme une gaussienne factorisée.
- Les expériences sont basées sur des techniques existantes introduites par DCGAN.
- Il suffit de régler λ sur 1 pour les codes latents discrets. Lorsque le code latent contient des variables continues, un λ plus petit est généralement utilisé.

IV.5.1 Base de données :

La procédure que nous suivrons pour la construction efficace de ce projet est de nous assurer que nous recueillons les meilleurs ensembles de données pour la tâche requise. Nos options consistent à utiliser des ensembles de données de haute qualité disponibles sur Internet ou les télécharger a travers les sites spécialisé. L'ensemble de données Celeb Faces Attributs (CelebA) a été choisi dans notre implémentation de nos réseaux génératifs de reconnaissance faciale.

La base des données contient des photos (Faces) (de 3500 jusqu'à 220.000), comme indiquer dans la figure suivante :

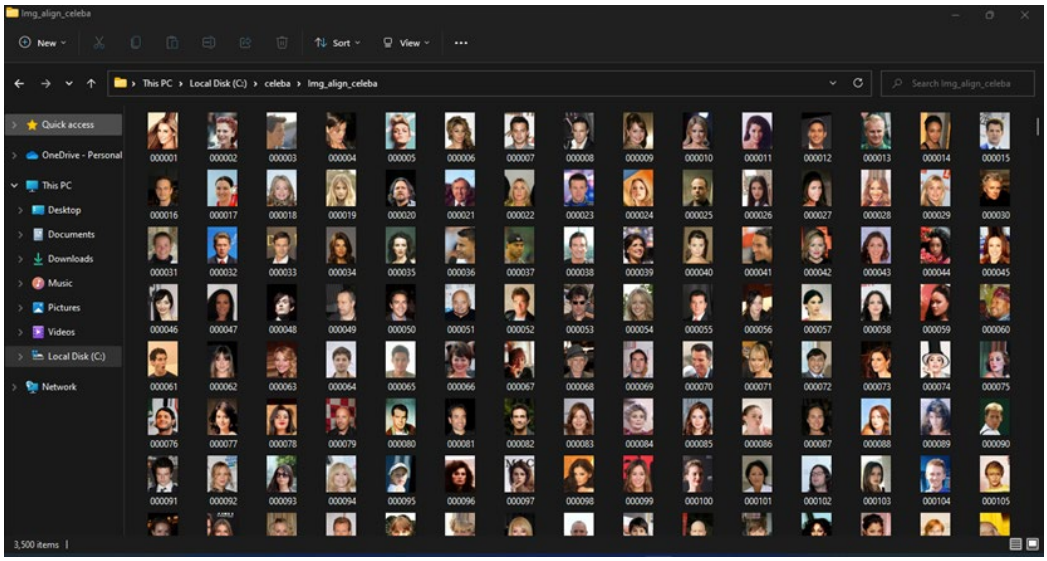


Figure (IV.8) : Base des données d'images

Celeb Faces attribues Dataset (Celeb A) est un ensemble de données d'attributs de visage à grande échelle avec plus de 200 000 images de célébrités, chacune avec 40 annotations d'attributs. Les images de cet ensemble de données couvrent de grandes variations de pose et un encombrement d'arrière-plan. CelebA a de grandes diversités, de grandes quantités et des annotations riches, y compris :

- 10 177 nombres d'identités,
- 202 599 nombres d'images de visage, et
- 5 points de repère,
- 40 annotations d'attributs binaires par image.

L'ensemble de données CelebA de visages de célébrités est entrainé avec des images recadrées à 64x64. Notre architecture de modèle contient cinq couches convolutionnelles/transposées avec normalisation par lots et activation Leaky ReLU ainsi qu'une activation sigmoïde pour la couche de sortie du discriminateur et une activation tanh pour la couche de sortie du générateur. L'optimiseur Adam et la perte d'entropie croisée binaire sont utilisés.

IV.5.2 Structure du programme :

Une fois que nous aurons accédé à l'ensemble de données, nous construirons à la fois les modèles générateur et discriminateur pour effectuer la tâche de génération de visage. L'ensemble de la construction architecturale se concentrera sur la création du cadre contradictoire pour générer l'image et classer les interprétations les plus réalistes. Après la

construction de l'ensemble de la construction architecturale, nous commencerons la procédure de formation (entrainement) et enregistrerons les images en conséquence. Enfin, nous enregistrerons les modèles générateur et discriminateur afin de pouvoir les réutiliser une autre fois.

Notre structure de program contient une Dossier parent (Appelé Pytorch-FACEGAN-MASTER) et 03 dossiers fils sont (celeba pour base des donnée, Model checkpoint [Facultatif], et Résultat) ; et 05 ficher python comme la figure suivant l'affiche :

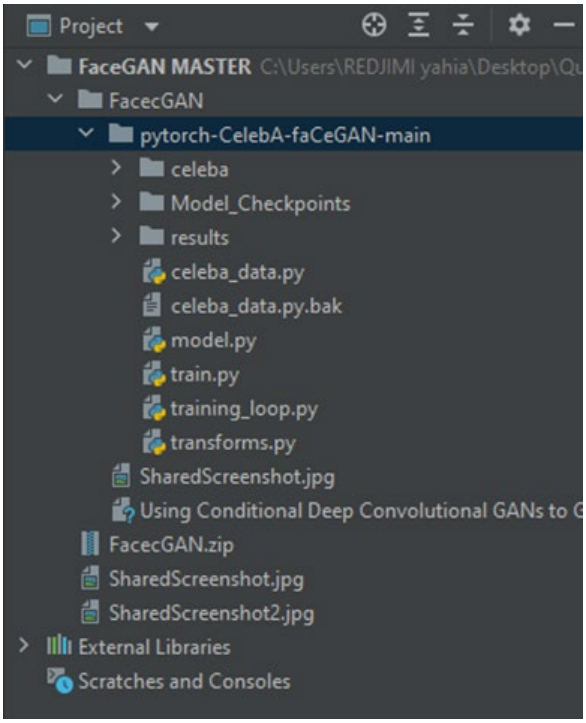


Figure (IV.9) : structure de program

Après avoir défini la conception nous devons écrire notre modèle dans PyTorch comme ci-dessus.

Le jeu de données, CelebA qui est livré avec 40 annotations d'attributs binaires par image comme (Young, male, Glasses, old, Bald, …etc.)

En utilisant ces attributs binaires comme étiquettes ou conditions, nous pouvons former notre cGAN pour générer des visages avec des caractéristiques spécifiques que nous contrôlons. De plus, nous pouvons sélectionner plusieurs attributs pour nous entraîner et apprendre en les concaténant simplement pour former des étiquettes multidimensionnelles. Avec suffisamment d'attributs, nous pouvons générer une grande variété de visages avec différentes

caractéristiques humaines. Par exemple, notre résultat de formation préliminaire a montré que le modèle pouvait apprendre les attributs binaires, [Male, Young], nous permettant de générer 4 combinaisons différentes de types de visage (jeune homme, "pas jeune" homme, jeune femme, "pas jeune" femelle)

Pendant l'apprentissage, ces étiquettes sont concaténées avec l'échantillon de données à l'aide d'une couche torch.cat (), après avoir traversé chacune une couche convolutive/déconvolutive. Le code pour créer et traiter les étiquettes dans les dimensions correctes se trouve dans la boucle de formation, qui est essentiellement la même qu'une boucle de formation GAN normale, sauf avec des étapes supplémentaires pour alimenter les étiquettes avec des échantillons de données.

IV.5.3 Résultats :

En utilisant la base de données publique CelebA comportant plus de 200 000 images de visages de célébrités pour exécution sur notre PC (l'exécution sur PC local), nous avons choisis 20 itérations (Epochs) seulement à cause de la limitation de notre computer (CPU et RAM). Les résultats suivants ont été obtenus (générés) :

En raison des limitations de notre ordinateur (CPU et RAM), Nous avons choisi la base de données publique CelebA de plus de 200 000 photos de visages de célébrités à exécuter pour 20 itérations seulement comme première étape.

IV.5.3.1 Images générées après 20 itérations :

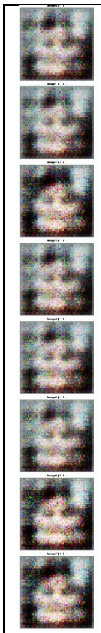
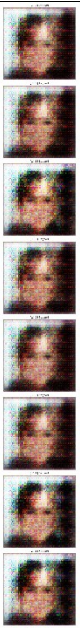
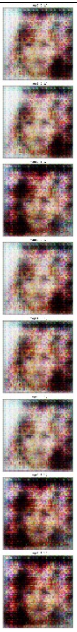
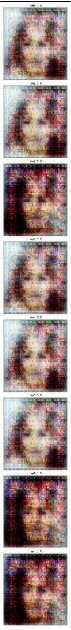
			
1 epo	4 epochs	10 epochs	20 epochs

Figure (IV.10) : visage de célébrités généré après 20 itérations

- Total Base Examples: 3500

Running on a CPU

No Epochs	Disc_loss	Gen_loss	Disc_Real Accuracy	Disc_Fake Accuracy
1	3.52	30.6	0.541	1
5	0.505	2.5	0.835	0.849
10	0.338	2.68	0.858	0.87
20	0.346	2.43	0.881	0.886

Tableau (IV.1) : Base exemple

IV.5.4 Boucle d'entraînement complète :

Définissons une fit fonction pour entraîner le discriminateur et le générateur en tandem pour chaque lot de données d'entraînement. Nous utiliserons l'optimiseur Adam avec certains paramètres personnalisés (bêtas) connus pour bien fonctionner pour les GAN. Nous enregistrerons également des exemples d'images générées à intervalles réguliers pour inspection dont les résultats des scores et pertes sont illustres dans les figures suivantes.

Dans ce cas, nous pouvons voir que le discriminateur fait un excellent travail pour identifier et classer correctement les lots de données, c'est-à-dire 1 pour les vraies images et 0 pour les faux lots d'images.

En conséquence, notre système a généré des visages, mais avec une qualité des images générées médiocre en raison de la faible résolution des images dans l'ensemble de données actuel. Néanmoins, même avec un ensemble de données d'images de faible qualité, on pouvait améliorer nos résultats en exécutant notre programme avec un nombre d'époques assez élevé mais avec un computer plus puissant ce qui demande temps d'entraînement élevé aussi. Dans le cas échéant, il faut avoir recors au laboratoire de google en l'occurrence COLAB.

IV.6 Google colab :

Les résultats dans le titre précédente sont limités par la configuration de matériel

Notre Pc config est :

CPU : ryzen 3 3400

GPU: Integer

RAM: 24 G (3 RAM X 08 G)

Ce configue est faible par port la base de données disponible donc pour meilleur résultat et pour des raisons d'optimisation le code on mignons vers google Colab

IV.6.1 Présentation :

Qu'est-ce que Colab ?

Colab, ou "Colaboratory", vous permet d'écrire et d'exécuter Python dans votre navigateur, avec :

Aucune configuration requise

Accès gratuit aux GPU

Partage facile

Notre code dans Colab :

meme chose pour Colab on a utilize pytorch colab:

IV.6.2 The PyTorch INFERNO package:

• Fournit un wrap per de formation DNN pour PyTorch, avec un système de rappel étendu

• Système d'inférence de base pour la minimisation NLL avec nuisances

• Implémentation de la classe abstraite d'INFERNO

• L'utilisateur peut en hériter et l'ajuster pour son cas d'utilisation spécifique

• Comprend également une approximation expérimentale d'INFERNO à utiliser avec les ensembles de données MC systématiques haut/bas utilisés dans HEP

• L'utilisateur n'a pas besoin de décrire analytiquement comment les caractéristiques d'entrée dépendent des nuisances

• Au lieu de cela, les effets de nuisance extraits de l'interpolation entre les ensembles de données systématiques et nominaux

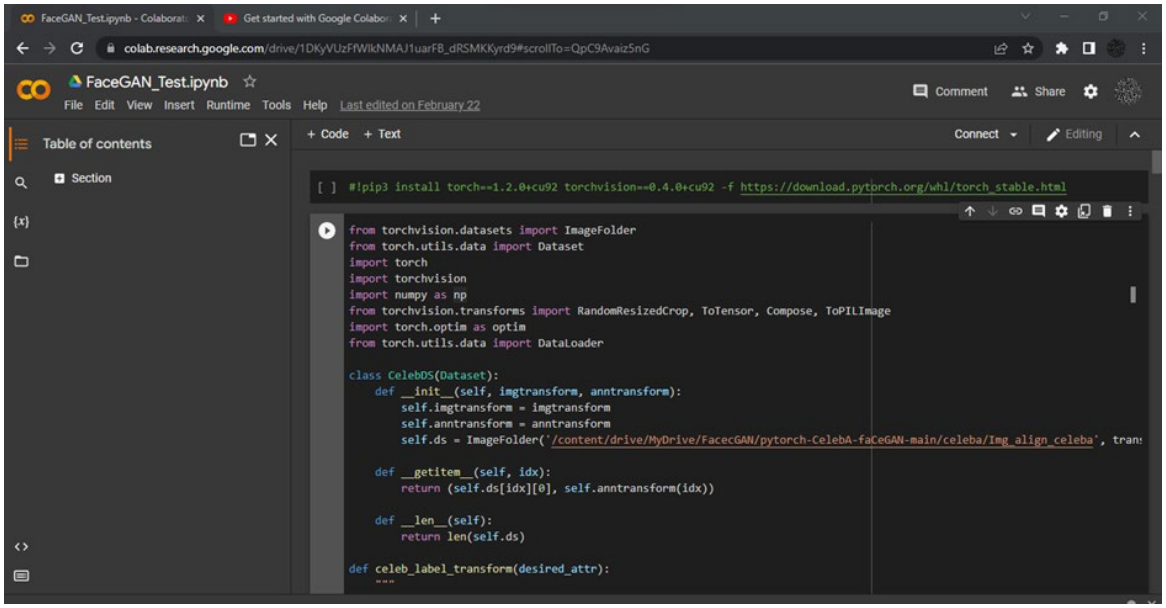


Figure (IV.11): The INFERNO package PyTorch

IV.6.3 L'exécution :

Nous préparerons notre code pour l'exécution pour la première fois dans Colab ; Run time All est la commande pour le faire, pour le code complet voire annexe 1.

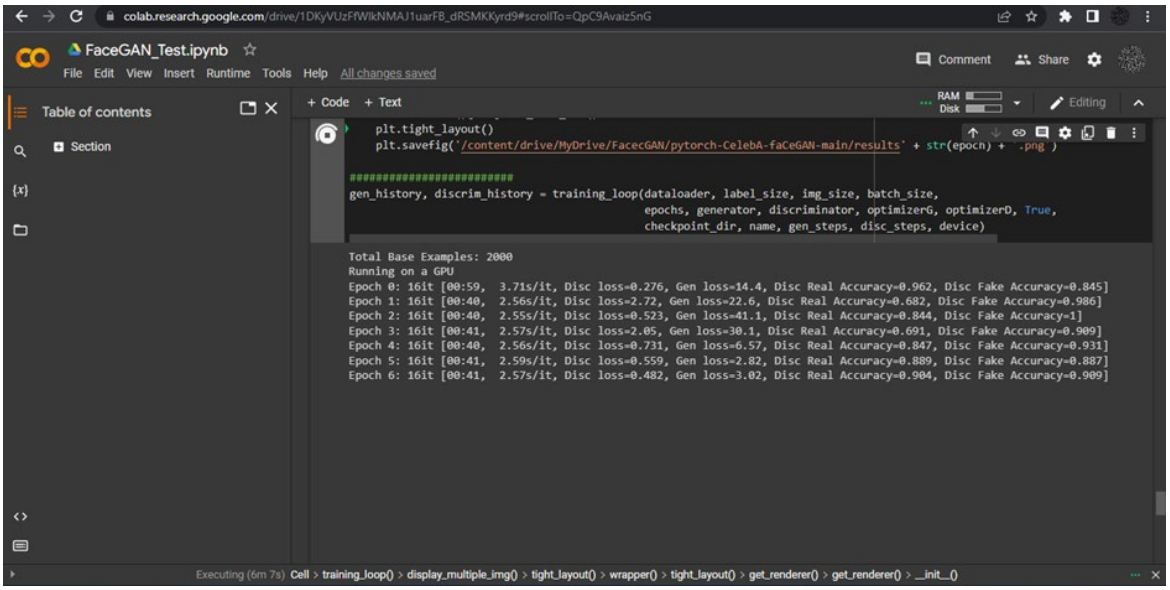


Figure (IV.12) : code pour l'exécution

IV.6.4 Comparaison :

	PC Local	Google Colab
CPU	OUI	OUI
GPU	NO	OUI
Vitesse de l'exécution Epoch 40	Rapid	Lent
Vitesse de l'exécution Epoch 60	Rapid	Très lent
Vitesse de l'exécution Epoch 100	Relativement Rapid	Très lent
Résultat final	Intreptable et acceptable	Mauvaise

Tableau (IV.2) : Comparaison entre PC local et Google Colab

IV.6.5 Commentaire :

- Les GAN fournissent des métriques d'image plus nettes et une texture 2D de meilleure qualité que l'image d'origine, tout en préservant parfaitement le niveau de détail, la couleur, etc.
- Le discriminateur reçoit quant à lui l'image du visage généré fusionné avec le contexte.
- Le générateur doit apprendre à rendre le visage le plus réaliste possible dans son contexte.
- Le but du discriminateur est de pouvoir différencier les données réelles des Données envoyées par le générateur. Cela montre que notre discriminateur est trop bon et que nous devons travailler sur le réseau du générateur afin d'optimiser le générateur pour

tromper le discriminateur en lui faisant croire que le faux ensemble d'images est réellement réel et l'étiqueter comme 1.

IV.7 Conclusion :

Dans ce chapitre nous avons décrit l'environnement de travail, notamment : les outils utilisés pour la réalisation de l'application, les bibliothèques de base, ainsi que la base de données utilisée exploitée.

Le GAN a été entraînée à partir de portraits de célébrités. Les algorithmes de cette intelligence artificielle ont d'abord été entraînés à partir d'une base de données contenant des photos de personnalités. Deux réseaux neuronaux en compétition qui vont alors se former l'un l'autre.

Un réseau créé des contrefaçons, l'autre essaye de les identifier. Mais le plus intéressant advient quand les deux réseaux travaillent conjointement. Ensemble, ils parviennent à produire des contrefaçons plus impressionnantes non seulement de visages, mais aussi des chiffres.

Certains des paramètres et attributs que vous pouvez rendre ajustables sont des traits humains tels que le sexe, l'âge, le type d'émotion, les nombreuses poses de la tête, l'origine ethnique, la tonicité de la peau, la couleur des cheveux, le type d'accessoires portés (tels qu'une casquette ou des lunettes de soleil), et bien plus encore. En rendant tous ces personnages variables avec le réseau antagoniste génératif que vous construisez, de nombreuses variations sont possibles. Un exemple d'exécution d'une telle action est perceptible sur le site Web Face Générateur, où vous pouvez générer près de 11 232 000+ variantes du même visage avec de nombreuses combinaisons. Nous pouvons également faire de telles avancées dans notre modèle d'entraînement pour obtenir des résultats supérieurs tout en ayant un contrôle décent sur le type de structures faciales que nous choisissons de générer.

Conclusion Générale

Conclusion Générale :

Les textes, le son, l'image et la vidéo synthétisés par l'intelligence artificielle (IA) sont transformés en armes à des fins d'imagerie intime non consensuelle, de fraude financière et de campagnes de désinformation. Bien qu'il faille prendre un peu de recul sur les abus de langage parfois liés à l'intelligence artificielle, il faut reconnaître que cette technologie offre des résultats plutôt impressionnants lorsqu'il s'agit de reconstituer des images.

Le GAN est devenu une technique très populaire et répandue dans diverses industries pour résoudre une grande variété de problèmes. Il peut sembler plus facile de les former, mais cela ne fonctionne pas de cette façon car ils ont besoin de deux réseaux pour le faire, ce qui les rend instables. Qu'ils soient utilisés pour le bien ou pour le mal, les GAN sont capables d'imiter n'importe quelle distribution d'informations. On apprend souvent à faire en sorte que les GAN créent des mondes qui reflètent presque le nôtre dans n'importe quel domaine : images, musique, discours ou prose. Ce sont des robots artistes dans un sens, et leur production est impressionnante, voire poignante. Mais ils voudront même générer de faux contenus médiatiques et sont la technologie qui sous-tend Deep Fakes.

Les chercheurs ont employé un GAN pour (Générative Adversarial Network) un outil permettant de limiter l'aide apportée par les humains à l'IA. En effet, lorsqu'un réseau neuronal apprend à reconnaître des images par milliers, il est nécessaire que celles-ci aient été catégorisées au préalable par les êtres humains en fonction de la mission qu'ils souhaitent donner à accomplir. Deux réseaux neuronaux en compétition ou le GAN offre une perspective intéressante pour permettre de limiter ce travail fastidieux, mais néanmoins incontournable dans la formation des algorithmes d'apprentissage. Plutôt que de ne former qu'un seul réseau neuronal, ce procédé consiste à en créer deux, qui vont alors se former l'un l'autre. Le générateur tente de créer de fausses images les plus réalistes possibles. Pendant ce temps, le second réseau cherche à identifier si ces images sont réels ou faux. Un réseau créé des contrefaçons, l'autre essaye de les identifier. Mais le plus intéressant advient quand les deux réseaux travaillent conjointement. Ensemble, ils parviennent à produire des contrefaçons plus impressionnantes non seulement de visages, mais aussi d'objets et de paysages.

Dans la plupart des cas le GAN génère les images de manière aléatoire. Comme contribution de notre travail nous avons amélioré le modèle de telle sorte à pouvoir contrôler des paramètres afin de pouvoir générer des visages selon les types d'attributs de la personne.

Certains de ces paramètres et attributs sont des traits humains tels que le sexe, l'âge, le type d'émotion, les nombreuses poses de la tête, l'origine ethnique, la tonicité de la peau, la couleur des cheveux, le type d'accessoires portés (tels qu'une casquette ou des lunettes de soleil), et bien plus

Encore. En rendant les attributs des personnages ajustables avec le réseau antagoniste génératif, on peut obtenir des meilleurs résultats tout en ayant un contrôle décent sur le type de structures faciales que nous choisissons de générer.

L'emploi de GAN dans l'industrie est un excellent moyen de répondre aux exigences de cette génération. Comme indiqué précédemment, un GAN peut détecter de fausses vidéos profondes, reconnaître des images fausses et réelles et produire de nouvelles images. Une étude plus approfondie pourrait conduire à une évaluation des risques contemporaine, qui pourrait être utilisée à l'avenir. Les idées proposées dans ce mémoire donnent un aperçu rapide du GAN dans l'entreprise et de la manière dont elles pourraient être utilisées à bon escient.

Bibliographie

Bibliographies :

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

[2] Yann LeCun, Yoshua Bengio et Geoffrey Hinton. Deep learning. Nature, vol. 521, pages 436–444, 2015.

http://dx.doi.org/10.1038/nature14539

[3] LeCun, Yann; Léon Bottou; Yoshua Bengio; Patrick Haffner (1998). "Gradient-based learning applied to document recognition"

Proceedings of the IEEE. 86 (11): 2278–23

[4] http://www.utstat.toronto.edu/~rsalakhu/papers/dbm.pdf

[5] http://ufldl.*stanford*.edu/tutorial

[6] Donahue, Jeff et al. (2014). "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition". Proceedings of Machine Learning Research

[7] https://deepmind.com/blog/dee p-reinforcement-learning

[8] Ian Goodfellow, Yoshua Bengio et Aaron Courville. "Deep Learning", MIT Press, 2016. ://www.deeplearningbook.org

[9] Deep learning : quand l'intelligence artificielle tente d'imiter le cerveau humain, Jérôme Cartegini

[10] Perceptron simple, Perceptron Multicouches, Nicolas Rougier

[11] Victor Roman. Supervised Learning: Basics of Classification and Main Algorithms, Jan 31 2019

[12] Concepts de réseau neuronal : Le guide complet des réseaux de neurones artificiels : concepts et modèles

[13] Geoffrey Hinton, Yoshua Bengio et Yann LeCun, Deep Learning NIPS'15 Tutorial, Chapitre 3.

[14 https://blog.octo.com/classification-dimages-les-reseaux-de-neurones-convolutifs-en-toute-simplicite/#:~:text=Les%20r%C3%A9seaux%20de%20neurones%20convolutifs%20sont%20%C3%A0%20ce%20jour%20les,d'une%20matrice%20de%20pixels

[15] https://medium.com/@El_Fares_Anass/a-better-understanding-how-a-cnn-works-code-available-e880f9a338dc

[16] https://medium.com/@zaralwinneirin/understanding-convolutional-neural-network-20893de2a105

[17] http://bib.univ-oeb.dz:8080/jspui/bitstream/123456789/10492/1/m%C3%A9moire%20.pdf

[18] https://www.anaconda.com/download

[19] https://www.tensorflow.org/install/install_windows

[20] (Source : article "Generative Adversarial Nets"

[21] https://developers.google.com/machine-learning/gan/gan_structure

[22] https://developers.google.com/machine-learning/gan/discriminator

[23] https://developers.google.com/machine-learning/gan/generator

[24] https : //tel.archives-ouvertes.fr

[25] Fei-Fei Li, Justin Johnson et Serena Yeung. “ CS231n : Réseaux de neurones convolutifs pour la reconnaissance visuelle. Conférence 13 | Modèles génératifs. ” Université de Stanford (printemps 2017).

[26] http://sh_tsang.medium.com/review-infogon-interpretable -representation-learning-by-information-maximizing-generative-4344bc45d0

[27] https://sh-tsang.medium.com/review-infogan-interpretable-representation-learning-by-information-maximizing-generative-4344b9dc45d0

[28] E. Choi, S. Biswal, B. Malin, J. Duke, W. Stewart et Jimeng Sun. "Génération de dossiers de patients discrets multi- étiquettes à l'aide de réseaux antagonistes génératifs. » MLHC (2017).

Annexes

ANNEXE :

I. Présentation des outils :

I.1 Le hardware :

L'apprentissage profond est un domaine avec de fortes exigences informatiques et une disponibilité des ressources (Surtout dans le GPU) dédié à cette tâche vont fondamentalement influencer sur l'expérience de l'utilisateur car sans ses ressources, apprendre de ses erreurs peut prendre beaucoup de temps.

I.2 Les outils et les bibliothèques utilisés :

I.2.1 Python :

Python est un langage de programmation de haut niveau interprété (aucune étape de compilation) et orienté objet avec une sémantique dynamique. Un grand nombre de développeurs et de programmeurs ont une forte demande. Python est un langage simple, facile à apprendre et peut réduire considérablement les coûts de maintenance du code. Les bibliothèques (packages) Python encouragent la modularité et la réutilisabilité du code. Python et ses bibliothèques sont disponibles gratuitement (en code source ou en fichiers binaires) sur la plupart des plateformes et peuvent être redistribués gratuitement

I.2.2 ANACONDA :

Anaconda est une distribution Python. A son installation, Anaconda installera Python ainsi qu'une multitude de packages. Cela nous évite de nous ruer dans les problèmes d'incompatibilités entre les différents packages.

I.2.2.1 Installation d'Anaconda (la version utilisée) :

A. Téléchargement :

Chercher dans Google « Anaconda Python » ➡ Python V 3.6 pour Windows 64 bits.

B. Installation:

- Installer le fichier d'installation Anaconda 64-bit pour Windows (Install for ➡ option : Just for me)
- Ajouter le chemin d'installation à l'environnement des variables (Rubrique : PATH) pour faciliter l'accès.

C. Lancement et mise à jour d'anaconda :

- Après la fin Del 'installation ouvrir Anaconda Prompt (C:\User\....\Anaconda3).
- Ouvrez Anaconda Prompt en mode ADMINISTRATEUR. Ceci vous évitera des erreurs de droits en écritures par la suite lors de l'installation des bibliothèques / paquets. (Anaconda prompt ➡ bouton droit ➡ menu ➡ mode administrateur ➡C:\WINDOWS\system32>).
- Modifier les package d'anaconda en tapant les deux commandes :

  >conda update conda.

  >conda update –all.

D. Installation des accessoires nécessaires de mon travail :

- C:\>conda install jupyter ➡ jupyter 1.0.
- C:\>conda install scipy ➡ la bibliothèque des tableau Scipy.
- C:\>pip install sklearn ➡ la bibliothèque de machine learning.
- C:\>pip install pandas ➡ la bibliothèque de machine learning.
- C:\>pip install pandas-datareader ➡ outils de lecture de bibliothèque ML pandas.
- C:\>pip install matplolib ➡ installer l'outil de traçage de courbes.
- C:\>pip install pillow ➡Outils complémentaire de ML.
- C:\>pip install requests ➡ Outils complémentaire de ML.
- C:\>pip install h5py ➡ Outils complémentaire de ML.
- C:\>pip install tensorflow/tensorflow ==1.2.1 ➡Installer la bibliothèque de machine learning et l'interface google tensorflow.
- C:\>pip install keras/keras==2.0.6 ➡ Installer la bibliothèque de machine learning et l'interface keras.

I.2.3 TensorFlow :

Nous avons utilisé dans notre travail TensorFlow qui 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, comme le montrent les dernières comparaisons suivantes, faites par François Chollet (auteur de la librairie Keras).

Bien qu'il ait initialement été développé pour optimiser les calculs numériques complexes, TensorFlow est aujourd'hui particulièrement utilisé 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, etc.

I.2.3.1 L'environnement tensorflow d'après anaconda :

1. Créer l'environnement de Tensorflow ➡ >conda create -n tensorflow.

2. Activer l'environnement de Tensorflow créée ➡ > conda activate tensorflow

I.2.4 Keras :

Keras est une API de réseau neuronal avancée écrite en Python qui peut s'exécuter sur TensorFlow ou Theano. Son développement se concentre sur l'expérimentation rapide. Être capable de passer de l'idée au résultat en un minimum

De temps est la clé d'une bonne recherche. Il fait partie des travaux de recherche du projet ONEIROS (Open Neuroelectronic Intelligent Robot Operating System) dont le principal auteur et mainteneur est l'ingénieur Google François Chollet

I.2.5 Quelque Bibliothèques :

1. Numpy : est une bibliothèque Python très populaire pour le traitement de

Matrices et de tableaux multidimensionnels, à l'aide d'une vaste collection de fonctions mathématiques de haut niveau. C'est très utile pour les calculs scientifiques fondamentaux en Machine Learning. Il est particulièrement utile pour l'algèbre linéaire, la transformée de Fourier et les capacités de nombres

Aléatoires. Les bibliothèques haut de gamme telles que TensorFlow utilisent Numpy en interne pour la manipulation de Tensors

2. Scipy : est une bibliothèque très populaire parmi les passionnés d'apprentissage automatique car elle contient différents modules d'optimisation, d'algèbre linéaire, d'intégration et de statistiques. Il existe une différence entre la bibliothèque Scipy et la pile Scipy.

Le SciPy est l'un des principaux packages qui composent la pile Scipy. Scipy est également très utile pour la manipulation d'images

3. Scikit-learn : est l'une des bibliothèques ML les plus populaires pour les algorithmes ML classiques. Il est construit sur deux bibliothèques Python de base, à savoir, Numpy et Scipy. Scikit-learn supporte la plupart des algorithmes d'apprentissage supervisé et non supervisé. Scikit-learn peut également être utilisé pour l'exploration et l'analyse de données, ce qui en fait un excellent outil pour les débutants en ML

4. Pandas : est une bibliothèque Python populaire pour l'analyse de données. Ce n'est pas directement lié à l'apprentissage automatique. Comme nous savons que le jeu de données doit être préparé avant la formation. Dans ce cas, les pandas sont pratiques car ils ont été développés spécifiquement pour l'extraction et la préparation de données. Il fournit des structures de données de haut niveau et de nombreux outils pour l'analyse des données. Il fournit de nombreuses méthodes intégrées pour tâtonner, combiner et filtrer les données

5.Matplotlib : est une bibliothèque Python très populaire pour la visualisation de données. Comme les pandas, il n'est pas directement lié à l'apprentissage automatique. Cela s'avère particulièrement utile lorsqu'un programmeur souhaite visualiser les modèles dans les données. C'est une bibliothèque de tracé 2D utilisée pour créer des graphiques et des tracés 2D. Un module appelé pyplot facilite le traçage des programmeurs car il offre des fonctionnalités permettant de contrôler les styles de trait, les propriétés de police, les axes de formatage, etc. Il fournit différents types de graphiques et de tracés pour la visualisation des données, l'affichage, l'histogramme, les graphiques d'erreur, les discussions en barres, etc.

6.Pillow : est une bibliothèque de traitement d'image, qui est un fork et successeur du projet PIL (Python Imaging Library). Elle est conçue de manière à offrir un accès rapide aux données contenues dans une image, et offre un support pour différents formats de fichiers tels que PPM, PNG, JPEG, GIF, TIFF et BMP.

Résumé :

L'apprentissage automatique est un domaine en pleine croissance qui révolutionne les programmes informatiques en offrant aux systèmes la capacité d'apprendre et de s'améliorer automatiquement à partir de l'expérience. Dans la plupart des cas, le processus de formation commence par l'extraction des échantillons à partir de données. Les données sont un facteur clé pour les algorithmes d'apprentissage automatique, sans données, les algorithmes ne fonctionneront pas. L'idée est d'utiliser des réseaux antagonistes génératifs pour générer des images synthétiques similaires à la vérité sur terrain, et ainsi élargir un ensemble de données.

Les algorithmes de cette intelligence artificielle appelée ont d'abord été entraînés à partir d'une base de données contenant des photos de personnalités. Deux réseaux neuronaux en compétition qui vont alors se former l'un l'autre (Générative Adversial Networks).

Un réseau créé des contrefaçons, l'autre essaye de les identifier. Mais le plus intéressant advient quand les deux réseaux travaillent conjointement. Ensemble, ils parviennent à produire des contrefaçons plus impressionnantes non seulement de visages, mais aussi des chiffres.

Certains des paramètres et attributs qui peuvent être ajustables sont des traits humains tels que le sexe, l'âge, le type d'émotion, les nombreuses poses de la tête, l'origine ethnique, la tonicité de la peau, la couleur des cheveux, le type d'accessoires portés (tels qu'une casquette ou des lunettes de soleil), et bien plus encore. En rendant tous ces personnages variables avec le réseau antagoniste génératif que vous construisez, de nombreuses variations sont possibles.

Mots-clés : générateur, discriminateur, classification, apprentissage profond, réseau génératif contradictoire, apprentissage automatique

Abstract:

Machine learning is a growing field that is revolutionizing computer programs by providing systems with the ability to automatically learn and improve from experience. In most cases, the training process starts with extracting samples from data. Data is a key factor for machine learning algorithms, without data the algorithms will not work. The idea is to use generative adversarial networks to generate synthetic images similar to ground truth, and thus expand a data set.

The algorithms of this artificial intelligence called were first trained from a database containing photos of personalities. Two competing neural networks that will then train each other (Generative Adversial Networks).

One network creates counterfeits, the other tries to identify them. But the most interesting thing happens when the two networks work together. Together they manage to produce more impressive fakes of not only faces, but also numbers.

Some of the parameters and attributes that can be adjustable are human traits such as gender, age, emotion type, many head poses, ethnicity, skin tone, hair color, the type of accessories worn (such as a cap or sunglasses), and much more. By making all of these characters variable with the generative adversarial network you build, many variations are possible.

Keywords: generator, discriminator, classification, deep learning, adversarial generative network, machine learning

ملخص:

يعد التعلم الآلي مجالًا متناميًا يُحدث ثورة في برامج الكمبيوتر من خلال تزويد الأنظمة بالقدرة على التعلم والتحسين تلقائيًا من التجربة. في معظم الحالات، تبدأ عملية التدريب باستخراج عينات من البيانات. البيانات هي عامل رئيسي لخوارزميات التعلم الآلي، وبدون البيانات لن تعمل الخوارزميات. الفكرة هي استخدام شبكات الخصومة التوليدية لإنشاء صور تركيبية مشابهة للحقيقة الأساسية، وبالتالي توسيع مجموعة البيانات.

تم تدريب خوارزميات هذا الذكاء الاصطناعي التي تسمى أولاً من قاعدة بيانات تحتوي على صور الشخصيات. شبكتان عصبيتان متنافستان ستشكلان بعضهما البعض (شبكات الخصومة التوليدية). تقوم إحدى الشبكات بإنشاء عمليات مزيفة، بينما تحاول الأخرى التعرف عليها. لكن الشيء الأكثر إثارة للاهتمام يحدث عندما تعمل الشبكتان معًا. تمكنوا معًا من إنتاج صور مزيفة أكثر إثارة للإعجاب ليس فقط الوجوه، ولكن أيضًا للأرقام.

بعض المعلمات والسمات التي يمكن تعديلها هي السمات البشرية مثل الجنس والعمر ونوع العاطفة والعديد من أوضاع الرأس والعرق ولون البشرة ولون الشعر ونوع الملحقات التي يتم ارتداؤها (مثل القبعة أو النظارات الشمسية)، وغير ذلك الكثير. أكثر. من خلال جعل كل هذه الأحرف متغيرة باستخدام شبكة الخصومة التوليدية التي تنشئها، يمكن إجراء العديد من الاختلافات.

الكلمات المفتاحية: **مُولِّد، مُميِّز، تصنيف، تعلُّم عميق، شبكة توليد خصومة، تعلُّم آلي**