République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mouloud Mammeri de Tizi-Ouzou

Faculté de : Génie électrique et d'informatique
Département : Informatique

Mémoire de fin d'études

En vue de l'obtention du diplôme de
Master en Informatique

Spécialité : Conduite de projets informatiques

Présenté par :
Tayeb M'hamed et **Saoudi Yasmina**

Thème :

Etude comparative de solutions de deep learning de classification d'images médicales

Soutenu le 15 juillet 2019 devant le jury composé de:

Mr. HAMMACHE Arezki	Président du Jury
Mlle. YESLI Yasmine	Membre du Jury
Mme. BERKANE Tassadit	Membre du Jury
Mme. BOUARAB Farida	Encadreur

Remerciements

La mise en oeuvre de ce mémoire est bien plus que la concrétisation d’efforts personnels mais le résultats d’efforts multiples que nous ne voulons pas occulter; ainsi ,nous tenons à exprimer notre gratitude envers notre encadreur le professeur Bouarab dont les conseilles avisés ont éclairés le chemin qui nous a mener à la concrétisation de ce travail , mais aussi à nos familles respectives et à nos proches qui ont su être présents et apporter un soutien indéfectible tout au long de notre parcours académique rendant ainsi possible la réalisation de ce projet.

Sommaire

Chapitre II Conception des différentes approches employées

Chapitre III Réalisation

Table des figures

Introduction générale

A l'ère des premières voiture autonomes [i1], des assistants électroniques personnels ,du déploiement du premier programme intelligent spécialisé dans la détection du cancer du sein [i2] , à l'aube de l'artiste artificiel , du smart-gérant et du médecin de poche ; il ne fait aucun doute qu'aujourd'hui, plus que jamais, l'intelligence artificielle c'est invité dans nos vies .

Que ce soit nos téléphones capables de reconnaître nos visages , de mettre nos paroles à l'écrit ou encore de tenir des conversations cohérentes avec nous ou simplement le moteur de recherche que nous utilisons tous les jours qui apprend à nous suggérer du contenu selon nos préférences ; cette dernière s'est rendue indispensable ,et est à l'origine des plus grand bouleversements socio-économique de notre temps et des temps à venir .

Au coeur de cette révolution un nom résonne celui du *Deep Learning* ou apprentissage profond.S'illustrant entre autres dans le domaine médicale en permettant pour la première fois l'automatisation de certaines tâches jusqu'alors inaccessible aux machines .
Fascinés par les exploits de cette technologie nous nous sommes plongé dans ce vaste domaine

L'apprentissage profond c'est illustré au cours de la dernière décennie en monopolisant l'attention des géants du monde informatique (que ce soit la création de plusieurs équipes de recherche spécialisées dans le domaine par Google . la politique de recrutement agressive des professeurs menée par tesla et SpaceX ou même les investissement pharaoniques (plusieurs dizaines de milliards de dollars)investies par samsung ou encore le passage de cette technologie en priorité d'état en Chine , en Finlande ou encore en France) et bien évidemment l'ensemble de la communauté scientifique (via les milliers de thèses de recherche sur le sujet , les concours à l'instar de ChestX de Stanford et les collaborations).

Les etats unis sont les premiers à avoir voter des lois permettant d'employer des outils issues d'intelligences artificielles dans le dépistage et l'aide au diagnostic ;ouvrant alors un nouveaux champs applicatif à une technologie déjà bien installée; c'est ainsi , qu'en Mai de cette années , Google a annoncé la création d'une branche spécialisée dans le diagnostique médicale automatisé tout en annoncé leur premier modèle capable de détecter les cancer du poumon (l'un des cancers les plus foudroyant et les plus communs) [i3] .

En parallèle l'on a vu naître un nombre impressionnant d'initiatives visant à simplifier et à encourager d'emplois d'intelligences artificielles dans le domaine médicale afin de réduire les coûts liés à l'accès aux soins permettant au plus pauvres et au plus isolées d'avoire une couverture médicale acceptable
the national health Système de grande bretagne s'oriente déjà vers la possibilitée d'inclure des outils issues des intelligences artificielles estimant que l'emplois de celles-ci pourrait à terme faire économiser au pays jusqu à 150 milliards de dollars d'ici l'horizon 2026 [i4] .

De remarquables efforts d'automatisation ont été déployés dans le domaine de l'exploration médicale notamment celui de la radiologie comme en atteste un certain nombre d'articles traitant de l'éthique de l'emplois de ce type de technologies [i5] ou l'on a vu naître un nombre impressionnant d'articles et d'essai et de prototype voir d'application .

Et au coeur de cette effervescence plusieurs architectures et méthodes propres au *deep learning* ont su s'accaparer l'attention du plus grand nombre ;parmis elles, l'architecture la plus courante basée sur l'emplois de réseaux de neurones convolutifs la plus très récente architecture basée sur l'emplois de capsules et plus récemment encore une technique révolutionnaire basé sur le transfert de compétences dite de *transfer learning.*

Problématique

Fort de ce constats nous nous sommes donné comme objectif d'employer les technique du deep learning afin de concevoir plusieurs solutions, basées respectivement sur chacune des techniques introduite précédemment ,capables de classer des images médicales de type téléthorax afin de dépister des cas de pneumonie puis de réaliser une étude comparatives entre chacune d'entre elles afin de déterminer laquelle ,des trois solutions considéré dans le cadre de ce mémoire , est la plus efficace.

Plan de travail

Nous procéderons en explorant chacune des architectures précédemment introduite avant d'implémenter ces dernières nous les évaluerons ensuite à l'aide de mesures préétablie faisant office de canon pour ce type de problématique avant de conclure en analysant les résultats obtenus.

Dans les prochaines pages nous tâcherons de répondre à bien plus que ces questions en nous intéressant dans un premiers temps à ce domaine à la frontière entre le monde des intelligences artificielles et celui du data mining , nous verrons ensuite les possibilitées qu'il offre une foi appliqué au domaine médicale nous verrons plus précisément comment l'employer dans le cadre de la classification d'images médicale de type radio du torse ou téléthoraxe; nous aborderons plusieurs approches que nous évaluons afin de les faire se confronter dans le but de désigner la plus efficace à l'heure ou l'on écrit ces lignes , en effet il est important de souligner l'évolution fulgurante de ce domaine et la volatilisées des lois n et canons qui le définissent .Nous conclurons notre mémoire par un rétrospective de notre travail suivie de l'énumération de quelques perspectives .

[i1] pilote automatique de tesla https://www.tesla.com/autopilot véhicules autonomes de Google https://waymo.com/

[i2] plateforme d'aide au diagnostique du cancer du sein https://www.analyticsinsight.net/artificial-intelligence-aids-breast-cancer-detection-and-saves-time/

[i3] division medicale de Google https://ai.google/healthcare/

[i4] initiative e-santé en GB https://www.ehidc.org/resources/artificial-intelligence-healthcare

[i5] rapport sur l'ethique et limites judiciaire de l'emplois du DL dans le domaine medicale en europe

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6206380/

Chapitre I Mise en contexte

Dans ce chapitre nous introduisons dans un premier temps le concept de réseaux de neurones puis celui de l'apprentissage profond pour enfin nous focaliser sur l'emplois de ce derniers dans le domaine médicale plus précisément dans la cas de la pathologie que nous traitons la pneumonie.

I .1 - Introduction

Dans ce qui suit nous aborderons le sujet des neurones biologiques dont les mécaniques de fonctionnement ont inspiré nos prédécesseur dans la conception de leurs équivalent mathématique puis algorithmique , nous verrons ensuite les bases des réseaux de neurones en commençant par la brique de base de ces derniers les perceptrons puis en nous attardant plus en détaille sur les procédés permettant à ces réseaux “d’apprendre “ avant d’enfin revenir sur la naissance du Deep Learning à proprement parler .

Mais avant d’aborder plus amplement ce thème il nous faut rappeler un ensemble d’informations que nous estimons vitale et plus que nécessaire à la compréhension de ce qui suit.

I . 1 . 1 Le domaine des Intelligences artificielles

Il convient afin d’assimiler les information distillées par notre écrit de se recontextualiser dans un premiers temps , en situant le paradigme auquel appartient la discipline dans laquelle nous évoluons ;
pour mieux nous repérer il est important de comprendre que le deep learning est une branche du machine learning qui lui même est un domaine issue du vaste monde des intelligences artificielles et qui tend à verser parfois dans le domaine de la data science ou bon nombre de techniques de ce derniers sont communes aux deux mondes (voir figure A)

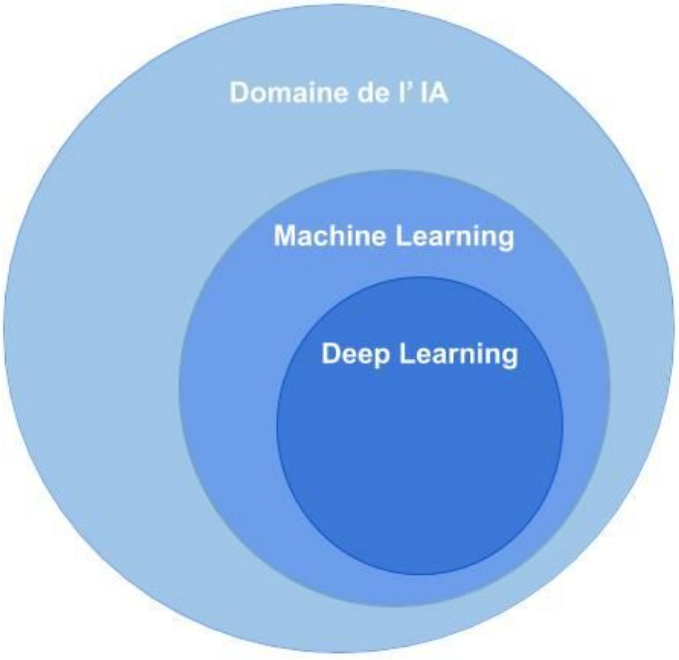

Figure A schématisation du rapport entre IA , ML et DL [f1] .

Nous rappelons de plus qu'il existes plusieurs paradigmes (ensemble de concepts,shéma de pensées ,théories et des méthodes d'évaluations s'apparentant à une philosophie ou vision d'un domaine) en informatique nous nous inscrivons dans celui du *deep learning* qui diffère de ces contemporains par plusieurs aspects résumé par ceci :

	Deep Learning	**programmation classique**
entrée	data sets	données
exécution	réseaux de neurones	algorithmes
sorties	Modèle	programme

I . 1 .2 Qu'est ce qu'un modèle

Enfin , le concept le plus important que nous puissions introduire reste le modèle tel qu'indiqué plus le haut il est la finalité de notre travail et bien que sa définition varie fortement d'une source à l'autre notamment du fait qu'il puisse être abordé par plusieurs angles il reste possible de le voir soit comme un modèle de distribution de probabilité (indiquant la probabilité d'une sortie pour une entrée donnée), ou encore comme le réseau de neurone implémenter une foi correctement pondéré et enfin comme une fonction mathématique associant une entré à une sortie.

Conséquemment , et selon les situations le sens qui sera prêté à ce terme pourrait varié , et de part son emplois par fois et par abus de langage pour désigner le réseaux de neurone duquel il résulte , une certaine confusion peut naître face à certaines formulation ainsi d'avance nous nous excusons pour ce désagrément.

I . 2 - Le neurone

I .2.1 - Le neurone biologique

Le neurone est l'unité fondamentale du cerveau humain. Un petit morceau de cerveau de la taille d'un grain de riz, contient plus de 10 000 neurones, dont chacun forme 6 000 connexions en moyenne avec d'autres neurones[1]. C'est cet énorme réseau biologique qui nous permet de faire l'expérience du monde qui nous entoure. Notre objectif dans cette section sera d'utiliser cette structure naturelle pour construire des modèles d'apprentissage automatique résolvant des problèmes manière analogue.

A la base, le neurone est optimisé pour recevoir des informations d'autres neurones, traiter cette information d'une manière unique, et envoyer son résultat à d'autres cellules. Ce processus est résumées à la figure I.1. Le neurone reçoit ses entrées le long de structures analogues à des antennes appelé dendrites. Chacune de ces connexions entrantes est dynamiquement renforcée ou affaibli en fonction de la fréquence d'utilisation,et c'est la force de chaque connexion qui détermine la contribution de l'entrée à la sortie du neurone. Après avoir été pondéré par la force de leurs connexions respectives, les entrées sont additionnées dans le corps de la cellule. Cette somme est ensuite transformée en un nouveau signal qui se propage le long de l'axone de la cellule et envoyé à d'autres neurones [C1].

[C1] , Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithm
https://books.google.dz

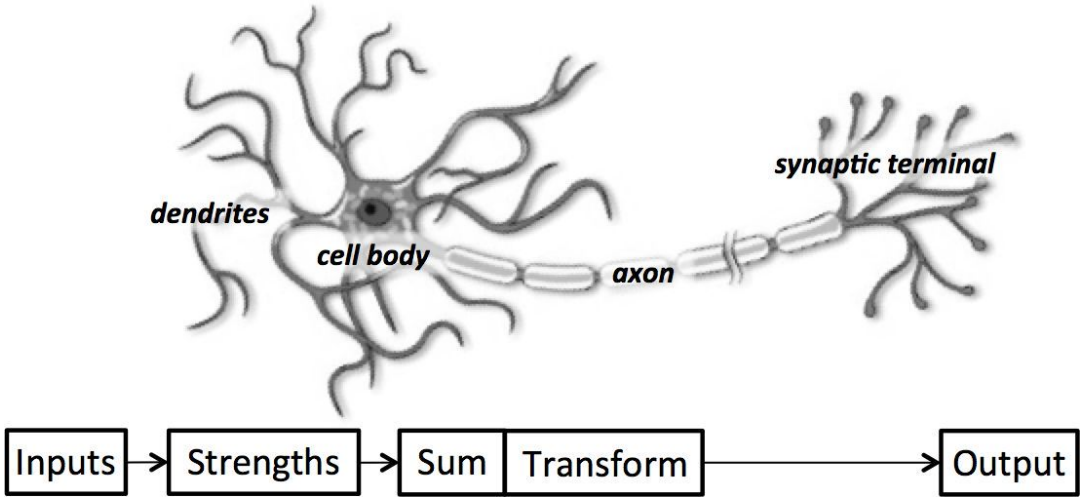

Figure I.1: Description fonctionnelle de la structure d'un neurone biologique [f2] .

les neurones sont des cellules nerveuses décomposées en 4 parties:

- Les dendrites
  c'est par eux que se fait la réception des signaux
- L'axone
  c'est la partie où passent les messages accumulées dans le corp de la cellule.
- Le corp de la cellule
  ou le noyau c'est l'unité de traitement
- Les synapses
  sont des point de connexion par où passent les signaux de la cellule.

I . 2.2 -Le neurone artificiel:

Les réseaux de neurones biologiques réalisent facilement un certain nombre d'applications telles que la reconnaissance de formes, le traitement de signal, l'apprentissage par l'exemple, la mémorisation, la généralisation. Ces applications sont pourtant, malgré tous les efforts déployés en algorithmique et en intelligence artificielle, à la limite des possibilités actuelles. C'est à partir de l'hypothèse que le comportement intelligent émerge de la structure et du comportement des éléments de base du cerveau que les réseaux de neurones artificiels se sont développés. Les réseaux de neurones artificiels sont des modèles, à ce titre ils peuvent être décrit par leurs composants, leurs variables descriptives et les interactions entre leurs composants[C2] .

I . 2.2.1 La Structure d'un neurone artificiel:

La figure I.2 montre la structure d'un neurone artificiel. Chaque neurone artificiel est un processeur élémentaire. Il reçoit un nombre variable d'entrées en provenance de neurones en amonts et à chacunes de ces entrées est associée un poids **w** abréviation de **weight** qui représente la force de la connexion[C2]. Chaque processeur élémentaire est doté d'une **sortie unique**, qui se ramifie ensuite pour alimenter un nombre variable de neurones avals. A chaque connexion est associée un poids.

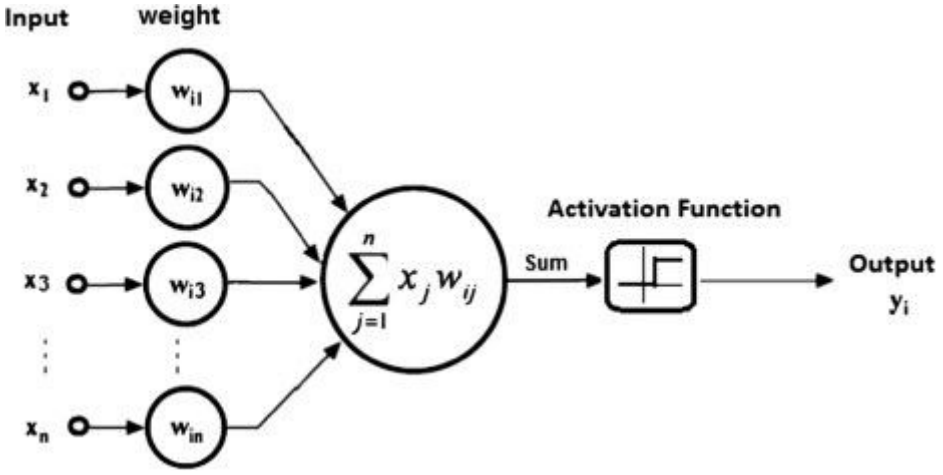

Figure I.2: Structure d'un neurone artificiel [f3] .

I .3 -Réseau de neurones artificiels :

Il s'agit du produit de l'interconnexion d'un grand nombre de neurones s'organisent généralement en couches superposées ou l'output des uns devient l'input des autres ;bien qu'il existe bon nombre d'autres topologies permettant de structurer ce type de réseaux; (dans le cadre de ce mémoire nous nous focaliserons sur la topologie en couches appelée aussi multicouches).

Les traitements que ce groupe d'unités parvient alors à accomplir se complexifie grandement d'autant plus si la fonction d'activation choisie est non linéaire . Le processus visant à raffiner les parametre de ce nouvel ensemble de neurones afin d'obtenir des sorties

particulières selon les entrées présentées à ces derniers est appelé entraînement et se traduit par la capacité à terme d'inculquer un comportement (se traduisant par le paramétrage d'une fonction pondérée) ce procédé ainsi décrit est à la base de l'apprentissage et nous verrons dans ce qui suit les principes fondamentaux de ce procédé.

[C2] : Claude Touzet , LES RESEAUX DE NEURONES ARTIFICIELS INTRODUCTION AU CONNEXIONNISME
http://www.touzet.org/Claude/Web-Fac-Claude/Les_reseaux_de_neurones_artificiels.pdf

I . 4 - Historique:

- **1890** : W. James, célèbre psychologue américain introduit le concept de mémoire associative, et propose ce qui deviendra une loi de fonctionnement pour l'apprentissage sur les réseaux de neurones connue plus tard sous le nom de **loi de Hebb**.

- **1943 :** J. Mc Culloch et W. Pitts laissent leurs noms à une modélisation du neurone biologique (un neurone au comportement binaire). Ceux sont les premiers à montrer que des réseaux de neurones formels simples peuvent réaliser des fonctions logiques, arithmétiques et symboliques complexes (tout au moins au niveau théorique).

- **1949** : D. Hebb, physiologiste américain explique le conditionnement chez l'animal par les propriétés des neurones eux-mêmes. Ainsi, un conditionnement de type pavlovien tel que, nourrir tous les jours à la même heure un chien, entraîne chez cet animal la sécrétion de salive à 7 cette heure précise même en l'absence de nourriture. La loi de modification des propriétés des connexions entre neurones qu'il propose explique en partie ce type de résultats expérimentaux.

- **1957** : F. Rosenblatt développe le modèle du Perceptron. Il construit le premier neuro ordinateur basé sur ce modèle et l'applique au domaine de la reconnaissance de formes. Notons qu'à cet époque les moyens à sa disposition sont limités et c'est une prouesse technologique que de réussir à faire fonctionner correctement cette machine plus de quelques minutes.

- **1960** : B. Widrow, un automaticien, développe le modèle Adaline (Adaptive Linear Element). Dans sa structure, le modèle ressemble au Perceptron, cependant la loi d'apprentissage est différente. Celle-ci est à l'origine de l'algorithme de rétropropagation de gradient très utilisé aujourd'hui avec les Perceptrons multicouches. Les réseaux de type Adaline restent utilisés de nos jours pour certaines applications particulières. B. Widrow a créé dès cette époque une des premières firmes proposant neuro-ordinateurs et neuro-composants, la "Memistor Corporation". Il est aujourd'hui le président de l'International Neural Network Society (INNS) sur laquelle nous reviendrons au chapitre Informations pratiques.

- **1969** : M. Minsky et S. Papert publient un ouvrage qui met en exergue les limitations théoriques du perceptron. Limitations alors connues, notamment concernant l'impossibilité de traiter par ce modèle des problèmes non linéaires. Ils étendent implicitement ces limitations à tous modèles de réseaux de neurones artificiels. Leur objectif est atteint, il y a

abandon financier des recherches dans le domaine (surtout aux U.S.A.), les chercheurs se tournent principalement vers l'IA et les systèmes à bases de règles.

- **1974 - 89** : Naissance de la première théorie stipulant qu'il pouvait être implémenter dans les réseaux de neurone en **1974** par P.Werbos l'algorithme de la ***Backpropagation*** ne sera réellement implémenté et capable de résultats satisfaisant qu'en **1989** suite au travaux de G.Hinton , D.Rumelhart et R.Williams ayant permis de l'optimiser et permettant depuis de ressusciter l'intérêt que portait la communauté scientifique à l'égards des réseaux de neurones.
- **1989 - 98** : Apparition de la première implémentation de l'architecture de réseaux de neurones convolutif **CNN** . Création de la base de données **MNIST**(Modified National Institute Of Standards and Technologies), naissance des architectures **LSTM** et **bidirectional RNN**
- **2009** Création de l'immense base de données d'images accessibles via internet **ImagNet**
- **2012** Implémentation d'une version altéré des CNN optimisées pour l'exécution sur GPU par A.Krizhevsky **AlexNet.** la même année mise au point de la technique du ***DroupOut*** visant à intégrer une couche dégradant aléatoirement les données lors de l'apprentissage afin de réduire *l'overfitting*

- **2014** Naissance de l'architecture GANs (Générative Adversarial Networks) introduits par J.Goodfellow permettant de générer des images lançant l'ère des IAs artistes

- **2016** Implémentation de la première IA capable de battre un champion en titre au jeu de Go réputé plus complexe encore que les échec (le nombre de configuration de jeu possible aux échec est de 10^{120} la avec le GO il en existe 10^{174}) ***AlphaGO*** par l'équipe de DeepMind

- **2017-18** Implémentation de l'architecture **Capsule Networks** proposée par G.Hinton, S.Sabour et N.Frosst reposant sur l'emplois de capsule travaillant en groupes afin de valider la détection de caractéristiques sur une image

I . 5 -Topologie de réseaux de neurones:

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

- **Réseau multicouche** :

Les neurones sont arrangés par couche. Chaque neurone d'une couche est connecté à tous les neurones de la couche suivante et celle-ci seulement les connexions se font qu'avec les neurones des couches avales (fig I.3) [C1]. Il n'y a pas de connexion entre les neurones d'une même couche . Ceci nous permet d'introduire la notion de sens de parcours de l'information (de l'activation) au sein d'un réseau et donc définir les concepts de neurone d'entrée, neurone de sortie. Par extension, on appelle couche d'entrée l'ensemble des neurones d'entrée, couche de sortie l'ensemble des neurones de sortie. Les couches intermédiaires n'ayant aucun contact avec l'extérieur sont appelés couches cachées.

Notez qu'il n'existe pas de méthode mathématique permettant d'estimer le nombre de couches cachées à employer dans un réseau de neurones (cas des réseaux profond du deep learning), il s'agit alors d'estimer ce nombre à l'aide d'expériences ou encore en se basant sur une expertise personnelle

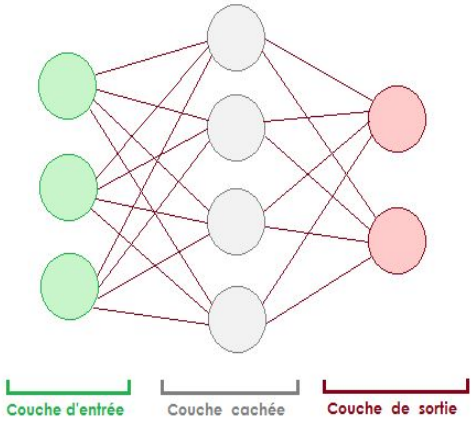

Figure I.3 : Réseau multicouche [f4]

- **Réseau à connexions locales :**

Il s'agit d'une structure multicouche, mais qui à l'image de la rétine, conserve une certaine topologie. Chaque neurone entretient des relations avec un nombre réduit et localisé de neurones de la couche avale (figure *I*.4) [C2]. Les connexions sont donc moins nombreuses que dans le cas d'un réseau multicouche classique.

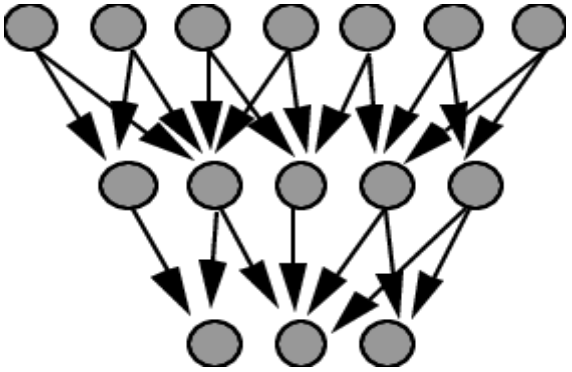

Figure I.4: Réseau à connexion locales [f4]

- **Réseau à connexions récurrentes :**

Les connexions récurrentes ramènent l'information en arrière par rapport au sens de propagation défini dans un réseau multicouche [C2]. Ces connexions sont le plus souvent locales (figure I.5).

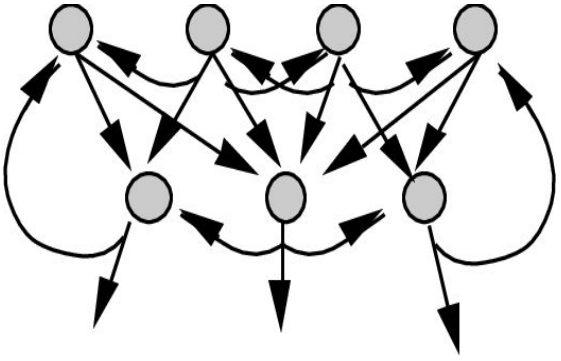

Figure I.5: Réseau à connexion récurrentes [f4]

- **Réseau à connexion complète :**

C'est la structure d'interconnexion la plus générale (figure I.6). Chaque neurone est connecté à tous les neurones du réseau (et à lui-même). [C2]

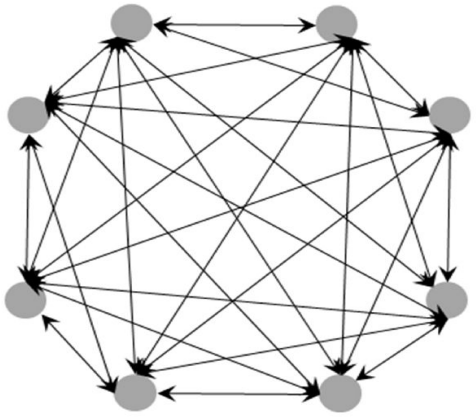

Figure I.6: Réseau à connexion complète [f4]

Il existe de nombreuse autres topologies possibles, mais elles n'ont pas eu à ce jour la notoriété des quelques unes que nous avons décrites ici.

I . 6 - Fonctionnement d'un réseau de neurones

I .6.1- Perceptron

Avant d'aborder les structures complexes des réseaux de neurones nous nous attarderons sur l'unité fondamentale de ces derniers le Perceptron (neurone unique) .

Il s'agit d'un réseau de neurone à une seule couche inventé par Rosenblatt il est formée d'une première couche d'unités (ou neurones) qui permettent de lire les données chaque unité correspond à une des variables d'entrée. On peut rajouter une unité de biais qui est toujours activée (elle transmet 1 quelles que soient les données) [C3]. Ces unités sont reliées à une seule et unique unité de sortie, qui reçoit la somme des unités qui lui sont reliées, pondérée par des poids de connexion le résultat en sortie est alors soumis à une fonction dite d'activation qui selon la valeur en entrée renverra une valeur indiquant si le neurone à été "activé " .Nous reviendrons plus en détaille sur les types de fonctions d'activations dans le point suivant .

[C3] :Chloé-Agathe Azencott, Architecture d'un perceptron, https://bit.ly/30tQby6

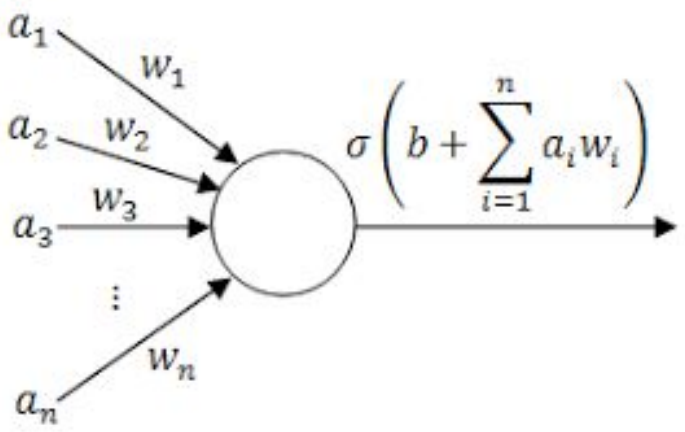

Figure I.7: le fonctionnement d'un perceptron [f4]

I . 6 .2 -Fonctionnement mathématique

Comme précédemment introduit le fonctionnement interne d'un perceptrons peut être résumé par la formule suivante

$$Sortie = \sigma\left(b+\sum_{i=1}^{n} \alpha_i w_i\right)$$

où :

- σ Est une fonction d'activation (Step Function) qui prend en entré le résultat de la somme des entrées pondérées du perceptron et du biais et renvois en sortie un valeur dit d'activation
- b Il s'agit d'un biai càd une constante de correction utilisée afin d'affiner les résultats en sortie du neurone
- a_i (i allant de 1 à n) Correspond à l'ensemble des entrées (inputs) perçus par le perceptron celles-ci pouvant être le résultat en sortie d'autres neurones ou des données brute perçues en entré
- w_i (i allant de 1 à n) Est l'ensemble des poids associées aux entrées , ou chaque entrée (input) se voit associé un poid unique

I .6.3- Fonctions d'activation

Comme vu précédemment le résultat de la somme des inputs pondérées et du biai sont soumis en entré à une fonction d'activation dont le rôle est d'associer en sortie du neurone un état correspondant soit à l'état "neurone activé" ou "neurone non activé" selon les entrées qu'elle reçoit , selon le type de fonction d'activation ses états se traduiront par des valeurs appartenant à un intervalle particuliers ([-1,1] pour la fonction tanh ou $[0, +\infty]$ pour ReLu).

Ce type de fonctions introduisent dans réseaux de neurones de propriétées dites non-linéaires (en introduisant des courbes)afin de permettre aux réseaux de neurone de simuler au mieux les fonctions mathématiques les plus complexes car la nature même de ces derniers est d'être des outils universels d'approximation de fonctions .

Il est à noter que chez les expert du domaine les fonction d'activation les plus populaire sont la fonction Sigmoid , la fonction de tangente hyperbolique (Tanh) et la fonction d'unités linéaires rectifiées (ReLu) qui se distinguent les unes des autres comme suit:

- La fonction Sigmoid

  définie par la formule suivante elle associe à tout réel x une valeur appartenant à l'intervalle]0,1[(voir figure I.8)

$$S(x) = \frac{1}{1+exp(-x)}$$

c'est l'une des premières fonction utilisé par les experts du domaine qui offre en plus l'avantage de normaliser les entrées qu'elle reçoit mais qui depuis a beaucoup perdu en popularité due aux problèmes suivants :

- L'incapacité de la fonction Sigmoid d'associer la valeur 0 à une entrée empêchant ainsi d'éliminer des inputs lors du calcule des sorties des couches inférieures de neurones ce qui alourdit les calcules

- L'inefficacité de l'algorithme de descente de gradient lorsqu'il est associé à cette fonction notamment à cause de la saturation qu'il finit par provoquer (inexistence de la valeur 0)

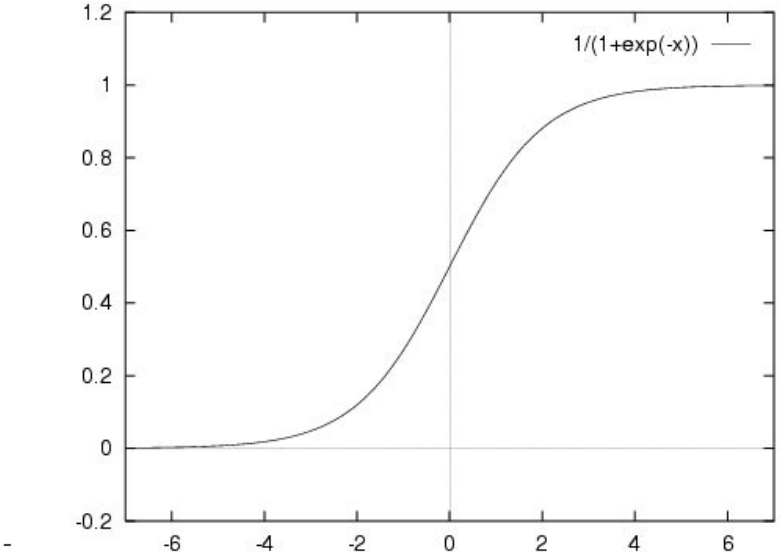

Figure I.8 courbe la fonction sigmoid [f5]

- La fonction tangente hyperbolique (tanh)
  elle est définie par la formule ci-bas , elle associe à tout réel x une valeure entre]-1 , 1[représenté ci-contre (voire figure I.9)

$$Tanh = \frac{1 - exp(-2x)}{1 + exp(-2x)}$$

Cette fonction corrige la faille la plus grave de la fonction sigmoid à savoir sa faible compatibilité avec l'algorithme de la descente de gradient à la base du processus d'apprentissage cependant l'emplois de cette fonction d'activation ne suffit toujours à se prémunir du problème du *Vanishing gradient problème* (affaiblissement de gradient) qui consiste en l'affaiblissement des changements apportées aux poids par l'algorithme de descente de gradient lors de l'entraînement qui se traduit par le fait que le réseau de neurone n'apprenne plus

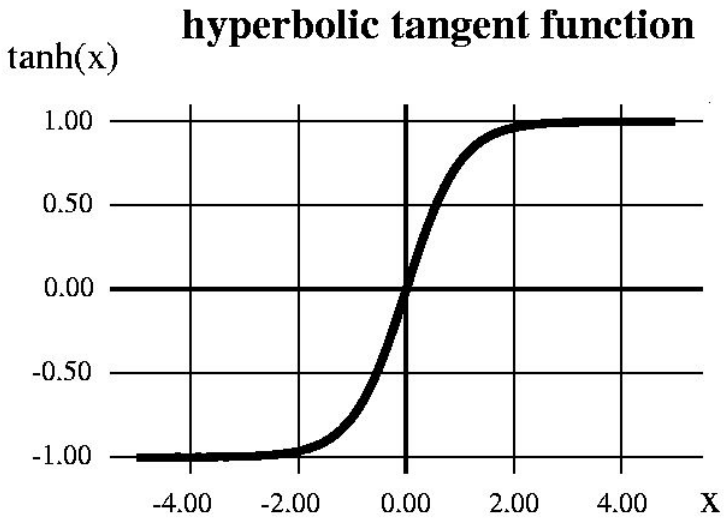

Figure I.9 courbe de la fonction de tangente hyperbolique [f5]

- La fonction d'unités linéaire rectifiées (ReLu)

  Sans doute la plus utilisée de nos jours ayant une meilleure convergence que tanh elle se définie comme suit

$$ReLu(x) = Max\,(0\ ,\ X)$$

C'est à dire

$$Si\ x\ <\ 0\ alors\ ReLu(x)\ =\ 0$$

$$Et\ Si\ x\ >\ 0\ alors\ ReLu(x)\ =\ x$$

Il est à noter que bien que d'apparence simple cette fonction permet de résoudre plus efficacement que les précédentes le problème de l'affaiblissement de gradient il est à noter que son utilisation optimale se fait lorsqu'elle employée dans les hidden layers (couches cachées du réseaux de neurones) .

Cette fonction n'en est pas moins exempt de défauts le fonctionnement de cette dernière peut en effet provoquer la mort de neurones en associant des poids nuls aux sorties de ceux-ci tout en réduisant les chances de quitter cet état d'où l'idée d'introduire des variantes

nommé ***Leaky ReLu*** modifiant les valeurs associées à un x < 0 en le multipliant par une constante **a** [CI.1] .

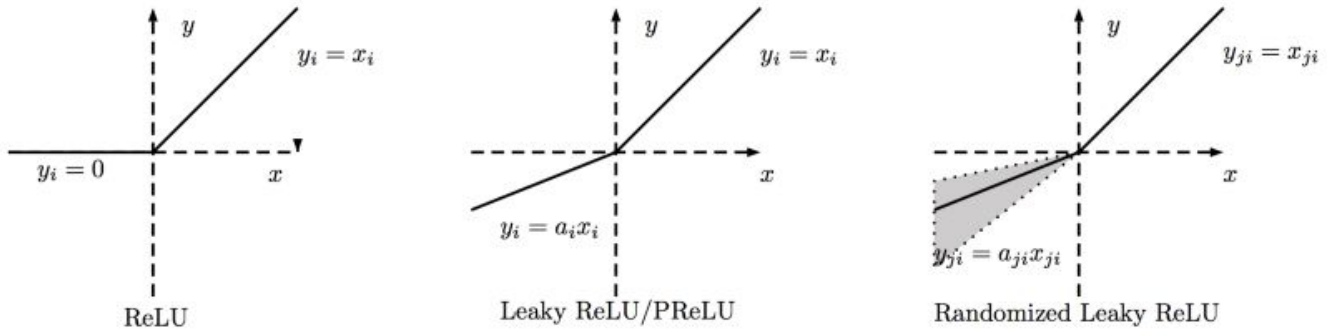

Figure I.10 différents tracées associées à la fonction ReLu et ses variantes LeakyReLu [f6]

I .6.4 - Apprentissage

L 'apprentissage est vraisemblablement la propriété la plus intéressante des réseaux neuronaux. Elle ne concerne cependant pas tous les modèles, mais les plus utilisés.

Il s'agit de la phase du développement d'un réseau de neurones durant laquelle le comportement du réseau est modifié jusqu'à l'obtention du comportement désiré.

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

[CI.1] Anish Singh Walia , Activation Functions and its types , which is better ?
https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f

Au niveau des algorithmes d'apprentissage, il a été défini deux grandes classes selon que l'apprentissage est dit supervisé ou non supervisé. Cette distinction repose sur la forme des exemples d'apprentissage. Dans le cas de l'apprentissage supervisé, les exemples sont des couples (Entrée, Sortie associée) alors que l'on ne dispose que des valeurs (Entrée) pour l'apprentissage non supervisé. Remarquons cependant que les modèles à apprentissage non supervisé nécessite avant la phase d'utilisation une étape de labellisation .

I .6.5- Backpropagation

I .6.5.1- Principe de base

Sans conteste la plus importante des mécaniques du *deep learning* il est en effet à l'origine de la faculté d'apprentissage des RN ; il intervient lors de la phase d'entraînement et indique aux RN les modification et ajustements à apporter aux valeurs de l'ensemble de ses poids afin d'aboutir à des résultat correspondant au mieux au attentes spécifiées. il opère couche par couche en démarrant par la couche de sortie pour ensuite remonter en propageant les modifications à apporter d'où son nom back propagation.

Une première version de cet algorithme est proposé dans les années 70 par P.Werbos mais ne serait réellement implémenter que plus tardivement avant de déceler des soucis de performances rendant cette première version inutilisable dans un contexte applicatif sérieux plus tard et durant les années ou la communauté portais le moins son attention à cette discipline D.Rumelhart ,G.hinton et R.Williams publient un article consacré à cette méchanique ou ils décrivent notamment une nouvelle approche plus efficace et plus performante de cet algorithme rendant pour la premiére sa mise en oeuvre en dehors des laboratoires possible

I . 6.5.2 -Fonctionnement et principes mathématiques

Plus concrètement la backpropagation repose sur l'emplois de deux algorithme la descente de gradient qui permet de calculer la valeur de la correction à apporter à chaque poids (ou biais) du réseaux en se basant sur une fonction de coût ,second algorithme, permettant de calculer l'écart entre les résultats attendus et ceux renvoyés par le réseaux et en appliquant sur celle ci des dérivées successives permettant de remonter jusqu'à chaque variable à ajuster.

I . 6.5.2.1 -Fonctions de coûts

Tel que stipulé plus tôt les fonctions de coût permettent d'évaluer l'écart entre les résultats attendus pour une entrée proposées au réseau de neurone et la prédiction de ce dernier ; il est à noter qu'il en existe un certain nombre de variantes qui selon l'architecture du réseau ou encore le type de problématique abordé peut plus ou moins convenir
Ces fonction sont réparties en deux catégories :

- Les fonctions de coûts associées à la classification :

  Sachant que dans le cadre d'une classification nous cherchons à obtenir une valeur unique associée à une classe ce qui implique que l'on travaille avec des valeurs finies

- Les fonctions de coûts associées à la régression :

  Lorsque on a des problèmes de ce type nous cherchons généralement à prédire des valeure continues comme le prix d'un appartement selon sa situation

Avant toutes chose il est à noter que les symboles suivant seront employés dans la définitions des fonctions que nous détaillerons plus bas

- n qui correspond au nombre d'exemple d'entraînement
- i indice d'un exemple de notre data set
- y(i) valeur de l'étiquette correspondant à l'élément i
- y^(i) valeur de la prédication du réseau pour l'élément i
- S(j) il s'agit de l'ensemble des valeurs de prédication accordée pour chaque étiquette possible par le réseau pour un élément i

Nous allons nous attarder sur quelques fonction de coûts plus généralement associées à la régression mais il n'est pas rare de les retrouver employés pour de la classification:

- Mean Square Error (Quadratic Loss) [carré de l'erreure moyenne]

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

Sans doute la fonction la plus simple qui permet d'obtenir l'écart moyen en se basant sur l'ensemble de prédictions effectuée lors de l'entraînement, le reproche le plus courent à l'égard de cette reste le fait qu'il est impossible de distinguer le sens de l'erreure (valeurs de la prédictions supérieures ou inférieures à celles attendues)

- Mean Bias Error

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

Elle se distingue de la fonction précédente par sa faculté à prendre en compte le sens de l'erreure cependant elle reste à manipuler avec prudence car les erreures négatives et positives peuvent se cumulées et s'annuler l'une l'autre

Nous pouvons enfin nous focaliser les fonctions de coûts propres aux problèmes de classification

- Hinge Loss / SVM Loss

$$SVM\ Loss = \sum_{j \neq y_i} Max\left(0\ ,\ s_j - s_{y_i} + 1\right)$$

Nous cherchons à vérifier si le score accordé à la catégorie réel de l'élément est plus important que la somme des scores des catégories incorrectes .

- Cross Entropy Loss

$$Cross\ Entropy\ Loss\ =\ -(y_i\ Log(\widehat{y}_i)\ +\ (1\ -\ y_i)\ Log(1\ -\ \widehat{y}_i))$$

La fonction la plus couramment employée ou le taux d'erreur augmente à mesure que la prédiction s'éloigne du résultat voulu l'une des forces de cette lois de calcule est sa sévérité avec les prédictions erronée confidente ou l'écart avec la bonne réponse est maximale . [CI.2]

I .6.5.2.2- La Descente de Gradient

Tel qu'établie plus haut la descente de gradient est un algorithme d'optimisation permettant de minimiser la valeure en sortie d'une fonction pondérée en permettant d'altérer pas à pas les valeurs des variables de cette fonction de sorte à aboutir à un minimum locale ou globale (à noter que le minimum globale est celui visé car il correspond à la valeure optimale recherchée).

Dans ce qui suit nous essayerons de nous plonger en profondeur dans les mécaniques sur lesquelles se base cet algorithme. [CI.3]

- **Notation des poids dans un réseaux de neurones**

Afin de désigner un poids particulier de façon non ambiguë dans un réseaux de neurone donnée nous employons la notation $w^{l}_{j\,k}$ où **k** est l'indice du neurone de la couche **(l - 1)** connecté au neurone **j** de la couche **l** (voir figure I . illustrant cette notation)

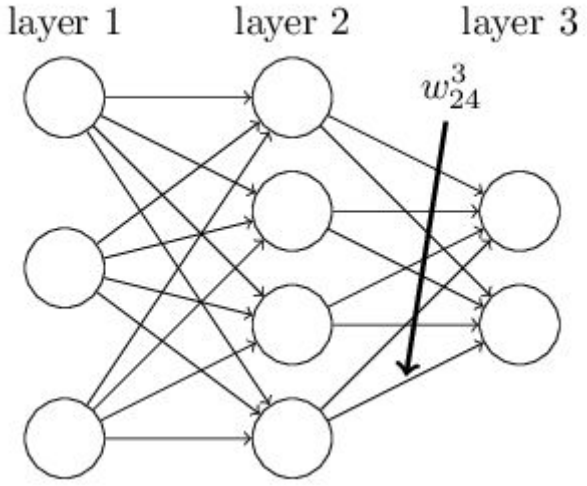

Figure I.11 : exemple de notation de poid [f7]

- **Notation des biais dans un réseaux de neurones**

Suivant une logique similaire les *bias* (que l'on peut traduire par biais) sont désigné comme suit b^l_j ou **l** correspond à la couche à laquelle appartient le neurone d'indice **j** recevant le biais qui nous intéresse (voir exemple tel qu'illustré par la figure I.12)

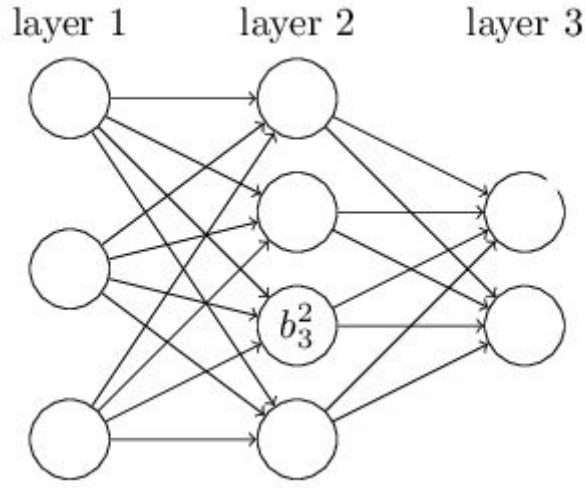

Figure I.12 : exemple de notation de bias [f7]

En nous basant sur cette notation il devient alors possible d'exprimer la formule désignant la sortie d'un neurone a_j^l soumise à une fonction d'activation $\sigma()$ d'indice **j** appartenant à une couche **l** comme suit

$$a_j^i = \sigma\left(\sum_k w_{j\,k}^l * a_k^{l-1} + b_j^i\right)$$

Cette formule peut ensuite être exprimé sous forme matricielle en considérant les poids associées à une couche **l** du réseau comme étant une matrice noté w^l, de même les bias associées à cette couche sont exprimé sous forme d'une matrice noté b^l, le dernier élément à exprimé sous forme matriciel reste la fonction d'activation $\sigma()$ pour se faire nous n'avons qu'a la notée $\sigma(v)$ où **v** est un vecteur (voire figure I.13 pour un exemple de déploiement d'une fonction sur une matrice)

$$f\left(\begin{bmatrix}2\\3\end{bmatrix}\right) = \begin{bmatrix}f(2)\\f(3)\end{bmatrix} = \begin{bmatrix}4\\9\end{bmatrix}$$

Figure I.13 : exemple de fonction appliquée à une matrice

Une fois ces transformations accomplis il nous est alors possible d'exprimer la fonction précédente comme suit

$$a^l = \sigma\left(w^l a^{l-1} + b^l\right)$$

De plus il est coutume de considérer la variable z^l définie comme suit

$$z^l = \left(z^l a^{l-1} + b^l\right)$$

L'algorithme de la descente de gradient permet d'évaluer la progression de la fonction de coût par rapport à chaque paramètre associé à cette dernière et procède par le calcule des dérivées partielles successives appliquées à la fonction de calcule de coûts **C** , ici nous considérons pour l'exemple que la fonction de coût employé est la fonction quadratique de formule

$$C = \frac{1}{2n} \sum_x \left\| y(x) - a^L(x) \right\|^2$$

Ou **n** est le nombre de d'exemples employées pour l'entraînement , **y(x)** la sortie attendue , **L** le nombre de couches de notre réseau et a^L la prédiction émise par le modèle.afin de pouvoir effectuer nos calcules nous considérerons le cas du calcule du coûts d'une seule prédication x .

- **Le produit de Hadamard**

Il s'agit d'une opération propre à l'algèbre linéaire que l'on appliquer sur des vecteurs de même dimension noté $\odot$ voici ci-bas un exemple de cette opération

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} \odot \begin{bmatrix} 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 1 * 3 \\ 2 * 4 \end{bmatrix} = \begin{bmatrix} 3 \\ 8 \end{bmatrix}$$

Afin de parvenir à calculer la progression des valeure des poids et biais d'un réseau l'algorithme de la descente de gradient procède par dérivation partielle successives que l'on peut exprimer comme suit

$\partial C / \partial w^l_{jk}$ pour les poids et $\partial C / \partial b^l_j$ pour les biais ,que nous employons afin d'évaluer la valeure d'une changement à apporter à chacune de ces variables pour ça il nous faut considérer une valeure intermédiaire δ^l_j permettant de gérer une marge d'erreur touchant le neurone d'indice **j** de la couche **l** qui se traduit par l'existence d'un Δz^l_j qui altère la sortie normale de ce neurone (il est due à l'évolution du neurone) qui sera alors $\sigma(z^l_j + \Delta z^l_j)$ de fait nous considérerons par la suite

$$\delta_j^l = \frac{\partial C}{\partial z_j^l}$$

Ainsi le fonctionnement de l'algorithme de la descente de gradient repose sur quatre formules que voici

$$\delta_j^L = \frac{\partial C}{\partial a_j^L}\,\sigma'\left(z_j^L\right)$$

La dérivé partielle de la fonction de coûts par rapport à la sortie de la couche de sortie **L** constitue le premier paramètre de cette fonction il permet de mesurer la vitesse de l'évolution de la fonction de coûts par rapport à la sortie du neurone d'indice **j** le second paramètre $\sigma'\left(z_j^L\right)$ lui permet de mesurer la sortie de cette couche **L**

En écrivant cette équation sous forme matricielle on aboutit à cette forme

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \qquad (1)$$

Ou $\nabla_a C$ est le vecteur exprimant l'évolution de la fonction C dont les composants sont le résultat de cette opération $\frac{\partial C}{\partial a_j^L}$ en appliquant cette formule à la fonction de coût quadratique On aboutit à cette expression (en effet la dérivé de MSE est $(a_j^L - y_i)$)

$$\delta^L = (a^L - y) \odot \sigma'(z^L)$$

La second formule permet d'exprimer l'évolution de la valeure de nos paramètres par rapport à celle de ceux de la couche suivante **L + 1** et s'exprime comme suit :

$$\delta^L = ((w^{L+1})^T\ \delta^{L+1}) \odot \sigma'(z^L) \qquad (2)$$

Il faut noter que nous employons la transposée de la matrice de poid de la couche suivante $(w^{L+1})^T$ cette formulation nous permet en réalité de remonter les couches du réseaux en passant par la dernière couche , en utilisant les équations 1 et 2 il est en effet possible de calculer la valeure de δ^L avec 1 puis en appliquant la formule 2 en considérant le résultat de 1 comme étant celui de la couche finale L la formule nous permettra de calculer δ^{L-1} et son

application successive en prenant en compte le résultat précédent permettra de remonter jusqu'à la couche initiale.

Cependant vous noterez que bien que la variable intermédiaire par couche δ^L soit calculée nous n'avons toujours pas réussi à calculer les valeure des dérivé des biais et des poids pour ce faire il faut considérer le deux formules restantes

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j \qquad (3)$$

Ou grâce aux deux formules précédentes (1 et 2) il a été possible de récupérer δ^L il nous suffit de considérer la valeure de ce dernier pour un neurone d'indice **j** au niveau de l'une couche (L , L-1 , ….,0)

Pour ce qui est des poids il suffit de soustraire à l'entré de la couche dont on souhait calculer les poids les biais précédemment calculées ce qui donne

$$\frac{\partial C}{\partial w^l_{jk}} = \delta^{l-1}_k \delta^l_j \qquad (4)$$

En appliquant ces formules successivement il est alors possible en parcourant le réseaux en sens inverse de calculer systématiquement l'évolution de la fonction de coûts par rapport à chaque poids et biais de chaque couche.

La figure suivante (figure A) est un exemple de l'application de l'algorithme de la descente de gradient ou il apparait clairement que l'on progresse pas à pas grâce au dérivées successives qui se traduisent graphiquement par des tangentes.dans la figure θ fait références aux valeurs des poids et biais de la couche en cours de plus le symbole $\widehat{\theta}$ est employé afin de désigner la valeure optimale des ces paramètres.

En suite la mise à jour des poids biais se fait pas à pas en soustrayant le gradient calculé par l'emplois des expressions précédentes multiplié par un paramètre η permettant de limiter le changement à apporter à chaque paramètre (c'est lui qui fixe la grandeur des pas) [la figure B montre le résultats obtenus si le pas est trop grand] ce qui peut se résumer par la formule suivante pour un paramètre a ou C est la fonction de calcule de coûts (notez que le symbole ∇ permet d'exprimer le gradient de la fonction associé à lui (plus précisément le

vecteur transposé des dérivation partielles appliqué à cette fonction par rapport à chaque paramètre))

$$a' = a - \eta \nabla C$$

Enfin , il faut noter qu'en générale les valeurs initiales des paramètres étant aléatoire la position de départ de l'algorithme l'est aussi.

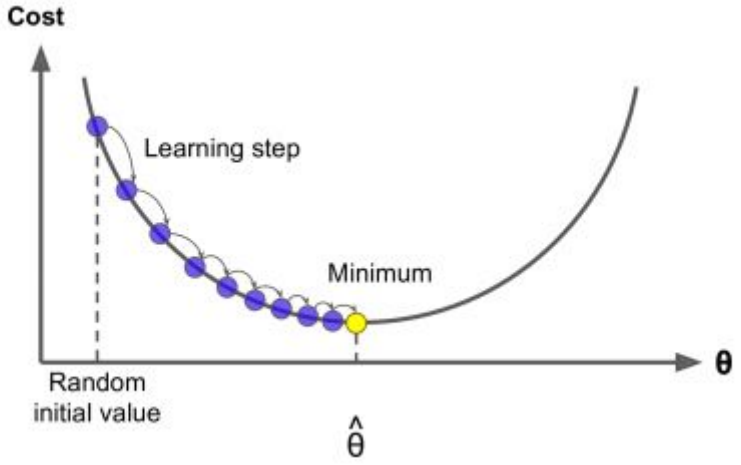

Figure A : illustration de l'algorithme de gradient descent [f14]

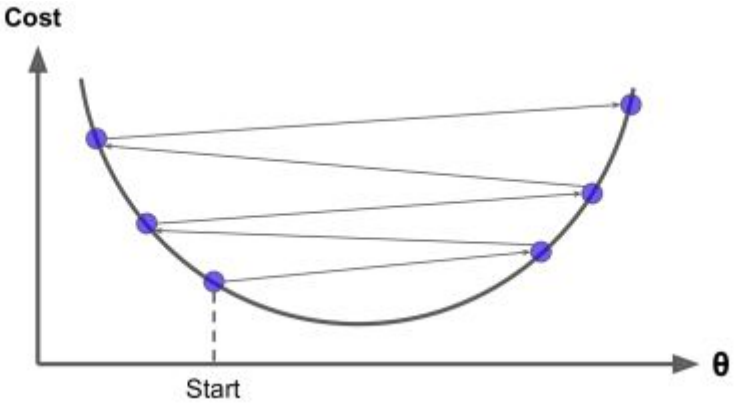

Figure B: illustration des résultats lié à l'emplois d'un trop grand pas [f14]

[CI.2] goerge seif ,https://towardsdatascience.com/understanding-the-3-most-common-loss-functions-for-machine-learning-regression-23e0ef3e14d3

[CI.3] Michael Nielson , http://neuralnetworksanddeeplearning.com/chap2.html

I .6.6 - Vers les réseaux de neurones profond

Depuis plus d'une décennies l'évolution des réseaux de neurones à mener ces dernier à se complexifier en gagnant en nombres de couches , en altérant plusieurs topologies de réseaux , en raffinant les algorithme de descente de gradients employés ou encore en diversifiant les fonctions d'activation employées à chaque couches ce qui nous à mener à parler de plus en plus d'Apprentissage profond ou "Deep Learning ".

Et pour cause et tel que mise en exergue par l'exemple du jeu de la vie de Conway qui démontre qu'un phénomène ou une simulation basé sur des règles simple peut très facilement produire des résultats imprévisibles et fascinant de complexité qui reste l'une des plus belles démonstrations du phénomène de l'émergence connu en physique qui décrit la naissance de nouvelles propriétés liées à l'augmentation du nombre ou de la quantitée; de faite à partir de banal perceptron au comportement très prévisible il devient très vite impossible de prédire ou de comprendre le fonctionnement complexe d'un réseau de neurone composé de centaines ou de milliers de neurones.

Bien que le terme deep learning apparaît la première fois dans les travaux de Rina Dechter en 1986 ce terme ne prend une réelle ampleur que depuis la dernière décennie ; aujourd'hui il sert ainsi à désigner une branche de l'apprentissage machine ou l'on emplois des algorithmes basées sur des réseaux de neurones multi-couches s'organisant selon un type particuliers d'architectures selon les problématiques pour lesquelles ils sont conçus (CNN pour la classifications d'images ,GAN pour la générations d'images , LSTM/RNN pour les traitement sur le langage , ..)

I .7- Les avantages et les inconvénients des réseaux de neurone:

I .7.1- Les avantages de réseaux de neurones:

- Flexibilité :

les réseaux de neurones artificiels ont la possibilité de généraliser et d'apprendre. Ils acquièrent des connaissances de leur environnement en s'adaptant aux paramètres internes et externes ,ils apprennent à partir d'exemples et s'adapte aux situations en fonction de ses conclusions et Ils généralise les connaissances pour produire des réponses adéquates à des situations inconnues.

- Bonne résolution:

les réseaux de neurones donnent de bons résultats même dans des domaines très complexes ils sont plus performants que les statistiques ou les arbres de décisions.

- Pertinence des donnés analysées:

Les réseaux de neurones peuvent travailler sur des données incomplètes ou bruitées. Cette imperfection des données peut être comblée par l'ajout de neurones supplémentaires à la couche cachée [C4].

- Apprentissage organique:

Les réseaux de neurones peuvent apprendre organiquement. Ils ne sont pas limités uniquement par ce qui a été donné à eux dans un système expert. Les réseaux de neurones peuvent généraliser à partir de leurs entrées, ce qui les rend précieux pour la robotique et les systèmes de reconnaissance de formes et pour l'analyse de données à grande échelle.

- La tolérance en pannes:

les réseaux artificiels ont un potentiel élevé de tolérance aux pannes. Le réseau est capable de régénérer un défaut dans l'un de ses composants sans la perte des données stockées ,Il utilise des instances et des exemples du passé pour remonter le fonctionnement d'un nœud endommagé ou les autres composants du réseau [C4]

I .7.2- les points faibles de réseaux de neurones:

- Non optimalité de l'architecture:

Il n'existe pas encore de moyens permettant de définir l'architecture optimale du réseau de neurones. parce que pour définir l'architecture du réseau de neurones, il faut déterminer tout le nombre de couches cachées du réseau et le nombre de neurones pour chacune des couches et tout cela relève de l'intuition de l'opérationnel [C5].

l'intervention humaine:

Comme c'est le cas du choix de la structure optimale du réseau de neurones, ce système fait souvent appel à l'intuition de l'utilisateur. L'apprentissage est guidé par des paramètres définis manuellement par l'opérationnel. Celui-ci devra par exemple définir à quel moment devra s'arrêter l'apprentissage pour que le réseau conserve ses capacités à généraliser.

- Codage des entrées:

toutes les entrées doivent être défini dans un intervalle entre 0 et 1 ce qui engendre des traitements supplémentaires et risque de fausser les résultats.

- Boite noire:

Les réseaux de neurones peuvent être assimilés à une boîte noire qui donne une réponse quand on lui fournit des données mais qui ne délivre pas toujours de justification simple à analyser. c'est très difficile d'analyser et comprendre le fonctionnement en face d'un problème donné,et de choisir la structure (type, nombre de noeuds, organisation, connexions,...etc) la mieux adaptée au problème.

I .8 - les domaines d'application:

Aujourd'hui, les réseaux de neurones ont de nombreuses applications dans des domaines très variés [C6] :

- traitement d'image : compression d'images, reconnaissance de caractères et de signatures, reconnaissance de formes et de motifs, chiffrement[1], classification, ...
- traitement du signal : traitement de la parole, identification de sources, filtrage, classification, ...
- traitement automatique des langues : segmentation en mots, représentation sémantique des mots (plongements lexicaux), étiquetage morpho-syntaxique, traduction automatique, ...
- contrôle : diagnostic de pannes, commande de processus, contrôle qualité, robotique, ...
- optimisation : allocation de ressources, planification, régulation de trafic, gestion, finance, ...
- simulation : simulation boîte noire, prévisions météorologiques
- classification d'espèces animales étant donnée une analyse ADN
- modélisation de l'apprentissage et perfectionnement des méthodes de l'enseignement
- approximation d'une fonction inconnue ou modélisation d'une fonction connue mais complexe à calculer avec précision [C6]

[C4] : wikiversity, avantages des réseaux de neurone; le lien : https://fr.wikiversity.org/wiki/R%C3%A9seaux_de_neurones/Avantages_et_possibilit%C3%A9s
[C5] : wikiversity, Les inconvénients des réseaux de neurone;le lien : https://fr.wikiversity.org/wiki/R%C3%A9seaux_de_neurones/Points_faibles_et_limites

[C6] : wikiversity, domaine d'application:;le lien :https://fr.wikiversity.org/wiki/R%C3%A9seaux_de_neurones/Applications_des_r%C3%A9seaux_de_neurones

I . 9. Approches du deep learning

On distingue dans le deep learning trois approches (voir une quatrième [qui constitue un cas particulier]) définies par le type de données destiné à l'apprentissage de notre réseaux de neurone à savoir:

- Le Supervised Learning :

  Ce type de solution est employée dans le cas ou l'on dispose d'un set de données étiquetées ou l'on se base sur le fait d'inculquer à un réseau de neurones une logique d'association en injectant durant la phase d'entraînement des entrées à analyser tout indiquant au réseaux les sorties attendue en fin de traitements pour ensuite corriger le réseaux si il venais à ne pas proposer la solution attendue. Dans ce cas de figure notre problématique se résume comme suit:
  ayant une Valeure x en entré et une valeur y en sortie , proposez f(x) tel que f(x) = y

  L'ensemble des problématique propres à ce type d'approche sont en générale des problèmes de classification ou de régression

- L'Unsupervised learning :

  Ce type d'approche est requis dans les cas où les données à disposition en sont pas étiquetées ici lorsqu'on les présente au réseau de neurone celui-ci doit parvenir à identifier de relations entre ces dernières que se soit en les regroupant ou en détectant des anomalies en se basant sur l'analyse statistiques des données ou encore des approches de clusterisation

  Le problème abordé ici peuvent se résumer par : étant donné les entrées $x_0 , \dots , x_n$ trouvez une relation les liant

- Le Semi-supervised learning

  Le propre de cette approche est de combiner les deux précédentes ou généralement car disposant d'un data set semi étiqueté l'approche de l'apprentissage supervisé n'est pas envisageable on opte pour cette approche ou le réseaux en se basant sur

les données étiquetées parviendras à recadrer son jugement et ses prévisions lors de phase d'apprentissage lorsqu'il chercheras à mettre en évidence les paternes présents dans les données du set .

Cette approche est très prisées par les chercheur car elle permettrait à terme de se passer d'experts des domaines de travail lors de la phase de préparation de données permettant de simplifier l'automatisation tout en en réduisant les coûts de mise en oeuvre .

- Reinforcement learning :

Ou l'apprentissage par le test généralement employé dans des conditions offrent la possibilité d'effectuer des test à la chaîne ou des règles de progression existe appelé récompense et ou les règle propres au fonctionnement du problème sont connue d'avance cette approche est particulière efficace une fois appliquée à des jeux vidéos ou l'on observe des résultats exceptionnels en sortie la machine parvenant en quelques heures à passer maitre et même à battre sans effort des champions dans jeux réputé hautement complexes tels que le jeu de Go (AlphaZero) ou Starcraft (GoogleIA).

I . 10. Quelques architectures de réseaux de neurones profond standard

Tel que vu précédemment , la construction de réseaux de neurones est fortement lié à la problématiques à laquelle ils sont destiné . ainsi à force de d'essais et d'expérimentation l'on a vu naître des architectures qui font office de canon dans le domaine parmi celles-ci :

- Les CNN réseaux de neurones Convolutifs

Théorisées en fin des années 80 puis implémenté et peaufinés jusqu'à nos jours cette structure à su s'imposer comme la plus efficace pour la résolution des problématique de classification notamment d'images sa force repose fortement sur les couches convolutive qu'il emplis afin de réduire la complexité des inputes à analyser tout en conservant le plus d'information pertinente à leurs sujet sans oublier l'efficacité de ceux-ci lors de l'extraction de caractéristiques.

- Les RNN réseau de neurones récurrent

  Ils se distinguent de part le type de liaison entre neurones et couches de leur structure formant un graphe orienté respectant une temporalité particulière ce qui leur offre une forme de mémoire leur permettant d'analyser efficacement les données séquentielles tel que du texte ou encore des discours ou des vidéos

- Les LSTM Long Short-Term Memory neural networks

  Sont une amélioration des RNN découverts en 1997 par Hochreiter et Schmidhuber mais dont l'impacte n'est apprécié que depuis peu ou leur efficace pour l'analyse de texte fut l'un des moteur poussant le géant chinois Baidu à investir dans des technologie basé sur cette architecture ou encore l'intégration de ce type de réseau dans l'application de recherche vocale de Google

- Les GAN Generative adversarial neural networks

  Sont en réalité composés de deux réseau de neurones concurrents ou l'un est chargé de traiter les données en entré pour en extraire une logique la ou le second cherche à recomposer une sortie similaire à la donnée à partir de cette logique ; ils sont très populaire car à l'origine d'une nouvelle vague d'IA artiste capable de créer des oeuvres en se basant sur les règles apprises lors de leurs entrainement

I . 11. Cycle de développement d'une solution deep learning

Afin de mettre en oeuvre nos solutions nous nous devons de procéder par étapes, en effet ce type de projets se conforme à une processus inspiré des pratiques de conception et développement de solution basées sur la data mining ,faisant état de norme dans le domaine , et pour cause le deep learning est un domaine sous domaine du machine learning où les techniques de ce derniers sont employée dans le cadre du data mining

Il est impératif de noter que bien que ce cycle se distingue des cycles dit classiques il ne s'agit en fait que d'une adaptation de ces derniers ainsi bon nombres des phases que nous allons aborder ci-bas sont similaire à d'autres plus classiques. De plus , il est à noter que le développement d'une telle solution peut se faire dans le cadre de l'ajout d'un module à une application existante ou non lui permettant ainsi de s'intégrer dans un processus de développement cyclique classique

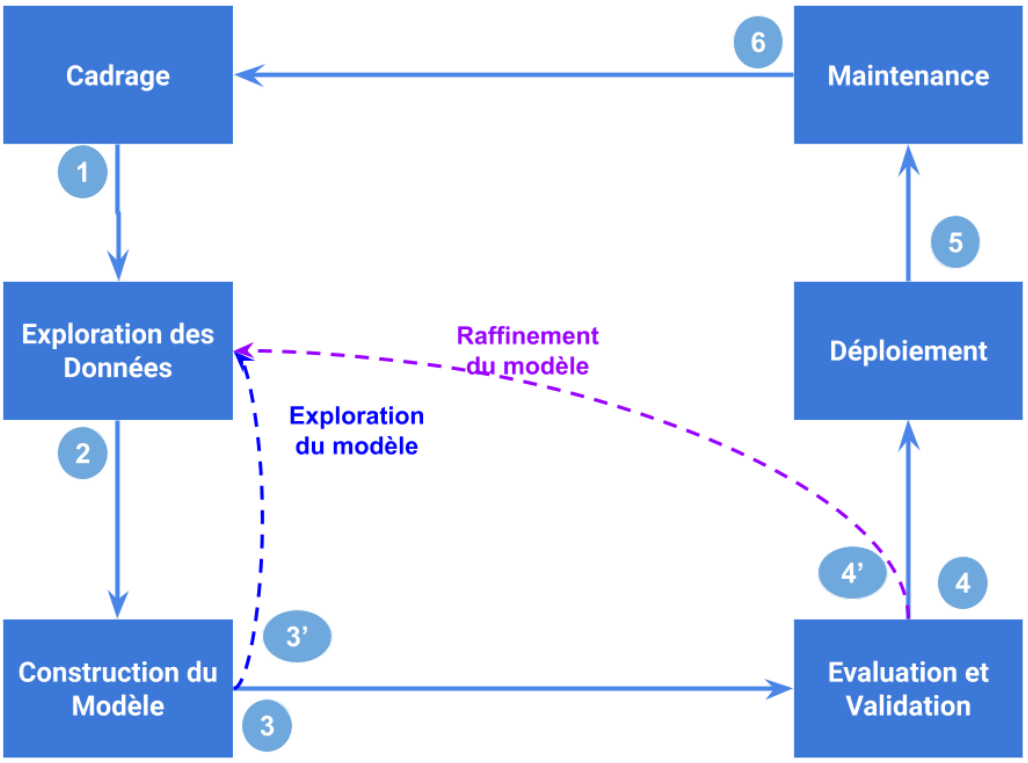

Figure I.13 **: Cycle de développement du solution DL**

Phase 1 : Cadrage du projet :

L'on se charge durant cette d'évaluer les objectifs à atteindre , de nous imprégner du contexte de notre projet et de préparer un plan pour ce derniers .

Plus concrètement l'une des propriétés les plus intéressante de ce domaine (DL) est au delà du bagage technique nécessaire à la conception et l'implémentation de la solution il est devenu impératif que les équipes ,en charge de ce type de projets ,soit elles mêmes fortement au faits des spécificités du domaine , ainsi une partie de taches se focalisent sur l'étude du domaines afin d'en comprendre les mécanismes et lois qui serviront dans la compréhension puis la résolution de la problématique abordée.

Par la suite forts de leur compréhension du domaine les membres de l'équipes se chargent de proposer une stratégie, des métriques et des principes permettant de résoudre la problématique à traiter c'est aussi à cette étape qu'une estimation de faisabilité est effectuée

S'ensuit une planification minutieuse des étapes de travail résumé par le schéma (Figure I.13)

Phase 2 : Exploration des données

Cette phase est sans conteste la plus délicate de tout le cycle car ses résultats constituent à la fois le facteur le plus déterminant dans le choix de la méthode de résolution à adopter mais aussi le paramètre le plus influent sur la qualité et la viabilité du modèle que nous cherchons à construire .

Comme précédemment expliqué dans le chapitre consacré à la présentation des principes du deep learning , nous savons que le modèle recherche des paternes, relations logiques dans les données d'entraînement que nous lui présentants ce qui implique un certain nombre de choses :

- la nature , l'architecture et les paramètres d'évaluations de notre modèle dépendent entièrement du type données que nous traitons .

- l'adaptabilité de notre modèle à de nouvelle situations est fortement lié à la variées des exemples d'entraînement qui lui seront présentés .

Cette phase se déroule en générale en deux temps :

- Première étape : Collecte de données

  Plus concrètement durant cette phase nous cherchons à regrouper le plus de données possibles relative à la problématique posé au préalable ; on se charge alors de collecter les données à partir des différents supports existants (numérique ou non au quel cas il faut numériser l'information) puis l'on s'attelle à la décrire en précisant son type (fichiers textes , vidéos , images ,fichiers audio , BDD SQL , ….) son contenu que ce soit un bref descriptif de la nature de la donnée ou encore sa dimension

  En se charge ensuite de la visualiser puis de la transformer si besoin de sorte à ce que son format soit utilisable puis l'on se charge d'estimer la qualité des données à disposition(incomplétudes , incohérences , dommages ,..)

- Seconde étapes :

  De plus , afin d'éviter les écueils liés à un dataset ne représentant pas la réalité il est vitale d'en concevoir un qui soit équilibré (ou chaque classe est représentée de la même façon , ainsi chaque étiquette à la même probabilité d'apparaître lors de l'entraînement) ; car et c'est un problème courant dans le cas de type de projet un

modèle travaillant sur un dataset dit déséquilibré les modèles en question finissent par souffrir de biais

Les données collectées et prêtes à l'emplois obtenu suite à l'étape précédente,subissent une sélection rigoureuse afin de ne conserver que celles jugées pertinentes ,celles-ci sont ensuite nettoyées (suppression de duplicatas , des bruits et points isolées , ….) les données peuvent aussi être enrichies par l'ajout d'étiquettes par exemple ou alors lissées de sorte à éliminer de l'information jugée non pertinente ou trop volumineuse (exemple changement de l'échelle d'une image à la baisse, suppression d'une colonne dans une table,...)

le résultats est alors formaté de sorte à être exploité directement par le modèle

A la suite de cette phase la donné dite exploitable est alors subdivisées en sets dont l'utilité est cruciale pour la suite des opérations ; selon la quantité et la nature de la données la subdivision de cette dernières; varie mais veuille en générale à constituer deux set

A commencer par le plus volumineux (selon les cas 9/10 des données) le set d'entraînement qui servira à entraîner notre modèle [nous reviendrons sur ça plus en détaille dans la suite de ce document] celui-ci bénéficiant fortement du grand nombre d'exemple fournis durant cette phase de plus il est vitale de garantir un équilibre dans le set de sorte à ce qu'il y ait un nombre d'occurence approximativement égale pour chacun des cas que l'on souhaite inculquer au le modèle .

Le second set lui dit de test et validation est plus modeste et constitué d'échantillons sélectionnés au hasard dans l'ensemble de données une forte variété des cas pris en comptes permet de meilleurs mesures des performances réelle du modèle ce set lui ne sera présenté au modèle qu'à la fin de son entraînement ainsi ce derniers n'auras pas pu s'y adapter de façon négative (Overfitting)

Phase 3 : Construction du réseau de neurones :

En se basant sur la nature de la donnée obtenue lors de la phase précédente nous chercherons à choisir une approche puis des méthodes et architecture de réseaux en conséquence ,permettant de répondre au mieux à la problématique posée

Les critères d'évaluation de notre modèle sont aussi choisies durant cette étape

Une fois le modèle construit il nous faut préciser les critère d'évaluation à employer lors de son entraînement à savoir dans le cas qui nous intéresse [voir chapitre suivant "réalisation"]:

A - La fonction d’optimisation à utiliser :

C’est elle que l’on charge de recalculer les poids et variables de notre modèle au fure et à mesure de l'entraînement de sorte à obtenir des résultats satisfaisant cette fonction prend notamment en considération les résultats de la fonction de Loss

Le principe clé derrière le fonctionnement de l’optimiser est la back-propagations [voir chapitre I] ;il est à noter que la majorité des optimizer reposent sur une version plus ou moins altérée de l’algorithme de la descente de gradient permettant entre autre d’optimiser les valeurs des variables employées par notre

B - La fonction de “Loss” ou fonction de coût :

Elle permet d’associer une valeure numérique à un état du système par exemple une estimation de l'écart entre un résultats attendu et le résultat renvoyé par le modèle ces résultats sont alors utilisés dans la phase d’optimisation du système ce qui correspond à ce que nous désignons comme apprentissage

C - Des métriques :

Ce sont des paramètres supplémentaires calculées au fur et à mesure de l'entraînement nous permettant de surveiller son dérouler il est à noter que ceux-ci n'interviennent pas dans le processus d'apprentissage et servent de valeurs de monitoring on peut citer parmis eux la précision ou exactitude du modèle ,

Une fois l’ensemble de ces paramètres choisis , à noter que certains sont intrinsèquement liés à la nature de la données ou même au problème traité mais il est plus courant de les choisir en se basant sur l'expérience ou encore des sessions d'entraînement multiples ou même l'instinct

Il ne reste qu'à préciser la durée de l'entraînement en signalant le nombre de fois ou le set d'entraînement sera parcourus ou encore pour quelle valeur de la fonction de loss il est préférable d'interrompre les passes ou même encore la sensibilité des variables aux changement à apporter lors de l'entraînement ainsi le modèle prend en entrées notre set de données et entame ses cycles de calcules et d’ajustements

Il est à noter que les calcules pouvant selon la nature de la donnees et la complexité de l’architecture du modèle (nombre de couches et nombre de neurones natures des couches

)être très long et coûteux en ressources matérielles bon nombre de laboratoire de recherche indépendants publics ou privés se sont penchées sur la question il est à noter que depuis les années 2010 l'emplois de processeurs graphiques doté de la technologie CUDA de Nvidia est préconisée due à la possibilitées de paralléliser les calculs à l'aide des nombreux coeurs et threads propres à ce type de processeurs et plus récemment 2015 il est possible d'employer des Tensor Processor unit ou TPU afin de réduire les temps de calculs ceux-ci étant des processeurs à l'architecture particulière optimisée pour les calcules sur les tensors de données manipulés ce composant étant une technologie propriétaire de Google proposées sur leurs plateformes cloud .

Exploration du Modèle :

Afin de s'assurer un suivie de la progression de la construction du modèle que nous mettons en oeuvre il est généralement conseillé d'implémenter des versions simplifiées de ce dernier afin de valider le processus de construction et d'entraînement avant d'entamer une phase d'enrichissement de ce dernier

Le fait de travailler avec des modèles simplifiés permet de tester plusieurs architecture en parallèles sans prise de risque ou perte de temps majeur en cas d'échecs

La possibilité de reprendre les phases 2 et 3 permet d'adapter au mieux son modèle ou encore de corriger très tôt des lacunes ou des erreures graves survenues lors du traitement des données

Phase 4 : L'évaluation et la validation :

A la fin de la phase précédente nous obtenons en sortie un modèle entraîné que nous allons mettre à l'épreuve en le confrontant pour la premières à un set de données de test qu'il n'as jamais vu de sorte à ne pas biaiser le test car en effet l'une des principale hantises du deep learning est aussi l'un des ces points fort la machine étant capable d'apprendre à une vitesse vertigineuse mais ne le faisant que dans un soucis d'optimisation il arrive que le modèle obtenu en sortie se soit adapter de façon excessive aux données d'entraînement employé lors de sa phase d'apprentissage c'est ce que l'on nomme l'over fitting en d'autres terme le modèle à appris à retenir les données en entrée et c'est adapter pour ne traiter que ces dernières plusieurs paramètres entre jeu dans l'apparition de cette tare dont :

- La maigreure du dataset : peu de données peu variées , ou très répétitives (cas de datasets enrichie artificiellement)

- Des paramètres d'évaluation inadéquates

- Une phase d'entraînement trop longue

Il est à noter qu'il existe des procédés et artifices logicielle permettant de réduire l'impacte de ce problème sur nos résultats

Suite à l'évaluation du modèle un ensemble de métriques est recalculés afin d'évaluer de façon plus vraisemblable le modèle plusieurs métriques sont importantes mais il est plus courant de se focaliser sur la précision de test , le coût de test et dans le cas d'une classification par exemple les valeurs de la matrice de confusion (valeurs de cardinalité nbr de cas au carré dans) permettant d'estimer le pourcentage des faux positifs , faux négatifs entre autres .

A l'issu de cette phase si les résultats sont jugées concluant les modèle est prêt au déploiement

Raffinement du modèle :

Durant cette phase il est possible d'effectuer selon les résultats obtenus par le modèle lors des ses entrainement un retours arrière vers la phase de préparation de données afin de corriger d'éventuelles problèmes de précision ou encore d'overfitting à savoir une trop grande adaptation du modèle au set d'entraînement le rendant totalement incapable de traiter de nouvelles données.

Auquel cas plusieurs solution sont possibles selon le problème il est possible d'opter pour l'enrichissement de données dans notre cas de figure (imagerie) cela consiste en générale à appliquer des transformations du type mirroire , rotation , redimensionnement ou encore découpage aux données de base afin de produire de nouvelles entrées exploitable .

Il est aussi possible qu'une phase de pré-traitement incomplète ou mal calibrée aboutisse à la conservation ou la suppression de données inutile / importante auquel cas changer de technique ou d'algorithme , réduire le nombre de filtres , ou en appliquer davantage résout la plupart des défaillances .

Phase 5 : Le déploiement

Il existe un certain nombre de méthodes permettant de déployer un modèle une fois ce derniers au point la solution la plus commune étant de passer par une plateforme cloud offrant en toute simplicité l'environnement d'exécution nécessaire au fonctionnement du modèle puis de créer une interface permettant d'interagir avec directement ou encore de l'intégrer dans un projet déjà existant.

Bien que la solution cloud soit celle de prédilection car elle permet de répondre aux besoins en puissance de calculs des modèles et cela à moindre coût , l'emplois de bibliothèque pensées pour l'optimisation et la réduction des coût il devient possible d'implémenter des solution capablent de s'exécuter sur des terminaux aussi légers que des téléphones la contrepartie d'une telle pratique étant une limitation des performance du modèle.

Il est à noter que depuis peu les framework de deep learning sont aujourd'hui capable de gérer l'exécution de code au sein même d'un navigateur web comme le fait par exemple la version Js de TensorFlow ce qui permet virtuellement de supporter la création et le déploiement de modèle sur tout appareil capable d'exécuter un navigateur web la encore au prix de performances réduites .

Phase 6 : La maintenance

Une fois mis à disposition des utilisateurs le modèle peut toujours êtres suivis et maintenu en effectuant des mises à jours de ce derniers en l'entrainant avec de nouvelles données accumulées avec le temps ce qui garantie une longévité accrue aux solutions implémentées.

Pour ce faire il existe plusieurs solutions selon les frameworks et les architectures que nous verrons plus en détaille dans le chapitre consacré à la réalisation

I . 12. Définition de la pathologie à traiter : la pneumonie

I .12.1Définition:

C'est une atteinte infectieuse du poumon profond au niveau des structures alvéolaire(pneumonie franche),bronchiolo-alvéolaire(bronchopneumonie) ou interstitielle

I .12.2 Épidémiologie :

Selon une enquête nationale de santé faite en 1990 , la pneumonie représente 27% des motifs de consultations et 35% de morbidité en algérie

I .12.3 Diagnostic positif:

C'est l'ensemble des signes cliniques et paracliniques qui posent le diagnostic de la pneumonie

- ➢ Clinique: il existe 02 formes de pneumonie selon que la représentation clinique est typique ou non

  - forme typique (Pneumonie franche lobaire aiguë)elle est caractérisée par :

    Un début brutal avec:

    Fièvre(40 C°)
    - Douleurs thoracique
    - Toux productive avec expectoration purulente
    - Dyspnée à type de polypnée
    - Syndrome de condensation pulmonaire avec
      Palpation: Augmentation des vibrations vocales
      Percusion:Matité
      Auscultation : Abolition du murmure vésiculaire

  - forme atypique= interstitielle : elle se caractérise par

    - Début progressif
    - Syndrome pseudo grippal
    - Toux sèche

I .12.4 Radiologie:

Sur une radiographie du thorax on retrouve :

Une opacité alvéolaire homogène systématisé à un lobe avec un bronchogramme aérique

Diagnostic différentiel: Afin d'éliminer ce qui n'est pas une pneumonie :

embolie pulmonaire
oedeme aigue du poumon
cancer broncho pulmonaire
pneumonie médicamenteuse

Diagnostic étiologique: la pneumonie peut être due à une infection

1- Bactérienne: elle cause le plus souvent une pneumonie typique
- Le pneumocoque est l'agent pathogène le plus fréquemment isolée
- autres agents: Legionella pneumophila
  Mycoplasma pneumoniae
  Chlamydia psittaci
  Haemophilus influenzae

2- virale: - virus influenzae(dans le cadre de la grippe saisonnière)
- virus respiratoire syncytial

Diagnostic de gravité: Toujour rechercher les critères de gravité dont la présence change la conduite à tenir

- âge>=65 ans
- confusion troubles de conscience
- polypnée:fréquence respiratoire >=30 cycles/mm
- PAS(pression artérielle systolique) < 90 mmhg ou PAD(pression artérielle diastolique)<=60 mmhg.
- Pao2(pression aléria en oxygène) <60 mmhg
- Pneumonie nosocomiale

la conduite à tenir sera la suivante :
Absence de ces critère == traitement ambulatoire
La présence de 1 critère= évaluation à l'hôpital

I .12.5 Complications :

En l'absence d'un traitement précoce et adéquat , la pneumonie évolue vers des complications qui aggravent l'état de santé du malade :

- Epanchement pleural
- abcès du poumon
- détresse respiratoire
- choc septique

I . 13 . La Radiologie en pneumologie

En médecine, les examens complémentaires sont d'une aide précieuse au praticien dans un but de diagnostic positif et/ou étiologiques .parmi ceux là, ya les examens radiologiques et biologiques

Parmi Les examens radiologiques, on distingue entre autres, 03 types selon leurs principes de fonctionnement :

- **Imagerie pulmonaire par rayon X** :

Radiographie de thorax (telethorax), Scanner (tomodensitométrie =TDM) : Ce sont des examens invasifs qui expose au risque des rayonnements ionisants.

Principe:

Le principe est celui des ombres chinoises. Le faisceau de rayons X produit par un tube à rayons X est émis en direction de la zone du corps humain à examiner, son intensité est « modulée » par l'absorption différentielle des organes traversés. L'image est recueillie en sortie sur un détecteur (plaque photographique par exemple).

Le coefficient d'atténuation m dépend de la composition chimique des tissus traversés. Il est élevé pour l'os, moyen pour les tissus mous et faible pour la graisse. Les os contiennent en effet des sels minéraux (phosphore, calcium, magnésium) qui sont des éléments de numéro atomique plus élevés que les constituants principaux des tissus mous (oxygène, carbone, hydrogène, azote…). Ils absorbent donc plus les rayons X.

Indications:

Le telethorax est un examen de 1ere intention qui sera complété ou non secondairement par la TDM qui explore mieux les structures anatomiques dans le but de poser un diagnostic positif et/ou étiologique

- Dyspnée
- Douleur thoracique fébrile ou non

- Diagnostic des principales infections ou affections pulmonaires comme tuberculose ou de la bronchite chronique

- Diagnostic, dépistage et le suivi des traitements des cancers du poumon

- Systématique (médecine du travail, bilan préopératoire...)

- **Imagerie par résonance magnétique (IRM)**

Considérée comme un examen de seconde intention après l'examen tomodensitométrique ,elle explore mieux le parenchyme pulmonaire et les vaisseaux est le plus souvent

Principe :

l'IRM fonctionne grâce à des propriétés quantiques de l'hydrogène (Le corps humain étant majoritairement composé d'eau, et l'eau majoritairement composé d'atomes d'hydrogène, ceci concerne environ 60 % des atomes de notre corps.) et d'importants dispositifs magnétiques Pour fonctionner, on place le patient dans un puissant champ magnétique : 100 000 fois le champ magnétique terrestre, ce qui est suffisant pour que les noyaux d'hydrogène s'orientent tous dans le même sens.
On envoie ensuite des impulsions magnétiques (appelées *radiofréquences* qui vont détourner les atomes d'hydrogène de leur alignement. Le changement d'orientation des noyaux d'hydrogène va induire un bref courant électrique qui permettra de produire une image

Indications :

En pratique, l'IRM est surtout utilisée pour résoudre des problèmes laissés en suspens par la tomodensitométrie.

- Extension des tumeurs de l'apex (paroi, vaisseaux, plexus brachial, vertèbre …) : IRM > scanner caractérisation d'une masse médiastinale : IRM > scanner

- Extension à la paroi des tumeurs broncho-pulmonaires : IRM = scanner

- Exploration angiographiques des vaisseaux du cou et du thorax (troncs supra-aortiques, veine cave supérieure, aorte, artères pulmonaires) notamment chez les patients allergiques à l'iode ou les patients jeunes

- **Imagerie par ultrasons : échographie pulmonaire**

L'échographie est un examen non invasif qui n'expose pas aux rayonnements ionisants.

Principe:

Il s'agit en fait d'une technique d'imagerie médicale basée sur l'utilisation des ultrasons .Ces derniers correspondent à des ondes sonores de hautes fréquences (supérieure à 20 000 Hz) dont la propagation va être modifiée par les obstacles qu'elles rencontrent sur leur chemin. En fonction de la densité du milieu traversé, ces dernières vont renvoyer un "écho" de plus ou moins forte intensité. L'appareil échographique est alors chargé d'interpréter ces signaux sous la forme d'images diffusées en temps réel.

Indications:

- Diagnostic des épanchements de la plèvre
- Guide la ponction ou le drainage des épanchements pleuraux cloisonnés
- Diagnostic et au prélèvement (biopsies) de certains nodules du poumon

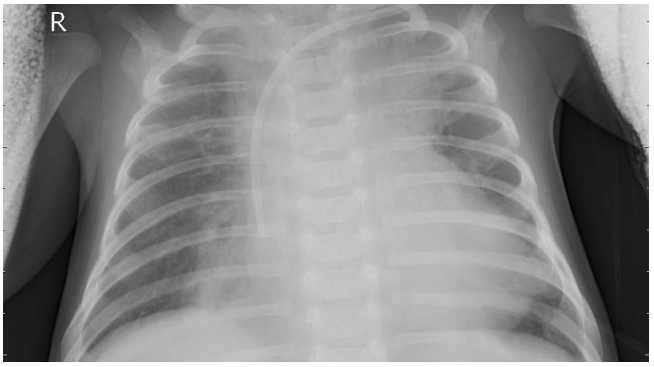

Figure I.14.1 : téléthoraxe d'un patient atteint de pneumonie [f9]

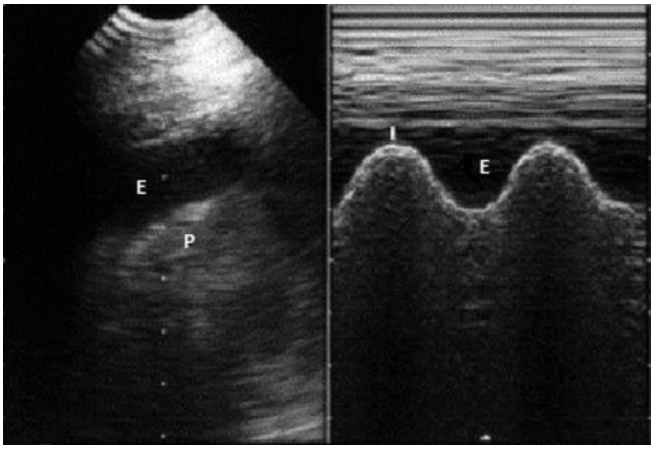

Figure I.14.2 : échographie pulmonaire [f15]

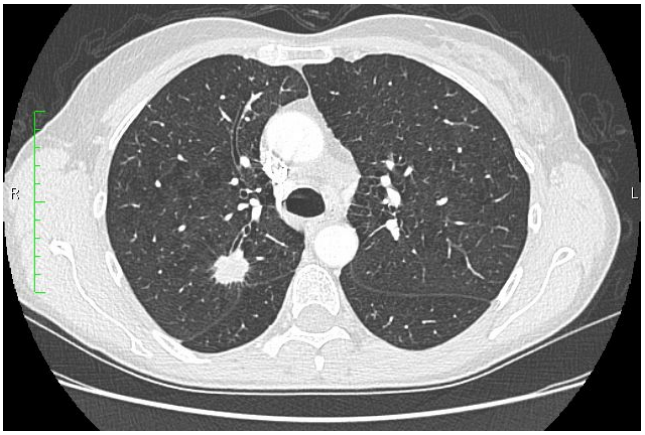

Figure I.14.3 : IRM pulmonaire [f16]

I . 14 les mesures que nous emploierons

Dans ce qui suit nous tâcherons d'expliquer les métriques que nous avons employés dans le cadre de l'évaluation de notre modèle sur lequel nous reviendrons par ailleurs dans les prochaines ligne.

Etant donnée la nature du domaine applicatif de notre thème à savoir la santé, et plus précisément le diagnostic automatisé, il nous a fallu employer les mesures de fiabilité les plus communes dans ce cas de figure, à savoir la sensibilité (Sensitivity) , la spécificité(Specificity)et la justesse(Accuracy) nous reviendrons dans ce qui suit sur chacun de ces principes en expliquant à la fois les formules de calcul associées mais aussi l'intérêt de l'usage de chacune d'entre elles .

Avant toute chose il est primordial de rappeler un certain nombre de principes qui seront utilisés dans la définition de nos lois de calcule ,celles-ci se sont modeler de sorte à répondre aux besoin du corps médical lorsque ce dernier a eu recours pour la première fois aux outils de diagnostique auxiliaires afin de minimiser les risques liés à l'emploi de méthodes automatique, il s'est très vite avéré vital de disposer de mesure et de contrôle permettant d'évaluer la fiabilité de ces outils . Dans un premier temps la fiabilité de ces solutions fut mesurer en se basant sur sa justesse c'est-à-dire la capacité du système à donner un bon diagnostic ; dans l'idéal ,un système fiable serait en mesure de fournir une précision de 100 % . Ce qui , hélas ,n'est pas le cas dans le monde réel car en effet, il subsiste toujours un risque d'échecs ou d'erreur ; c'est pourquoi de nouvelles mesures se sont imposées de sorte à cerner au mieux les capacités réelles du système testé .

Afin de pouvoir utiliser chacune des mesures dont nous parlons il faut tout d'abord présenter les variables qui sont manipulés par ces dernières :

- **nombre de personnes malade**
- **nombre de personnes saines**

- **les vrais positif (VP) :**
  sont définis comme étant des personnes souffrant réellement de la maladie et diagnostiquées comme telles par notre système.

- **les faux positifs (FP) :**
  sont définis comme des personnes ne souffrant pas de la maladie mais diagnostiquées comme étant malade par notre système.

- **Les vrais négatifs (VN) :**
  sont définis comme étant des personnes saines diagnostiquées comme telles par notre système

- **les faux négatif (FN) :**
  sont définis comme étant des personnes malades diagnostiquées comme étant saines par notre système

Notez bien que de par la définition de chacuns de ces paramètres ,le plus critique et celui des faux négatif ,et pour cause ,le système dans ce cas de figure considère à tort qu'une personne souffrant de la maladie que nous cherchons à identifier est en bonne santé ce qui nous le devis non aisément peut conduire à de véritables drames.

I .14 . 1 Les Matrices de Confusion

Afin d'obtenir les valeurs propres de chacunes de ces variables l'on procède par le calcule d'une **matrice de confusion** sur la base des valeurs obtenues suite à la prédiction (diagnostique du système) et des véritables résultats associés au inputs de ce derniers (celles-ci sont connues d'avance [voir supervised-learning]) ce type de matrice est notamment utilisé dans l'évaluation des système de classification (ce qui est le cas de notre modèle) plus concrètement la matrice sera composée d'autant de lignes que de classes gérées par notre système ou les éléments classés par ce derniers sont répartis en autant de colonnes que de classes ,ce qui donne $2^{nombre\ de\ classes}$ cellules en tout , où l'on auras le nombres d'éléments classés répartis selon leurs véritables classes .

par exemple : si nous prenons le cas d'une matrice de confusions réalisée pour 100 imagesde personnes saines ou malades où 50 sont celles de personnes malades et 50 autres sont celles de personnes saines et que l'on représente les résultats données par un système créé pour l'exemple selon le principe précédent on obtient

		classes estimées par le classeur	
		Malade	Sain
Classes Réelles	Malade	45 (VP)	5 (FP)
	Sain	1 (FN)	49 (VN)

I. 14 .2 Lois de calcul des mesures:

- la justesse (Accuracy) :

Cette mesures comme introduit précédemment permet en toute simplicité d'estimer le nombre de diagnostics corrects émis par le système , elle se calcule comme suit :

$$Accuracy = \frac{VP + VN}{VN + VP + FP + FN}$$

- La Sensibilité (Sensitivity) :

Cette mesure permet d'estimer l'exactitude des diagnostics de notre système(sa capacité à détecter les cas suspects) en se basant sur l'ensemble des vrais positifs et celui des faux négatifs ce qui correspond à l'ensemble des personnes diagnostiqué comme étant souffrantes

$$Sensitivity = \frac{VP}{VP + FN}$$

- La Spécificité (Specificity) :

Cette mesure permet d'estimer la capacité du système à indiquer les cas de personnes en bonne santé en se basant sur l'ensemble des vrais négatifs et celui des faux positifs qui correspond donc à toutes les personnes estimée comme en bonne santé par notre programme

$$Specificity = \frac{VN}{VN + FP}$$

Il est à noter que l'ensemble de valeurs de ces mesures varient entre le 0 et le 1 , et que de plus , quelles que soient la métrique considérée sa valeure optimale est de 1 .

Ainsi , on nous basant sur la définition de chacunes de ces métriques et au vu des objectifs de chacunes d'entre elle nous pouvons très rapidement voir que notre objectif lors du développement de notre solution sera idéalement d'atteindre la valeure de 1 pour chacune d'entre elles ; cependant , et comme expliqué ultérieurement et plus en détaille dans les paragraphes se plongeant plus en détaille dans les mécaniques interne de la solution que nous proposons , il nous faut considérer plus vraisemblablement des valeurs approchant du 1 tant convoité ainsi notre travaille serra du point de vu de ces mesures un travail d'optimisation.

Plus encore , en prenant en considération le paramètre de sensibilité qui correspond à la sensibilité de notre solution quand au dépistage de la pathologie traité par ce dernier , ainsi si elle est trop faible le nombre de faux négatifs augmente ce qui dans notre cas peut avoir des conséquences graves , ainsi ces personnes mal diagnostiquées risquent de voir leur accès aux soins retardé ce qui peut avoir des retombée fatale .

Conséquemment , nous tâcherons de nous focaliser sur la sensibilité du système afin de la faire tendre vers 1 .

I . 15. Conclusion

Dans ce qui a précédé nous avons eu le loisir d'introduire les réseaux de neurones dans un premier temps en expliquant l'origine de cette approche des intelligences artificielles ,puis dans un second temps, nous avons mis en évidence l'évolution de ces derniers de sorte à ce que l'on commence à parler de réseaux de neurones profond puis d'apprentissage profond ,pour enfin conclure sur la problématique traité par ce mémoire en présentant la pathologie que nous avons traité.

A la lumière de ces informations , nous avons décider de réaliser notre étude comparative visant à désigner la meilleure solution de deep learning pour ce type de problématique de fait le prochain chapitre est consacré à la conception de ces approches

Chapitre II Conception des différentes approches employées

Dans ce Chapitre nous nous attarderons sur la conception des différentes solutions que nous avons implémenté tout en rappelant le cycle que nous comptons suivre.

II.1 Introduction

Afin de pouvoir aborder notre projet de la meilleur façon possible, il nous faut revenir sur les étapes suivres et délimiter clairement le travail que nous souhaitons mettre en oeuvre ; nous reviendrons sur les différentes approches que nous avons implémenter et prendrons le temps de détailler les principes fondamentaux de ces dernières.

II .2 Plan de Travail:

Nous avons dans le cadre de ce mémoire souhaitez répondre à la problématique posée plus haut et pour y parvenir nous nous sommes tenu au cycles exposé précédemment (Chapitre I. partie 2)

Phase 1 : Cadrage :

Nos objectifs dans le cadre de ce projets sont de parvenir à affecter des étiquettes, permettant de distinguer une personne saine d'une personne souffrent de pneumonie, à des radiographies du thorax cela dans la cadre de la conception d'un outil d'aide au diagnostic médicale .

Nous avons comme précisé dans le chapitre consacré à la description de la pathologie que nous traitons pris le temps d'étudier l'ensemble des facteurs intervenant dans le diagnostic de cette maladie et on avons conclus que la radio du thorax véhicule des informations déterminantes dans l'établissement de ce derniers .

Dans un second temps , et au vue de la nature de la problématique abordé il nous a semblé pertinents de prendre en considération la précision et la sensibilité et la spécificité du modèle à produire

Phase 2 : données

- Le Set de Données :

Compte tenus de la nature de nos données à savoir des Radiographies du thorax il nous faut nous attarder plus amplement sur les propriétés de ce type de données ; dans le cadre de ce mémoire nous avons eu recours à l'un des data sets proposés par **CheXpert** est l'un des projet du **groupe Stanford ML** consacré au machine learning ,il s'agit d'une immense base de données regroupant une large collection de radiographies du thorax labellisées par des radiologues agréés servant de base de travail libre d'accès employée par les experts du

monde entiers souhaitant mesurer leurs modèles à ceux déjà proposé ainsi on retrouve sur le site officiel de cette plateforme le classement des modèles ayant les meilleurs scores de précision .

En ce qui nous concerne , nous avons employé une partie de leur data sets consacrée à la pneumonie d'origine bactérienne ou virale confondues , et l'avons employée comme base de notre travail .

- Format et aspect des données

L'ensembles des radios que comporte notre set de données est au format JPEG [Joint Photographic Experts Group] il s'agit d'une norme définissant le format d'enregistrement du d'une image numérique compressée par l'algorithme eponyme .

L'emplois de ce formats offre un certains nombres d'avantages dont le fait d'alléger la taille des fichiers images tout en veillant à conserver un maximum d'informations bien que l'une des plus grandes faiblesse de ce format reste la dégradation de cette information au fure et à mesure des modifications pouvant touché le fichier transformé .
En générale chaque pixel composant l'image est encodé sur 24 bits ou chaque unité de 8 bits (octets) permet de préciser la valeur de l'une couleurs des primordiales rouge (R) vert (G) bleu (B) du système de couleur RGB , où par conséquent , chaque valeure de chacunes des couleurs appartient à l'ensemble [0 - 255] .

- Aspect des images :

  Bien que nos images proviennent toutes du même dataset il est à noter que les dimensions de ces dernières est variables ainsi sur un ensemble de presque six milles images la taille de images varient de [500 / 1700 , 500 / 1700] ce qui de par la nature des réseaux de neurones qui assimilent chaque pixel de l'image comme une entrée unique ce qui pose bien évidemment un soucis à savoir que cette variation des dimensions modifie les nombre d'entrées perçue par le réseau de neurone ce qui peut fortement perturber le processus d'apprentissage du réseau, qui durant la phase de réajustement de poids propre à chacunes des entrées se retrouverai à éliminer des entrée pour devoir ensuite les réintégrer ralentissant voir rendant impossible la progression du modèle .

  De plus bien que l'ensemble des images fournis soit d'apparence des dégradés de gris l'encodage des images est toujours de type RGB ce qui implique nos images sont des tensor à 3 dimensions [500 - 1700 , 500 -1700 , 3] ce qui du point de vu de notre réseau de neurones à naître représente approximativement 3 630 000 entrée à gérer auxquelles il faut associer en première couche ,sans la prise en compte des suivantes ,autant de poids à gérer ce qui au vu de ces premiers calculs indique déjà un problème de complexité qu'il faudra s'atteler à corriger.

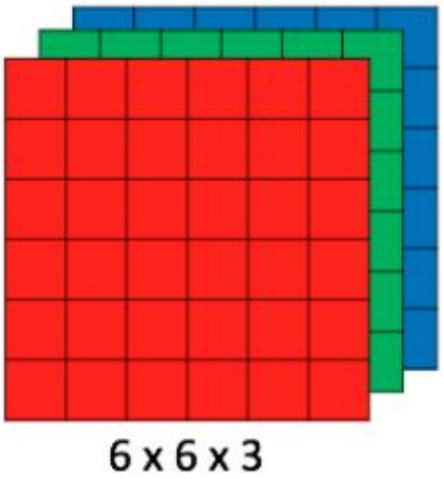

Figure II.1 : schéma d'une image RGB de dimension 6 par 6 [f10]

- Traitement de l'image :

  Face à la complexité induite par respectivement l'inconstance des dimensions de nos images mais aussi leur encodage au format RGB qui triple la quantité de données à analyser lors de la création de notre modèle ; le prétraitement de données qui par ailleurs est une phase clé du cycle de développement de modèles de deep learning , s'impose ; ce faisant nous serons amener à corriger les deux principales tares de notre dataset .

Pour ce faire nous aurons l'usage d'un certain nombre d'outils fournis par la bibliothèque **OpenCV .**

Face à la nature de notre projet nous remarquons que la phase de pré-traitement des données est communes à l'ensemble des approches que nous comptons réaliser , et que de fait hormis les spécifiée lié à l'architecture ou la mise en oeuvre nous n'aurons à traiter notre dataset qu'une seule fois

phase 3 : Construction des réseau de neurones

Compte tenu de la nature de la problématique abordée nous avons décidé d'employer plusieurs architecture et méthodes de deep learning adaptées à ce type de problématique dont nous comparerons dans la suite de ce documents les résultats obtenus par chacunes de ces solutions .

II .3 Les réseaux de neurones Convolutif

La première de ces architectures étant les réseaux de neurones convolutifs CNN réputés comme étant très efficaces pour les tâches de classification d'images dans ce qui suit nous tâcherons de les introduire plus en détaille.

Les réseaux de neurones convolutifs ont une méthodologie similaire à celle des méthodes traditionnelles d'apprentissage supervisé , ils reçoivent des images en entrée, détectent les *features* de chacune d'entre elles, puis entraînent un classifieur dessus. Cependant Les CNN apprennent les *features* de chaque image et il réalisent eux-mêmes tout le boulot d'extraction et description de *features* contrairement aux techniques d'apprentissage supervisé et C'est là que réside leur force les réseaux de neurones convolutifs apprennent les *features* de chaque image automatiquement De plus, l'architecture spécifique du réseau permet d'extraire des *features* de différentes complexités, des plus simples au plus sophistiquées.

Le premier bloc fait la particularité de ce type de réseaux de neurones, puisqu'il fonctionne comme un extracteur de *features*. Pour cela, il effectue du *template matching* en appliquant des opérations de filtrage par convolution. La première couche filtre l'image avec plusieurs noyaux de convolution, et renvoie des *"feature maps"*, qui sont ensuite normalisées (avec une fonction d'activation) et/ou redimensionnées [C7].

Ce procédé peut être réitéré plusieurs fois : on filtre les *features maps* obtenues avec de nouveaux noyaux, ce qui nous donne de nouvelles *features maps* à normaliser et redimensionner, et qu'on peut filtrer à nouveau, et ainsi de suite. Finalement, les valeurs des dernières *feature maps* sont concaténées dans un vecteur. Ce vecteur définit la sortie du premier bloc, et l'entrée du second.

Le second bloc n'est pas caractéristique d'un CNN : il se retrouve en fait à la fin de tous les réseaux de neurones utilisés pour la classification. Les valeurs du vecteur en entrée sont transformées (avec plusieurs combinaisons linéaires et fonctions d'activation) pour renvoyer un nouveau vecteur en sortie. Ce dernier vecteur contient autant d'éléments qu'il y a de classes ou l'élément représente la probabilité que l'image appartiennent à la classe. Chaque élément est donc compris entre 0 et 1, et la somme de tous vaut 1. Ces probabilités sont calculées par la dernière couche de ce bloc , qui utilise une fonction logistique (classification binaire) ou une fonction *softmax*(classification multi-classe) comme fonction d'activation.

Il existe quatre types de couches pour un réseau de neurones convolutif : la couche de convolution, la couche de *pooling,* la couche de correction ReLU et la couche *fully-connected*..

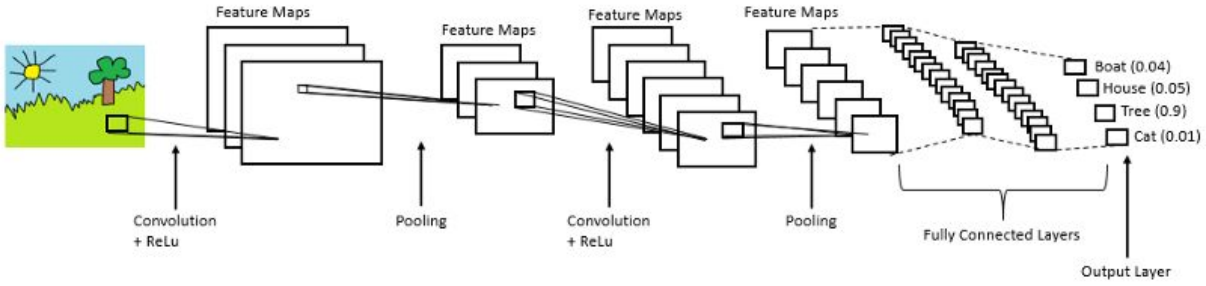

Figure **II**.2: Architecture d'un CNN [f11]

- **La couche de convolution :**

  La couche de convolution est la composante clé des réseaux de neurones convolutifs, le but de cette dernière est de repérer la présence d'un ensemble de *features* dans les images reçues en entrée. Pour cela, on réalise un filtrage par convolution ou on glisse une une fenêtre qui représente la *feature* sur l'image et de calculer le produit de convolution entre la *feature* et chaque portion de l'image balayée [C7].

  La couche de convolution reçoit donc en entrée plusieurs images, et calcule la convolution de chacune d'entre elles avec chaque filtre. Les filtres correspondent exactement aux *features* que l'on souhaite retrouver dans les images.

  On obtient pour chaque paire (image, filtre) une carte d'activation, ou *feature map*, qui nous indique où se situent les *features* dans l'image : plus la valeur est élevée, plus l'endroit correspondant dans l'image ressemble à la *feature*.

  On considère qu'on a une image dont les pixels d'image sont 0, 1 et une matrice de filtrage 3 x 3, Ensuite, la convolution de la matrice d'images 5 x 5 se multiplie avec la matrice de filtres 3 x 3,on obtient en sortie une matrice de «Feature Map» comme indiqué ci-dessous

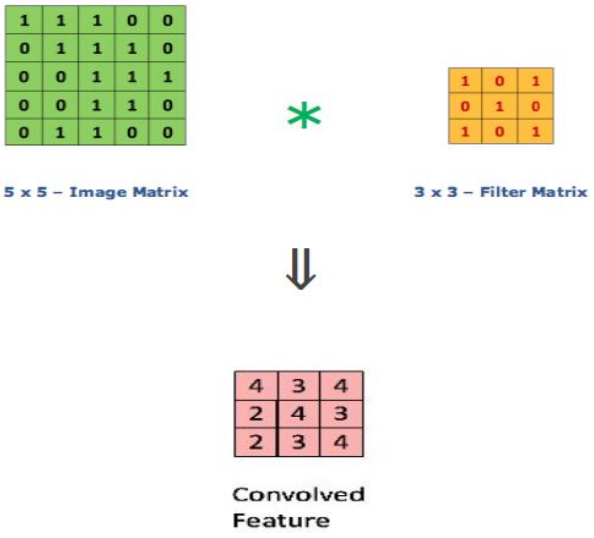

Figure II.3: 3 x 3 Output matrix [f11]

- **la couche de pooling :**

Ce type de couche est souvent placé entre deux couches de convolution : elle reçoit en entrée plusieurs *feature maps*, et applique à chacune d'entre elles l'opération de *pooling*. qui consiste à réduire la taille des images, tout en gardant leurs caractéristiques importantes.

Pour cela, on découpe l'image en cellules régulière, puis on garde au sein de chaque cellule la valeur maximale [C7]. En pratique, on utilise souvent des cellules carrées de petite taille pour ne pas perdre trop d'informations.

On obtient en sortie le même nombre de *feature maps* qu'en entrée, mais qui sont plus petites.Le but de l'opération de pooling est de réduire le nombre de paramètres et de calculs dans le réseau et d'améliorer ainsi l'efficacité du réseau.

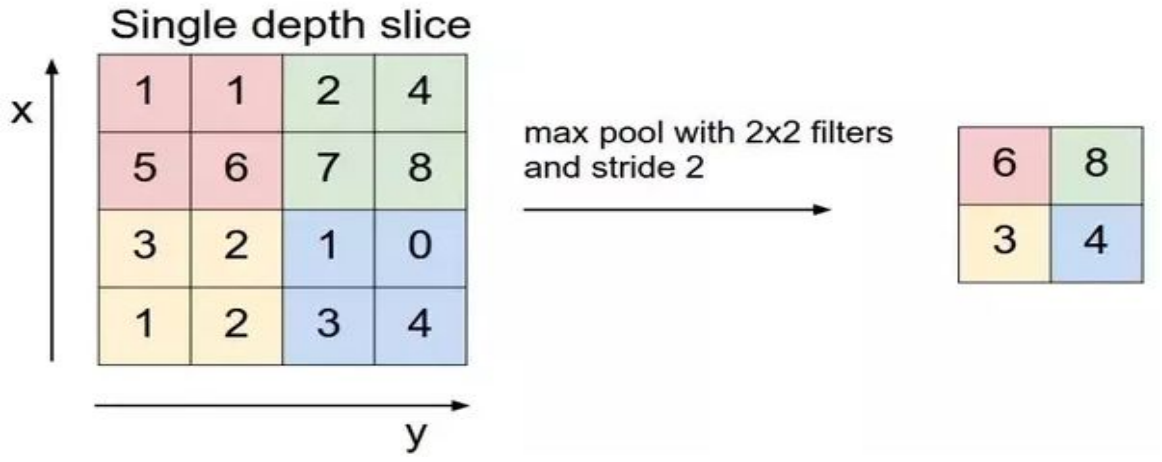

Figure II.4 : Max Pooling [f11]

- **la couche ReLU :**

ReLU (*Rectified Linear Units*) est une fonction réelle non-linéaire définie par $ReLU(x) = max\ (0, x)$

La couche de correction ReLU remplace donc toutes les valeurs négatives reçues en entrées par des zéros. Elle joue le rôle de fonction d'activation.

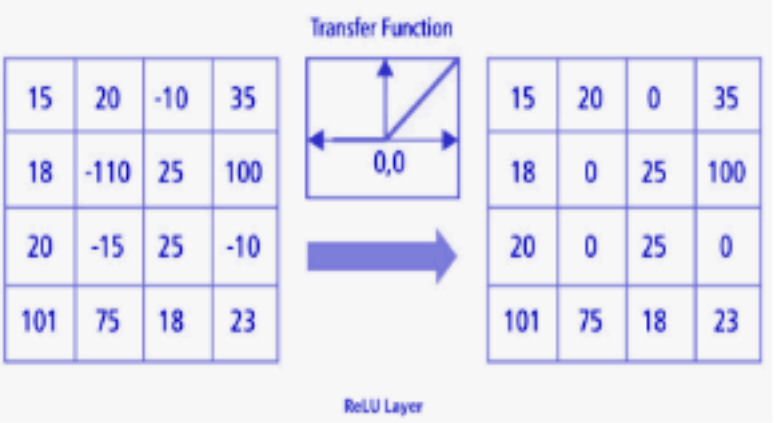

Figure II.5: opération ReLU [f11]

- **La couche *fully-connected* :**

  La couche *fully-connected* constitue toujours la dernière couche de tout les type de réseau de neurones

  Ce type de couche reçoit un vecteur en entrée et produit un nouveau vecteur en sortie. Pour cela, elle applique une combinaison linéaire puis éventuellement une fonction d'activation aux valeurs reçues en entrée [C7].

  La dernière couche *fully-connected* permet de classifier l'image en entrée du réseau : elle renvoie un vecteur de taille N qui représente le nombre de classes dans notre problème de classification d'images. Chaque élément du vecteur indique la probabilité pour l'image en entrée d'appartenir à une classe

  Pour calculer les probabilités, la couche *fully-connected* multiplie donc chaque élément en entrée par un poids qui est accordé par la couche *fully-connected* et qui dépend de l'élément du tableau et de la classe., ensuite elle fait la somme, puis applique une fonction d'activation

  Ce traitement revient à multiplier le vecteur en entrée par la matrice contenant les poids. Le fait que chaque valeur en entrée soit connectée avec toutes les valeurs en sortie explique le terme *fully-connected*.

[C7] :Pascal Monasse et kimia Nadjahi,lassifiez les images à l'aide de réseaux de neurones convolutifs;le lien : https://openclassrooms.com/fr/courses/4470531-classez-et-segmentez-des-donnees-visuelles/5082166-quest-ce-quun-reseau-de-neurones-convolutif-ou-cnn

II .4 Les Capsule networks

Dans un second temps nous avons eu l'idée d'employer une architecture plus récente celle des capsules networks que nous définissons comme suit

Les réseaux de neurones convolutifs ont connu un énorme succès dans la résolution des problèmes de reconnaissance et de classification des objets, cependant ils ont des inconvénients dans leur architecture de base, ce qui les empêche de très bien fonctionner pour certaines tâches.

Si un objet CNN est présenté dans une orientation inconnue ou dans laquelle des objets apparaissent à des endroits auxquels il n'est pas habitué, sa tâche de prédiction échouera probablement. Par exemple, si vous renversez un visage, le réseau ne sera plus en mesure de reconnaître les yeux, le nez, la bouche et la relation spatiale entre les deux. De même, si vous modifiez la position d'un des éléments de visage le réseau ne sera pas capable de reconnaître le visage parce que les CNN apprennent des modèles statistiques dans les images, mais ils n'apprennent pas les concepts fondamentaux sur ce qui donne l'impression que quelque chose ressemble à un visage.

Pour résoudre les problèmes liés aux réseaux de neuronnes convolutifs Geoffrey Hinton, le père de l'IA a donc décidé de développer une nouvelle architecture qui ne soit pas aussi dépendante de l'opération de pooling donc il a proposé les capsule networks comme solutions aux problèmes pour lesquels les modèles de réseaux de neurones convolutifs sont inadéquats.

Hinton et Sabour ont emprunté l'idée des neurosciences qui suggèrent que le cerveau est organisé en modules qui sont appelés capsules. Ces capsules sont particulièrement efficaces pour manipuler les caractéristiques des objets telles que la position, taille, orientation, etc. Selon eux, le cerveau doit disposer d'un mécanisme lui permettant d'acheminer les informations visuelles de bas niveau vers ce qu'il considère être la meilleure capsule pour les manipuler [C8].

Les capsules représentent les différentes caractéristiques d'une entité particulière présentes dans l'image ou la caractéristique la plus importante est l'existence de l'entité instanciée dans l'image. cette entité instanciée est un paramètre tel que la position, la taille, l'orientation, la déformation ,etc.

Pour représenter son existence il faut utiliser une unité logistique distincte, dont le résultat est la probabilité que l'entité existe et pour obtenir de meilleurs résultats que les CNN, nous devrions utiliser un mécanisme de routage par accord itératif. Ces fonctionnalités sont appelées paramètres d'instanciation.

Dans les réseaux de neurones on utilise les fonctions d'activation qui sont de simples opérations mathématiques appliquées à la sortie des couches elles agissent généralement sur les valeurs scalaires, par exemple, en normalisant chaque élément d'un vecteur de sorte qu'il se situe entre 0 et 1.

Par contre dans Capsule Networks, un type spécial de fonction d'activation, appelé fonction squash, est utilisé pour normaliser la magnitude des vecteurs plutôt que les éléments scalaires eux-mêmes.

Les sorties de ces fonctions squash nous indiquent comment acheminer les données à travers différentes capsules entraîné pour apprendre différents concepts. Les propriétés de chaque objet de l'image sont exprimées dans les vecteurs qui les acheminent. Par exemple, les activations d'un visage peuvent acheminer différentes parties d'une image vers des capsules comprenant les yeux, le nez, la bouche et les oreilles.

Un neurone traditionnel dans un réseau de neurones effectue les opérations scalaires suivantes: pondération des entrées,somme des entrées pondérées,non linéarité ces opérations sont légèrement modifiées dans les capsules et sont effectuées comme suit[C8]:

- Multiplication matricielle des vecteurs d'entrée avec des matrices de pondération. Ceci encode des relations très importantes entre les entités de bas niveau et les entités de haut niveau dans l'image.
- Vecteurs d'entrée de pondération ou les poids déterminent à quelle capsule de niveau supérieur la capsule actuelle enverra sa sortie. Cela se fait par un processus de routage dynamique
- Somme des vecteurs d'entrée pondérés cette opération ne change pas c’est la même dans le cas d’un neurone traditionnel.

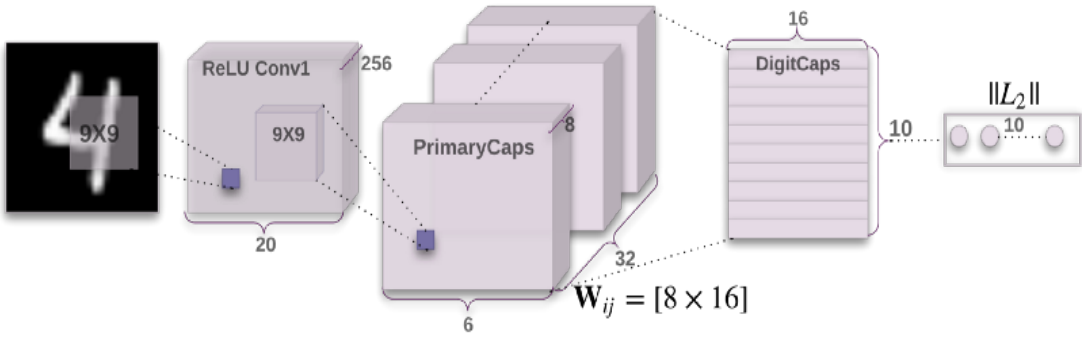

Figure II.6: Architecture des CapsNet [f12]

Il existe quatre types de couches pour un réseau de neurones convolutif : la couche de convolution, la couche de correction ReLU , la couche PrimaryCaps et la couche DigitCaps

- La couche de convolution:

  La première partie de CapsNet est une couche convolutionnelle traditionnelle.L'objectif est d'extraire de l'image d'entrée certaines caractéristiques extrêmement basiques, telles que des bords ou des courbes.

  Une convolution est faite en alignant une petite "fenêtre" dans le coin de l'image qui nous permet seulement de voir les pixels dans cette zone. Nous faisons ensuite glisser la fenêtre sur tous les pixels de l'image, en multipliant chaque pixel par un ensemble de poids, puis en additionnant toutes les valeurs figurant dans cette fenêtre. Cette fenêtre est une matrice de poids, appelée "noyau"

- La couche Primary Caps :

  Ou plus concrètement la couche subdivisée en capsule de laquelle cette architecture tient son nom, il s'agit d'un groupe de neurones capables d'analyser la détection de features mais aussi de gérer des informations relatives à celles-ci comme leur orientation leur tailles ,et qui de plus grâce au principe du *routing by agreement* ^peut estimer la probabilité qu'un objet (composé de feature plus basiques)soit ou non détecté. Son fonctionnement se résume comme suit les sorties de la couches de convolution sont divisées en autant de primary capsules qui pour chacune d'entre elles traiterons les tensor en entré en leur appliquant des kernels (filtre) pour produire en sortie un vecteur de sorties à 4 dimensions (les sorties de la couche de convolution

sont de dimension 4).

Ce vecteur est ensuite multiplié par une matrice de poids dit de prédiction dont le résultat permet de prévoir à lesquelles des capsules de la couches suivante (chargé de détecter des features plus complexe) la sortie de cette capsule doit être envoyée,les poids de cette matrice de prédictions se mettent alors à jour en augmentant le poid du liens propre à la connexion ayant donné le meilleur résultat et en abaissant le poid des liens entre les capsule de l'étage suivant et celle de la couche précédente ayant donné des moins bon résultats

- La couche DigitCaps:

  Il s'agit de la couche de capsule de niveau supérieure vers lesquelles les sorties de la couche précédentes sont dirigé grâce au procédé du routing by agreement ,cette couche rassemble alors les sortie des précedente et produit un vecteur à N dimensions (dépendant principalement de la taille des matrices de poids de prédiction)contenant l'ensemble des paramètres d'instentiations de l'images analysées ,il est alors possible de soit analysé ce vecteur afin de déterminer quel classe à été reconnue ou alors de soumettre cette sortie à une couche dite *Decoder* dont le rôle est de reconstruire les images ayant généré les paramètres d'instanciations en analysant le vecteur de sortie et on le comparant à l'image en entré cette couche étant principalement composée de couches de neurone densément connectés dans notre cas de figure nous n'aurons pas à exploiter la couche decoder

- La couche ReLU :

  ReLU est une fonction d'activation qui prend une valeur. Nous appliquons cette fonction à toutes les sorties de nos convolutions,si elle est négative, elle devient nulle et si elle est positive on garde sa valeur

$$X = max(0 , X)$$

- Routing by agreement:

  Ce processus consiste à décider des informations à envoyer au niveau suivant ou les capsules de niveau inférieur envoient leur entrée à des capsules de niveau supérieur . Dans les réseaux traditionnels, nous ferions probablement quelque chose comme le «pooling maximal qui permet de réduire la taille en ne transmettant que le pixel activé le plus élevé dans la région à la couche suivante [C8].

  Cependant, avec les réseaux de capsules, nous allons faire quelque chose qu'on appelle routing by agreement

  Avec le routing by agreement,, nous ne faisons que transmettre les informations utiles et jetons les données qui ne feraient qu'ajouter du bruit aux résultats. Cela nous donne une sélection beaucoup plus intelligente que de simplement choisir le plus grand nombre, comme dans le pooling maximum.

II .5 Le Transfer Learning:

L'idée derrière le *transfert learning* ou transfert de connaissances est d'aller plus loins que le simple apprentissage ou mimétisme mais de procurer à la machine la faculté de transformer des connaissances acquise précédemment dans un domaine afin de les adapter à la résolution d'une problématique issue d'un domaine proche de celui ayant servi à l'acquisition des compétences initiales (le transfert étant plus efficace si le nouveau domaine est sensiblement proche du premiers);un exemple illustrant ce type de transfert serrais la faculté d'un bon programmeur ,ayant assimilé les bases de l'algorithmique et des structures de données, à maîtriser très rapidement de nouveau langages de programmation reposants sur ce principes .

Jusqu'à il y a peu, la création d'un modèle de deep learning se faisait en isolant une problématique spécifique autour de laquelle on construisait notre solution; il fallait ainsi repartir de zéro à chaque nouveau projet ou dès que la nature du problème à traiter commencer à changer ou si il s'opérer une trop grande évolution dans les besoins auxquels il faut répondre c'est à partir de ce constat qu'a émergé l'idée du transfert de connaissances appliqué au *deep learning* .

Cette nouvelle approche du deep learning offre en effet et pour la première fois un espoir important de progresser vers la création de la première AGI ou intelligence artificielle générale selon certains des plus grand experts du domain à l'instar du Dr Andrew Ng qui déclare en 2016 que " le transfer learning pourrait être le prochain cheval de bataille économique des sociétés spécialisées dans le domaine des intelligences artificielles".

- **Comparaison entre le TL et le ML**

Il faut comprendre que l'entraînement d'un modèle de deep learning classique passe par une phase d'analyse du domaine aboutissant à la sélection d'un data set le plus riche possible pensé pour la résolution de la problématique à traiter afin de pouvoir aboutir à une solution.

La ou dans le cas du **TL** le processus globale est plus rapide car l'on se contente de récupérer un modèle déjà entraîné ayant accumulé un savoir que l'on va ensuite entraîner avec moins de données pour des résultat équivalents voir supérieures à ceux des méthodes traditionnelles

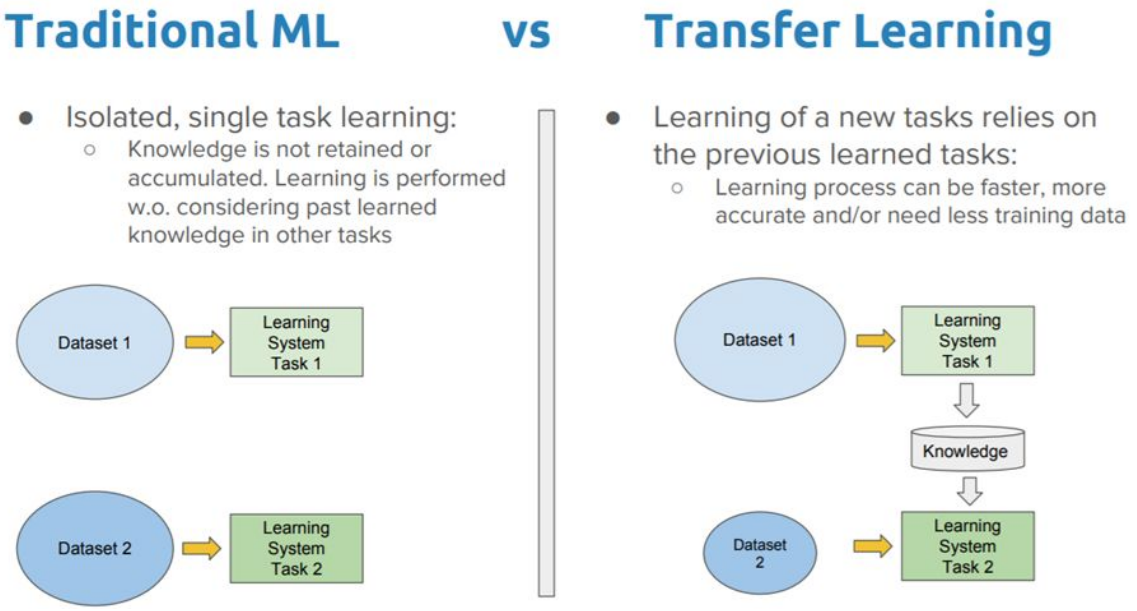

Figure II.7: illustration de la différence entre le ML et le TL [f13]

- **De façon plus formelle**

En nous basant sur les travaux de Pan et Yang [t1], nous reprenons les notations employées pour désigner respectivement le domaine d'application et la tâche correspondant dans le cas du *DL* au modèle employé pour résoudre la problématique propre au Domaine.

Ainsi un Domaine est définie comme étant un tuple composé de l'espace des *features* χ (propriétés et paternes propres aux données) et d'une marge de probabilité P(X) tel que X = $\{x_1, \dots, x_n\} \in \chi$ de fait nous notons $D = \{\chi, P(X)\}$

De plus *the task* ou tâche est représenté par un tuple composé d'un espace d'étiquettes noté γ (l*abel space*) et d'une fonction d'optimisation η (*objective function*) que nous pouvons exprimer comme suit $P(Y|X)$ (la probabilité d'associer une étiquette à un set de *features*) conséquemment nous pouvons noter
$T = \{\gamma, \eta\} = \{\gamma, P(Y|X)\}$ *tel que* $Y = \{y_1, \ldots, y_n\} \in \gamma$

En considérant les représentations que nous venons de définir il est alors possible d'exprimer les objectif du transfert de connaissance comme suit:

"Pour une tâche source T_S appliquée à un domaine initiale ou source D_S, et un domaine cible D_t auquel une tâche cible T_t est associée ;l'objectif du transfert de connaissance sera de nous permettre d'apprendre la distribution de probabilité de la tâche cible $P(Y_t|X_t)$ associée à D_t à partir des informations acquises depuis le domaine D_S et sa tâche T_S tel que $D_S \neq D_t$ et $T_S \neq T_t$; en ayant comme contrainte en générale un nombre d'éléments étiquetés exponentiellement plus petit que celui employé lors de la création de la tâche source."
Il s'offre alors à nous quatre scénarios dépendant d'un certain nombre de paramètres

- $\chi_S \neq \chi_t$ les features (propriétés des données) diffèrent d'un domaine à l'autre (par exemple des documents écrient dans deux langues différentes) ce cas de figure est nommé *cross-lingual adaptation*

- $P(X_S) \neq P(X_t)$ la marge de distribution de probabilité des deux domaines différent (par exemple avec le cas de deux document traitant de deux sujets différents) ce scénario est nommé *domain adaptation*

- $\gamma_S \neq \gamma_t$ l'espace des étiquette employé par les deux tâches à associer à leurs domaines respectifs divergent (par exemple avec le cas ou dans le domaine cible il faut assigner une autre étiquette au documents initialement étiqueté par la tâche source) en générale ce scénario s'accompagne du scénario 4

- $P(Y_S|X_S) \neq P(Y_t|X_t)$ la probabilité conditionnelle de distribution des deux tâches sont différentes (ceci peut correspondre par exemple au cas ou les documents des domaines sources et cible sont présent en quantitées déséquilibrés par rapport aux étiquettes auxquelles ils sont respectivement associés) ce cas est le plus courant (du fait de la présence de moin de données dans le cas du domaine cible)

[t1] Pan et Yang ASurvey on *Transfer Learning*
https://www.cse.ust.hk/~qyang/Docs/2009/tkde_transfer_learning.pdf

- **le *transfer learning* appliqué au *deep learning***

Lorsque appliqué au deep learning , il nous faut considérer la catégorie du transfert de connaissances dite *inductive learning* qui désigne son utilisation dans le cas ou le domaine source et le domaine cible sont les mêmes et ou les tâches associée aux deux sont quant à elles différentes , il s'agira alors d'essayer d'exploiter les paternes assimilé suite au traitement du domaine source afin d'aider au renforcement de la tâche cible .

Plus concrètement afin d'apprendre à reconnaître ces paternes que l'on désignera par biais inductifs il est possible d'intervenir en choisissant le domaine source le plus à même de contenir les paternes qui nous intéresse ou encore en optant pour l'utilisation d'un modèle dédié à une tâche similaire à celle ciblé.

De fait il existe un certain nombre de stratégies employées par les expert au cas par cas , nous énumérerons les plus populaires :

- Emplois de modèles d'extraction de features pré-entraîné

  L'architecture hiérarchique des modèle fait que ces derniers parviennent à reconnaître des features plus ou moins complexes selon que l'on progresse en profondeur dans les couches ou en générale la couche finale sert à générer une sortie calculées par rapport aux informations extraites par les couches antérieures .
  De fait il devient possible d'utiliser un modèle ,privé de sa couche de sortie, et entraîné à reconnaitres les paramètres qui nous intéresses (voir figure II)

  Cependant il faut prendre garde à ne mettre à jours que les poids des nouvelles couches sans altérer ceux du modèle extracteur .

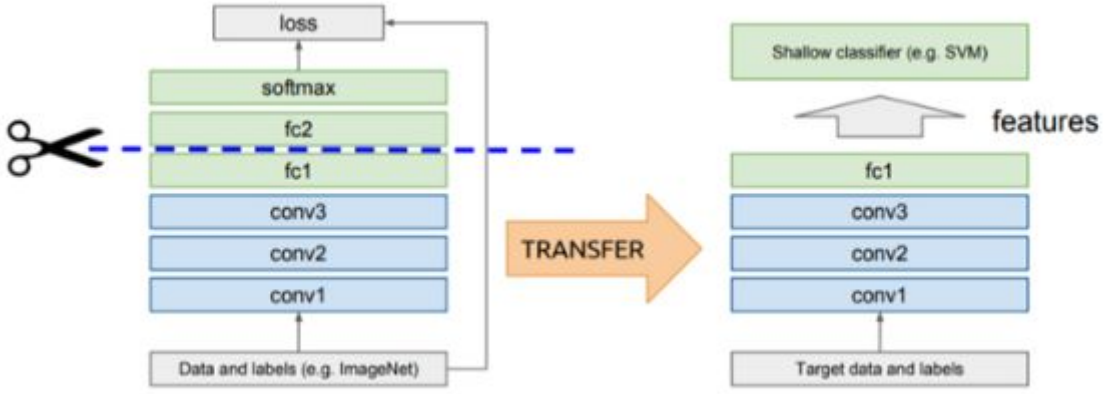

Figure II.8: utilisation d'un modèle pré-entraîné comme extracteur de features [f13]

- Emplois de modèle d'extraction de features pré-entraîné puis les calibrer

  Au delà de l'emplois d'une partie d'un modèle pré existant afin de bénéficier de sa faculté à reconnaître des paternes cette technique propose de recalculer de façon sélective les poids de certaines couches du modèle d'extraction en exploitant le fait que les feature détectée par les modèles du **DL** sont de plus en plus complexes de fait il devient possible de cibler certaines couches capable de détecter certaines feature plutôt que d'autres . (voir figure II.)

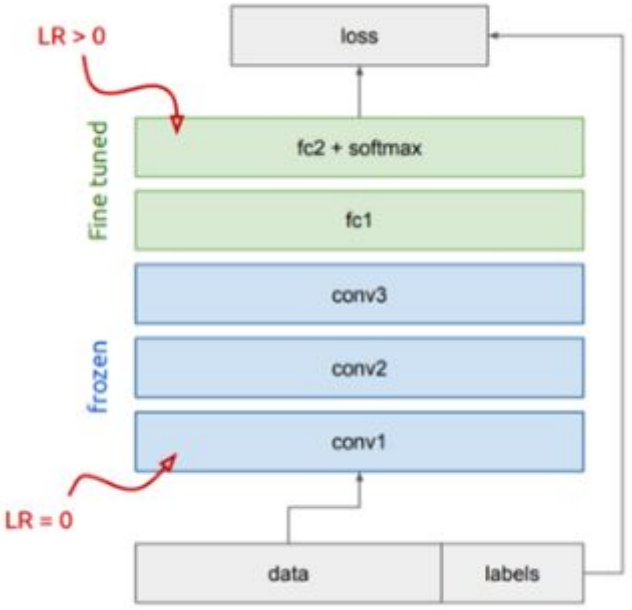

Figure II. 9: illustration d'un modèle d'extraction recalibré [f13]

De part l'implication de la communauté scientifiques mais aussi de la communauté de l'open-sources très tôt dans ce domaine la majorité des architecture les plus performantes dans le domaine sont mises à disposition de la communauté de façon libre nous pouvons ainsi citer les plus populaires

- Pour le traitement d'images

  - VGG-19
  - XCeption
  - ResNet-50

- Pour le traitement du langage naturel

  - Word2Vec
  - GloVe
  - FastText
  - BERT de Google

- **l’outil AutoML de Google**

Il s’agit du nouvel outil de la gamme de produits **Firebase** lié au service cloud de Google (**GCP**) de type **Baas** (Backend as a service : catégorie propre à Google permettant de fournir des outils permettant de créer un backend adapté au besoin des utilisateur et hébergé sur le cloud de google , le tout étant tarifé à l’utilisation bien que le service en question dispose d’une offre gratuite pour le moment); cet outil bien qu’en version bêta à l’heur où nous écrivons ces lignes [il à été lancé en mai de cette année] offre à ces utilisateurs la possibilité de créer et de déployer un modèle spécialisé dans la classification d’images (pour le moment) à partir d’images étiqueté par l’utilisateur .

Le modèle en question est implémenté en se basant sur les principes du transfert de connaissances en exploitant un modèle de classification d’images entraîné par les équipes de Google capable de reconnaître jusqu'à 400 catégories d’images différentes; pour le moment les modèles ainsi obtenu sont optimisé pour l'exécution dans un environnement mobile bien que la plateforme propose pour les modèles les plus complexes la possibilité de gérer les exécutions de requêtes sur le cloud.(voir figure II. illustrant l’interface de cet outil)

Bien que cet outils est l’un des premiers de son genre il n’est pas le seul de ce type en effet , l’outils Auto-Keras qui lui est open source se fixe comme objectif de permettre au personnes disposant de peu de connaissances techniques dans le domaine de l’apprentissage profond de parvenir à construire eux-même leurs propres modèls; cependant cet outils est encore au stade de bêta mais reste accessible aux personne intéressées.

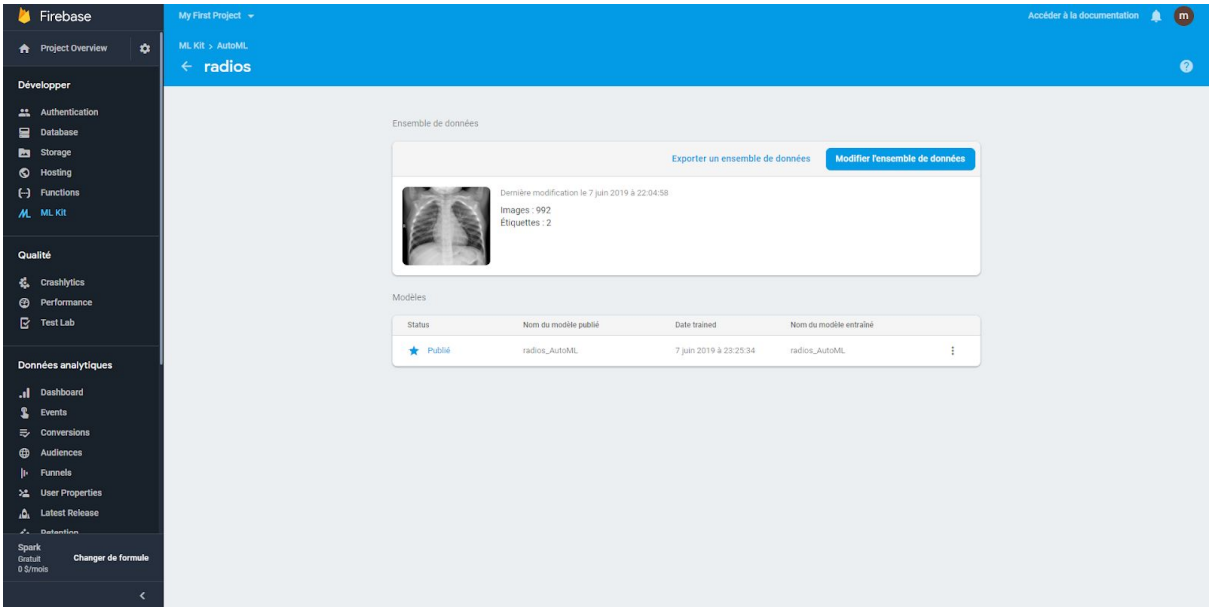

Figure II.10: interface utilisateur de l’outil AutoML

II.6 Conclusion

A la suite de notre présentation des différentes techniques possibles afin de résoudre la problématique initiale posée par notre mémoire à savoir “réaliser un outils d’aide au diagnostic de la pneumonie via la conception d’une solution de classification d’images médicales”, et face au floue apparent quand à la désignation de la meilleure solution Nous avons opter pour l’implémentation de l’ensemble de ces derniers afin par la suite de réaliser une étude comparatives entre ces dernières.

C’est ainsi que le chapitre suivant est consacré à l’implémentation de l’ensemble des ces architectures .

Chapitre III Réalisation

Dans ce Chapitre nous reviendrons sur le détaille de l'implémentations des différentes approches présenté dans le chapitre précédent ,nous verrons plus en détaille les outils et les différentes phases de la réalisation.

III.1 Introduction

Fort des enseignement de la partie conceptuelle de notre projet , nous avons entamé la phase d'implémentation de l'ensemble des solutions abordée plus en détaille précédemment.

En nous basant sur les observations réalisé au préalable dans la partie 2 du chapitre précédent nous entamerons la réalisation par la phase de prétraitement des données qui est communes à l'ensemble des approches pour ensuite revenir sur chacunes des approches séparément.

notons que par rapport au cycle de développement que nous suivant cette phase est équivalente à la phase trois

De plus n'ayant pas eu à retoucher à notre dataset du fait de sa qualité exceptionelle (classe équilibrée , suffisamment d'entrées) nous n'avons pas spécialement eu de phase d'expansion du modèle .

III .2 - Nos outils

Afin de mettre en oeuvre notre modèle il nous a fallu recourir à l'une des bibliothèques les plus populaire consacrée à la réalisation de projets Deep Learning à savoir Keras qui nous permet d'exploiter en toute simplicité la bibliothèque Tensor Flow de google ,celles-ci étant toutes deux des bibliothèques open source écrites en Python.

III . 2. 1 Python et ses librairies

Il est l'un des langages de programmations les plus populaires dans le domaine de la data science mais aussi du machine learning ; il est multi-paradigme ,ainsi il offre plusieurs approches de la programmation afin d'apporter une solution à un problème donné , de plus ,il est interprété ,c'est à dire non compilé et directement interprété par la machine à l'exécution ,et il favorise la programmation impérative structurée et l'orienté objet ,enfin il est multiplateforme employé aussi bien que dans des serveurs web sous linux que des objet intelligents en passant passant par un ordinateur de bureau sous windows .

Ce langage étant placé sous license libre il bénéficie plaine du soutiens de la communauté open source qui le maintient à jour et lui offre quotidiennement de nouveau packages et nouvelles bibliothèques ce qui lui permet d'être présent dans bon nombre de domaines que ce soit du calcule scientifique ou encore de la création d'application web ou même plus récemment le machine learning .

Durant l'implémentation de notre solution nous avons eu recours à un certain nombre de bibliothèques plus anecdotiques que celles présentées précédemment que nous tâcherons de définir plus en détaille dans ce qui suit :

- **Pickle**

  Il s'agit d'un module permettant de gérer la sérialisation et la désérialisation binaire d'objets python ce qui se traduit par la conversion d'un objet (en générale une structure de données) en un flux d'octets stocké sous la forme d'un fichier .pickle et le processu inverse consistant en la création d'un objet python à partir du fichier obtenu précédemment .

  comparaison d'autres outils de la même catégorie

- Json :
  Les données sont stockées sous forme de text encodé sous UTF-8 la ou pickle les stocke sous forme binaire
  Json ne peut permettre de représenter qu'un nombre limité d'objet python selon leurs types la ou pickle les prend tous en charge
  Les fichiers Json sont lisibles par un humain la ou ceux sous pickle ne le sont pas
  Json est un format non exclusif au langage python

- marshal :
  Pickle garde en mémoire les références des objets sérialisées de sorte à ne pas créer de duplicatas lors de cette opération la ou marshal ne le fait pas
  Marshal ne permet de sérialiser des objets issues de classes créées par l'utilisateur
  Les format marshal n'est pas portable d'une version de python à une autre au contraire de pickle

- **OS**

Il s'agit d'un module permettant d'exécuter des instruction de type ligne de commande shell ou CMD et d'avoir accès à l'ensemble des outils offerts par les interface CLI des systémes d'exploitation

III . 2. 2 Le Framwork Tensorflow

Il s'agit de la libraire open source de Google compatible avec python (bien que des portages sous d'autres langages de programmation existes tel que java ou encore C++) , consacrée au machine learning et plus précisément au calcule et à l'optimisation des opérations complexes ce qui permet d'accélérer le processus d'apprentissage de nos modèle.
En outre , elle a permis de grandement simplifier la partie gestion de hardware et code des

projets ML en servant d'interface simple d'utilisation entre les deux à noter qu'elle est disponible sous plusieurs versions optimisées pour l'emplois d'un GPU , un TPU (*tensor processe unite* , qui est un processeur modifié et optimiser pour les calcules sur les tensors et est une technologie propriétaire de google qui reste accessible via le service cloud de ces derniers) ou encore un simple CPU (peu conseillé car peu performant),la force de cette outils est sa capacité à simplifier la prise en charge de grandes quantitées de calcules ce qui en fait un très puissant moteur d'exécution exploitable lors de la réalisation de modèles de deep learning par exemple) .Il est optimisé pour la gestion des opérations relatives au réseaux de neurones.

- **son fonctionnement**

  Tensorflow permet aux développeurs de créer "des graphes de flux de données structurées" servant à décrire les mouvements des données à travers un graphe ou plus exactement à travers une série de noeuds, qui dans notre cas sont des opérations mathématique à appliquer sur ces données , chaque noeud recevant en entré des données arrivant des arc entrant sous forme de vecteurs multidimensionnels connus le nom de tensors (d'où le nom de la bibliothèque tensor-flow (flux de tensors)) .

  L'utilisateur ayant la capacité d'utiliser cette outil via des du code python, ainsi , les noeuds du graphes deviennent des objets sous python ,de plus , les applications TensorFlow deviennent des application python .
  Il est à important de noter que bien que la construction de ce graphe se fasse à l'aide du langage python , l'exécution des opération mathématiques (noeud) se fait sous C++ pour des raison de performances ; ainsi le scripte que vous créez ne sert qu'à déclarer la structure et gérer le transite des données en son sein .

  Enfin l'une des forces de TensorFlow est de permettre l'exécution des modèles sur de nombreux supports matériels que ce soit localement ,un cluster de machines voir même des mobiles .

- **Avantages de tensorflow**

- Offre un haut niveau d'abstraction permettant à l'utilisateur de se focaliser sur l'architecture de son modèle sans se soucier de l'implémentation des algorithmes de bases récurrents dans ce genre de projets ou encore de l'acheminement des données ou de l'exécution des opérations mathématiques .
- Il offre un grand nombres d'outils d'optimisation simple d'utilisation permettant d'évaluer et de modifier les opérations exécutées dans le graphe que vous avez créé et vous permet de le faire à l'aide d'une interface graphique simple d'utilisation "TensorBoard".
- Le soutien apporté à la fois par le communauté du fait du statut open source de ce projet mais aussi du soutien technique offert par Google ayant largement contribué à le rendre portatif (passage au mobile et IOTs) entre-autres.

Notez que tensorflow bien qu'étant penser pour être multi plateforme semble présenter des variations de résultats selon les plateformes que l'on attribue aux dépenses liés à des fonction aléatoires ou encore à la gestion des GPUs qui s'exécutent différemment d'un système / hardware à un autre

- **Comparaison à d'autres outils similaires**

Tensorflow est certes le plus populaire des frameworks dédiés au machine learning mais il reste confronté à la compétition d'autres outils tels que :

- PyTorch :

  Conçue entièrement sous python , il est très similaire à tensorflow ; il offre un nombre conséquent de composants optimisé pour le hardwar afin de réduire le temps d'exécution des différentes opération mathématique , il reste très performant pour la réalisation d'applications de ML de petite à moyenne ampleure mais on lui préfère tensorflow pour des projet plus conséquents.

- CNTK :

  Microsoft Cognitive Toolkit pensé principalement pour aider à la création de réseau de neurones profond , il suit la même logique que tensorflow quand à la représentation des données et opération sous forme de graphe orientés , de plus il permet de réaliser en toute simplicité des combinaisons entre les types de modèles les plus courants dans le domaine, il disponible sous forme de bibliothèque python , C# ou encore C++ à inclure à votre projet ou peut servir à la création de modèle de deep learning directement via son outil de description de modèle *BrainScripts* auquel on reproche un manque de lisibilité et une difficulté d'utilisation pour les novices .

- Apache MXNet :

  Est le framework de deep learning adopté par amazon et l'un des efficaces en terme de scalabilité matériel ou il permet d'exploiter au mieux un maximum de ressources tel que des GPUs ou des CPUs, il est l'outils proposé par défaut sur la plateforme cloud AWS il est basé sur du C++ .

III . 2. 3 l'API Keras

Il s'agit d'une API (application programmable interface) haute niveau dédiées au réseaux de neurones et écrite en Python , elle capable de s'exécuter en tant que couche supérieure de tensorflow,CNTK ou encore Theano , et permet de simplifier l'utilisation des fonctionnalitées offertes par ces derniers et d'accélérer et de simplifier le développement d'application de DL.

- **Ce qu'il offre aux utilisateurs**

- Simplifier et accélérer le prototypage à l'aide de ses commandes simplifiés , sa modularité et son extensibilité.

- Vous permet de trés facilement d'implémenter des réseaux de neurones convolutifs (CNN) ou des réseaux de neurones récurrents (RNN) ou une combinaison des deux

- S'exécute aussi bien sur des CPUs que des GPUs

- **Ses points fort**

- Keras est une API pensée pour être la plus simple d'utilisation possible en offrant à ces utilisateur des méthodes simple d'utilisation via des objets préalablement offerts à l'utilisateur

- Elle est principalement pensée pour une utilisation faite par des humains et propose donc un excellente documentation et un nombre impressionnants de messages d'erreurs très explicites et précis simplifiant les opérations de débogage et la correction d'erreures.

- Sa simplicité apparente n'indique pourtant pas un compromis avec la performance qui reste prioritaire et prise en charge par sa forte compatibilité avec les couches inférieures qu'elle emplois notamment tensorflow

- keras reste très flexible et permet d'implémenter toutes sortes de modèle notamment lorsqu'elle est intégrée un bibliothéque de deep learning bas niveau comme tensorflow

- Sa communauté qui compte un peu plus de 250 000 utilisateurs (en 2018) cette API est aussi bien présente dans l'industrie (Netflix , Uber , Yelp, Square) que dans la recherche ou elle jouit d'un soutiens et d'un nombre de contributions conséquents

- **Utilisation**

Keras fournis à ses utilisateurs un nombre de classes de bases qu'il peuvent utiliser afin d'implémenter leurs modèles , il en propose deux grandes catégories les modèles séquentiels et les modèles fonctionnels , ceux-ci se partagent un certains nombre d'attributs et de méthodes en communs dont les plus importants sont :

- L'attribut Layers (couches) :
  Il s'agit d'une liste de couches composant le modèle que nous construisons.

- L'attribut Input (entrées) :
  Il s'agit de la liste des entrées sous forme de tensors reçues par notre modèle.

- L'attribut Output (Sortie):
  Il s'agit de la liste des tensors obtenus en sortie de notre modèle

- La méthode get_weights() :
  Qui permet de récupérer un tensor contenant l'ensemble des poids des valeurs des poids de notre modèle

- La méthode set_weights(w):
  Qui permet d'attribuer aux poids du modèle les valeurs du tensor w passé en entrée

- La méthode save_weights(file_path)/load_weights(file_path) :
  Qui permet de sauvegarder/charger les valeurs des poids de notre modèle sous forme/à partir de fichiers HDF5

- Les méthode to_json()/to_yaml():
  Permettant de retourner une représentation de l'architecture de notre modèle sans les valeurs des poids sous forme de fichier json ou yaml

III . 2. 3. 1 - Sequential models :

Ces modèles vous permettent d'implémenter vous mêmes le vôtre basé sur une architecture en couches successives mono directionnelles , ou chaque couche reçoit les résultats du traitement effectué par la couche que la précède et envois le résultat des ses propres traitements à la couche qui lui succède à l'exception de la couche d'entré et de sortie qui interagissent avec le milieu extérieure .

III . 2. 3. 2 - Functional API :

Ce type de modèle se différencie du précédent on vous offrent une approche plus flexible que la précédente afin de construire votre modèle en vous offrant la possibilité de créez de modèle ayant plusieurs inputs ou/et plusieurs outputs mais aussi de créer des modèles se partageants de couches de neurones vous permettant ainsi de créer des réseaux beaucoups plus complexes

Pour ce faire il vous permet de déclarer un couples de couches et de lier entre elles sans prendre dans l'ordre que vous souhaitez avant de les intégrer à votre modèle

Keras vous offre aussi quelques soit le type de modèles que vous employez des outils vous permettant de customiser au maximum les paramètres d'entraînements de ceux-ci,via la Méthode Compile qui prend en paramètres notamment :

- Un Optimizer :

  Il s'agit de la fonction mathématique à employer lors du processus dit de back propagation permettant de recalculer les valeurs des différents poids de notre modèle , en générale il s'agit de variantes de la descente de gradient adapté au problème auquel nous faisons face [voir chapitre I : BackPropagation - Optimizer].

- La fonction de calcule d'objectif (loss function):

  Il s'agit de la fonction que nous employons afin d'estimer l'écart entre les résultats renvoyés par notre modèle et ceux attendu l'objectif de la phase d'entraînement étant de réduire cet écart
  Notez qu'il en existe un certain nombre adapté à plusieurs type de problématiques

- Les métriques à surveiller durant l'entraînement

  Il s'agit de l'ensemble des résultats d' évaluation de notre modèle que nous souhaitons suivre afin de gérer au mieux la progression de notre modèle .

Par la suite il vous suffira de préciser la durée de l'entraînement de votre modèle ainsi configurer via la méthode fit() qui prend les arguments suivant:

- X : à savoir les données non étiquetées sur lesquelles le modèle cherchera identifier des patterns

- Y : les étiquettes de celles -ci qui peuvent être unique pour chaque entré ou un vecteur

- *Batch-size* : est un entier qui nous permet de spécifier le nombres d'entrées à traiter avant d'optimiser les valeurs des poids de notre modèle

- *epoch* : un entier indiquant le nombre de fois que la totalité du data set serra parcourus

- *callbacks* : permet de spécifier une liste d'actions à effectuer durant l'entraînement comme des contrôles ou encore des arrêts si certaines conditions sont atteintes

- *validation_split* : compris en 0 et 1 il indique le pourcentage du data set que nous souhaitons mettre de côté à chaque epoch afin de s'en servir comme set de test et non d'entrainement (on veille à ce qu'elle soit nettement plus faible que celle consacré à l'entraînement)

Une fois le modèle entraîné il ne vous reste qu'a l'évaluer par rapport à n set de données de test via la fonction evaluate() qui vous retournera la valeure de la fonction objectif et la précision de votre modèle .

Ces étapes accomplies ,si vous êtes satisfait des résultat de votre modèle vous pourrez l'exploiter via la méthode predict() qui prend une données non labellisée en entrée pour vous retourner des prédictions associées

III . 2. 4.Jupyter Notebook

Il s'agit d'un IDE (Integrated development environment) appartenant au projet Open-source Jupyter compatible avec plus d'une douzaines de langages de programmation, s'exécutant sous forme d'application web (sur un navigateur de votre choix) vous permettant de créer et de partager des documents contenant du code exécutable , des équations mathématiques ,des graphiques et images en toute simplicité ; il est destiné à toutes sorte d'utilisateurs allant de l'expert en préparation de données aux expert en statistiques en passant par les développeurs de solution machine learning .

Avantages de l'outils :

- La prise en charge de plus de 40 langages de programmation parmis les langages de programmation dont R, Julia , Scala et Python.

- La possibilité de partager votre code en toute simplicité par mail , DropBox , Jupyter Notebook Viewer ou encore Github .

- Utilisation des notations markdown permettant d'organiser et de commenter le code

- La simplification de la gestion et de l'affichage des output multimédia de votre code tel que dh HTML , des images , vidéos , LaTeX ou même des sorties personnalisées .

- Intégration d'outils pour gérer les big data tel que Apache Spark ou les bibliothèques Pandas , scikit learn , ggplot2 pour python , R ou scala

- L'outil se base sur les standards de la communauté open source que ce soit pour les formats de documents , l'exécution du code ou encore l'exécution de commandes shell .

- Il est utilisées par des grand noms de l'industrie informatique tel que Google , Microsoft, IBM ou même des universités comme Berkeley, Clemson , Michigan State univ voir des agences gouvernementales tel que la NASA .

III . 2. 5 Collaboratory

Collaboratory est un outils basé sur Jupyter notebook ne nécessitant aucune configuration car s'éxécutant entièrement sur la cloud de Google , ce qui permet d'exécuter son code partout et en tout simplicité, de plus cet outils est gratuit et accessible à partir de n'importe quel navigateur .

Ce qu'il offre :

- Un accès limité gratuit au ressources matériel du cloud de Google à savoir des CPUs (un coeur d'un processeur Xeon cadencé à 2.3 GHz disposant de 2 thread) , GPUs (un GPU de type Tesla V9 disposant de 12 Go de mémoire et plus de 2400 Coeur Cuda)mais aussi des TPU .

- Un moyen simple de partager et de récupérer du code voir de travailler en groupe sur un même code et de stocker/charger au besoin vos fichiers stockés dans GDrive.

- Un environnement d'exécution sous python (2 ou 3) ou Swift prêt à l'emplois sans configurations supplémentaire à fournir

- Ajout de bibliothèques tierces permettant de visualiser les données

- Une machine virtuelle propre à chaque utilisateur disposant de capacitées de stockage de plus de 30 Go .

- Une compatibilité avec les formats de jupyter notebook .

- SEEDBank une banque de template de code et de tutoriel exécutable directement depuis votre navigateur.

- Un thème graphique obscure disponible de base .

III .2 .6 OpenCV

Fondée en 1999 il s'agit d'une large bibliothèque d'algorithmes et d'outils dédiés au traitement d'images pour la vision par ordinateur ou encore le machine learning disponible dans une large variété de langages de programmation allant du C++ et allant jusqu'au javaScript en passant par Python , les outils qu'elle propose s'exécutent sans soucis sur un large panel d'environnement que ce soit windows ou linux ;plus récemment OpenCV intégre les mécaniques des GPUs notamment de celles de l'architecture CUDA de NVidia dans l'exécution de ces programmes ce qui lui permet d'accélérer ses traitements .

Nous aurons l'usage de principalement deux des algorithmes offert par cet outils à savoir :

A - La fonction de transformation géométrique des images (Scaling):

Celle-ci invocable par l'appel de *cv.resize()* permet en proposant une image de dimensions X_a , Y_a d'obtenir en sortie une image de dimensions X_b , Y_b précisées en entrée

Il s'opérera par la suite une opération de redimensionnement ou seront regroupées des groupes de pixels adjacents de sorte à réduire la taille de l'image à la condition que $X_a \geq X_b$ *et* $Y_a \geq Y_b$.

Ou à l'inverse un dédoublement de pixels au besoin jusqu'à l'obtention de la taille voulue

Figure III .1: exemple de Scaling

B - La fonction de conversion en niveaux de gris (Grey Scaling)

Elle nous permettra de réduire la complexité des données que nous aurons à traiter de près de 3 fois il suffit de préciser l'image à traiter ainsi que le répertoire de stockage du résultat en entrée de la fonction appelée par *cv.cvtColor()*

Figure III .2 Exemple de Grey Scaling

En appliquant successivement les fonction A et B sur notre set d'images nous parvenons à fixer leurs dimension de sorte à en faire un tenseur de forme [X , X , 1] ou nous fixions une valeur de X jugée convenable nous permettant de réduire le nombre de données à prendre en compte sans pour autant dégrader l'image l'origine de façon trop importante.

Notez que le choix d'une forme d'image carrée n'est pas anodin car ceci simplifie un certains nombre d'opérations futures à réaliser sur ces dernières lors du traitement par notre réseau d'architecture CNN

III .2 .7 Tensorboard

Il s'agit d'un outil offert par tenserflow et dédié au monitoring et la surveillance , il permet entre autre de construire le graphes des opérations lié à un réseau de neurones **,** de suivre l'évolution de paramètres choisis au préalable , d'analyser les composant du réseaux et de déterminer leur compatibilité avec les TPU et d'aider à l'optimisation des réseaux implémenté

III . 3 Implémentation

Dans ce qui suit nous suivrons étape par étapes la mise en place de chaque composant de notre script finale

III .3 . 1 Traitement du dataset

Dans un premier temps nous nous sommes intéressés à notre dataset composé de près de six milles images de bonne résolution mais de tailles différentes et encodé pour disposer de trois couleures ce qui nous pousse à les redimensionner et colorisé en niveaux de gris à l'aide d'OpenCV avant de les étiqueter.

Pour cela nous créons un script python chargé d'accéder aux répertoirs contenant les images séparées en deux groupes un répertoire contenant les radios de personnes saines et un second contenant celles des personnes diagnostiquées comme étant malade ; puis de réaliser trois opération en parcourant les deux répertoire :

- Coloriser les images en niveaux de gris
- Redimensionner les images de sorte à en faire des carré de taille (175 , 175) [pour rappel ceci permet de réduire la puissance de calcule nécessaire au traitement des images par la suite]
- Ajouter l'images traité ainsi qu'une étiquette généré par rapport au répertoire source de l'images

Nous nous retrouvons ainsi en sortie avec une matrice de dimension (nbr_images , 2) ou la première colonne contient nos images et ou la seconde contient leur étiquette associée .

Suite à cela nous procédons au mélange des élément de la liste en conservant les paires intactes ceci a pour but d'éviter au réseaux les problèmes liés à un entraînement déséquilibré ou le modèle s'adapte à une classe puis à une autre ce qui à tendance à générer des résultats calamiteux .

Le vecteur des étiquettes sera ensuite séparer de celui des images afin de pouvoir les utiliser lors de l'entraînement des modèles

Afin de pouvoir exploiter nos images il faut encore les redimensionner pour passer du vecteur d'images vers un tensor à l'aide la bibliothèque numpy ou l'on précisera quatres

dimensions

- La première de taille 1 servira à stocker l'ordre de l'images lorsqu'elles seront soumises au réseau de neurones au traver de batch (portions du dataset)
- La seconde sera de taille 175 ce qui correspond à la largeur de l'image
- De même la troisième de taille 175 correspond à la hauteur
- Enfin la dernière dimension correspond au nombres de dimensions de couleurs de l'images ici 1 après pré-traitement.

A la fin de cette étape l'ensemble des vecteurs et tenseurs ainsi obtenus sont convertis en fichiers binaires et enregistrés grâce à la bibliothèque pickle. Il est à noter que ces opérations sont réalisées en deux fois un premières pour les données d'entraînement puis une secondes pour celle de tests.

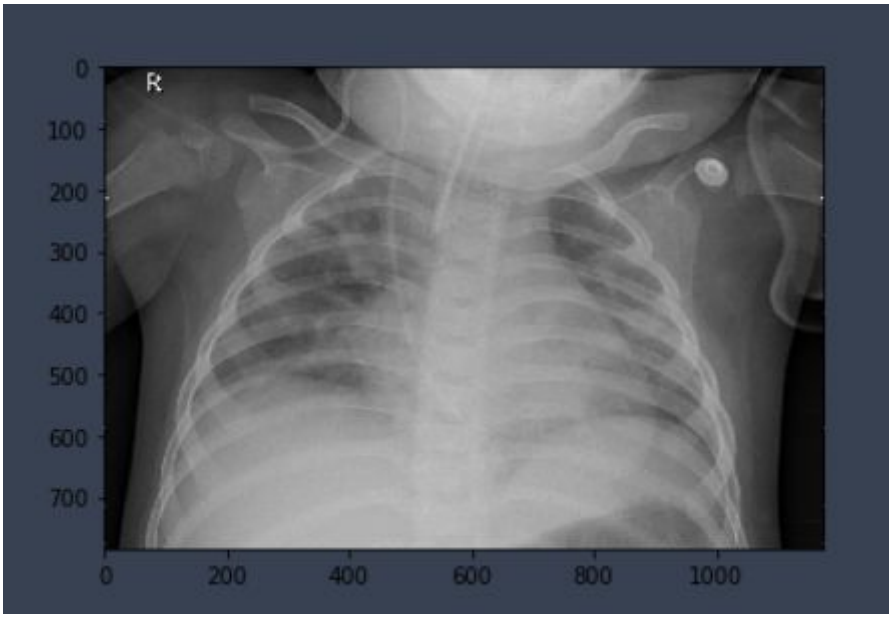

Figure III.3 image du dataset avant traitement

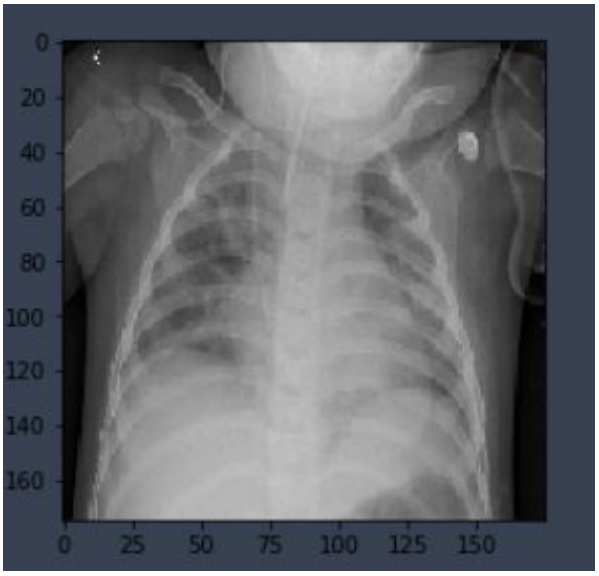

Figure III.4 image du dataset après traitement

III .3 . 2 Implémentation du CNN

Afin d'implémenter notre propre réseaux de neurones employant des couches convolutives nous nous sommes basées sur les propriétés de ce type de réseaux décrites dans le chapitre précédent , ainsi , nous avons procéder comme suit:

Première nous avons importer l'ensemble des bibliothèques nécessaires à la création de notre réseau à savoir tensorflow , keras , pickle et tensorbord (qui lui servira au monitoring du modèle)

Nous commençant par importer nos données prétraitées grâce à pickle , puis en raison de problèmes de performances procédant à la division normative du vecteur d'images par 255 afin de n'avoir à travailler qu'avec des valeurs allant de 0 à 1.

Ensuite, nous déclarons deux callbacks (ou conditions d'exécutions) le priers étant tensrboard qui nous permettra par la suite de suivre l'évolution des paramètres de notre choix durant l'entraînement du modèle le second lui est un callback chargé de sauvegarder les poids mis à jour durant cette session d'entraînement afin de nous permettre de sauvegarder le modèle obtenue en fin de procédure;

Nous passons ensuite à la phase de construction du modèle à proprement parler il est possible de voir l'architecture finale obtenue en se référant la figure III.5 obtenue grâce à tensorboard.

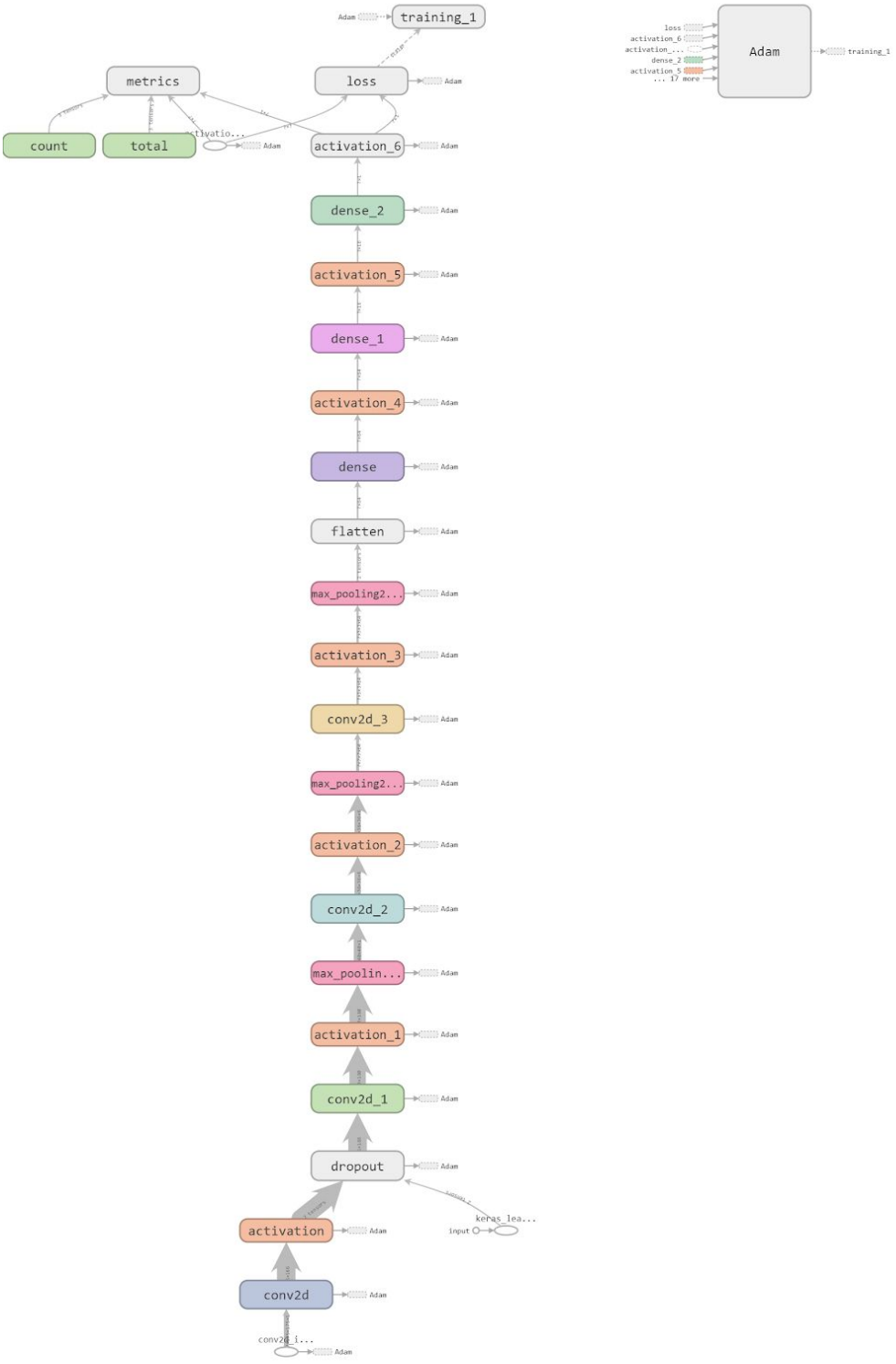

Figure III.5: Graphe du réseaux de neurones CNN

Afin d'implémenter l'architecture ainsi décrite , il vous ai possible de vous référer à notre code consacré à la déclaration des couches de notre réseau le dernier appel fonction quand à lui permet d'instancier les paramètres d'entraînement du réseau

```
[ ]  model = Sequential()
     model.add(Conv2D(256, (3, 3), input_shape= images_train.shape[1:]) )
     model.add(Activation("relu"))
     model.add(Dropout(rate = 0.3))
     model.add(Conv2D(128, (3,3)))
     model.add(Activation("relu"))
     model.add(MaxPooling2D(pool_size = (4,4)))
     model.add(Conv2D(64, (5,5)))
     model.add(Activation("relu"))
     model.add(MaxPooling2D(pool_size = (5,5)))
     model.add(Conv2D(64, (5,5)))
     model.add(Activation("relu"))
     model.add(MaxPooling2D(pool_size = (3,3)))
     model.add(Flatten())
     model.add(Dense(64))
     model.add(Activation("relu"))
     model.add(Dense(16))
     model.add(Activation("relu"))
     model.add(Dense(1))
     model.add(Activation("sigmoid"))
     model.compile(loss      = "mean_squared_error",
                   optimizer = "adam",
                   metrics   = ['accuracy'])
```

A la suite de l'implémentation de ce réseau , il nous a fallus paramétrer la fonction d'optimisation (variante de la descente de gradient) utilisé lors de l'entraînement du modèle nous avons opter pour la plus populaire "Adam" , le second paramètre à régler étant la fonction de coûts nous optons pour la fonction *Mean Square Error* introduite au chapitre I enfin nous précisons quel metrics souhaitons suivre lors de l'entraînement nous optons pour la justesse (accuracy) .

Nous lançons ensuite une estimation du nombre de paramètres entendables de notre modèle afin de mieux visualiser la complexité de ce dernier (voir figure III.5).l'on peut y voir un totale approchant des 2 millions de variables.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 166, 166, 256)	25856
activation (Activation)	(None, 166, 166, 256)	0
dropout (Dropout)	(None, 166, 166, 256)	0
conv2d_1 (Conv2D)	(None, 160, 160, 128)	1605760
activation_1 (Activation)	(None, 160, 160, 128)	0
max_pooling2d (MaxPooling2D)	(None, 40, 40, 128)	0
conv2d_2 (Conv2D)	(None, 36, 36, 64)	204864
activation_2 (Activation)	(None, 36, 36, 64)	0
max_pooling2d_1 (MaxPooling2	(None, 7, 7, 64)	0
conv2d_3 (Conv2D)	(None, 3, 3, 64)	102464
activation_3 (Activation)	(None, 3, 3, 64)	0
max_pooling2d_2 (MaxPooling2	(None, 1, 1, 64)	0
flatten (Flatten)	(None, 64)	0
dense (Dense)	(None, 64)	4160
activation_4 (Activation)	(None, 64)	0
dense_1 (Dense)	(None, 16)	1040
activation_5 (Activation)	(None, 16)	0
dense_2 (Dense)	(None, 1)	17
activation_6 (Activation)	(None, 1)	0

Total params: 1,944,161
Trainable params: 1,944,161

Figure III.6: résumé du nombre de paramètres entrainable du modèle

Ayant précisé que lors de l'entraînement qu'une partie des données serait alloués à des test réalisés à chaque epoch nous pouvons suivre l'évolution de notre modèle avec davantage de précision (voir figure III.6).

```
Train on 4708 samples, validate on 524 samples
Epoch 1/2
4675/4708 [============================>.] - ETA: 1s - loss: 0.1935 - acc: 0.7455
Epoch 00001: saving model to .
4708/4708 [==============================] - 174s 37ms/sample - loss: 0.1937 - acc: 0.7462 - val_loss: 0.2157 - val_acc: 0.8473
Epoch 2/2
4675/4708 [============================>.] - ETA: 0s - loss: 0.1396 - acc: 0.8088
Epoch 00002: saving model to .
4708/4708 [==============================] - 139s 29ms/sample - loss: 0.1394 - acc: 0.8090 - val_loss: 0.0871 - val_acc: 0.8798
<tensorflow.python.keras.callbacks.History at 0x7fe9f855ea20>
```

Figure III.7: résultat de l'entraînement du modèle CNN

Et c'est avec ceci , que se conclut l'implémentation du réseau à base de CNN

III .3 . 3 Implémentation du CapsNets

Afin de mettre en oeuvre les théories abordées par la thèse de Hinton, nous nous sommes inspiré de l'implémentation d'une couche de capsules utilisant le *routing by agreement* proposé par la documentation du framework keras qui bien que fonctionnelle n'est toujours pas pleinement intégré à ce dernier . ce qui comme nous le verrons par la suite , créer des soucis de performance .

Nous reprenons les mêmes étapes que celle effectuées précédemment quant à l'importation des bibliothèques , la création de callbacks et l'importation du dataset et sa simplification .

Dans le but de réussir à implémenter la couche dites de *Capsules* de notre modèle il nous faut au préalable adapter les fonctions offertes par le framework à l'emplois de vecteurs et non de scalaires tel que normalement implémenter par Keras.

Ainsi nous déclarons une fonction dite "Squache" qui servira de fonction d'activation de cette couche et qui à la faculté de normaliser des vecteurs définie comme suit

$$Squashe\ (v) = \frac{||v||^2}{1 + ||v||^2} \cdot \frac{v}{||v||}$$

puis nous définissons une fonction softmax qui elle est une fonction d'activation employée en sortie de la couche que nous adaptons à l'emplois de vecteurs.
Pour enfin , implémenter une nouvelle fonction de coûts basé sur la fonction *hing_loss* capable de traiter les sorties de ce type de couches .

quand à l'implémentation de la couche capsule nous la prenons tel qu'elle est définie par keras à la seule exception que nous optons pour une version de l'algorithme du routing by Agreement plus fidèle aux travaux de Hinton .

ne nous reste plus qu'à construire notre modèle couche par couche (voir figure III.8 du graphe de notre implémentation des caps nets) .

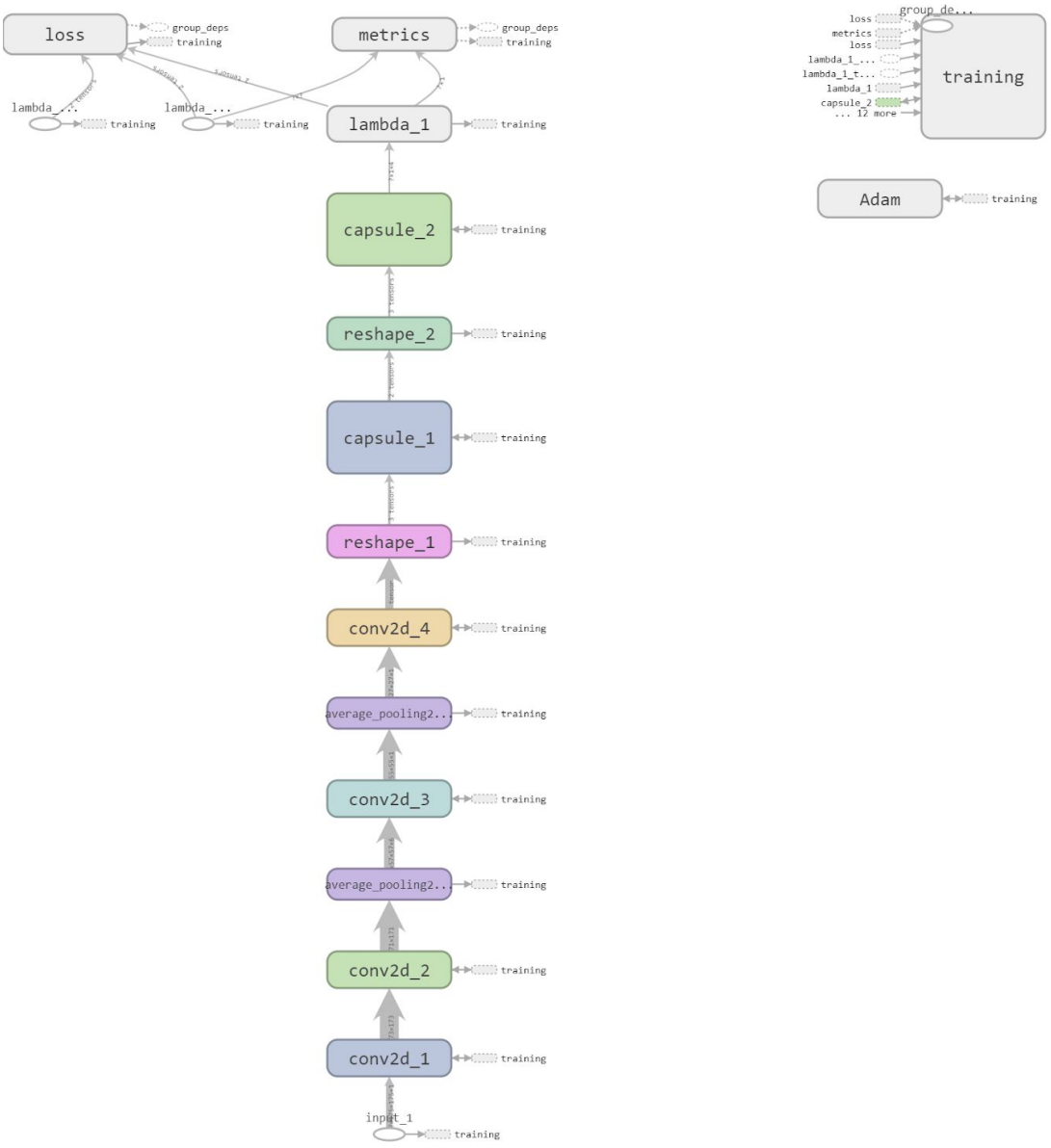

Figure III.8: Graphe des couches de notre implémentation de CapsNet

De plus vous pouvez vous référer au code ci-bas décrivant l'implémentation du réseau il faut noter que la méthode employé pour déclarer ce réseau et celle utilisée pour la création du CNN différent en raison de l'emplois de la couche capsule qui n'est pas normalement prise en charge par Keras de fait nous avons été contraint d'employer une méthode de déclaration qui tend à devenir obsolète.

```
i = Input(shape= images_train.shape[1:])
y = Conv2D(128, (5, 5), activation='relu')(i)
y = Conv2D(64, (4, 4), activation='relu')(y)
y = AveragePooling2D((3, 3))(y)
y = Dropout(rate = 0.5)(y)
y = Conv2D(64, (3, 3), activation='relu')(y)
y = AveragePooling2D((2, 2))(y)
y = Dropout(rate = 0.5)(y)
y = Conv2D(64, (3, 3), activation='relu')(y)
y = Reshape((-1, 64))(y)
c = Capsule(12, 20, 3, True)(y)
y = Reshape((-1, 20))(c)
y = Capsule(1, 4, 3, True)(y)

m = Model(inputs=i, outputs=y)
m.compile(loss=margin_loss, optimizer='adam', metrics=['accuracy'])
m.summary()
```

Nous procédons au paramétrage de l'entraînement de ce modèle pour lequel nous optons pour l'emplois de notre fonction dérivée de *hinge_loss* pour la fonction de coûts , nous opterons pour l'optimiseur adam une seconde fois et nous suivrons l'évolution de la justesse lors de l'entraînement.

Nous procédons ensuite au résumé des paramètres entrainable pour ce modèle et nous observons qu'ils sont au nombre de 200 milles (voir figure III.8) bien que 100 fois moins nombreux que ceux du CNN la complexité de fonctionnement des couches basées sur les capsules couplé aux problèmes de compatibilité avec le framework Keras l'entraînement de ce modèle prend un temps non négligeable comparé à d'autre type de réseaux (on parle de durées allant jusqu'à 24 heures voir davantage)

```
Train on 4708 samples, validate on 524 samples
Epoch 1/10
4708/4708 [==============================] - 1900s 403ms/step - loss: 0.0717 - acc: 0.7441 - val_loss: 0.0741 - val_acc: 0.7252
Epoch 2/10
4708/4708 [==============================] - 1899s 403ms/step - loss: 0.0691 - acc: 0.7438 - val_loss: 0.0676 - val_acc: 0.7252
Epoch 3/10
4708/4708 [==============================] - 1901s 404ms/step - loss: 0.0643 - acc: 0.7441 - val_loss: 0.0745 - val_acc: 0.7252
Epoch 4/10
4708/4708 [==============================] - 1903s 404ms/step - loss: 0.0620 - acc: 0.7438 - val_loss: 0.0622 - val_acc: 0.7252
Epoch 5/10
4708/4708 [==============================] - 1906s 405ms/step - loss: 0.0548 - acc: 0.7604 - val_loss: 0.0468 - val_acc: 0.7462
Epoch 6/10
4708/4708 [==============================] - 1907s 405ms/step - loss: 0.0422 - acc: 0.8430 - val_loss: 0.0375 - val_acc: 0.8511
Epoch 7/10
4708/4708 [==============================] - 1904s 404ms/step - loss: 0.0350 - acc: 0.8828 - val_loss: 0.0313 - val_acc: 0.9160
Epoch 8/10
4708/4708 [==============================] - 1894s 402ms/step - loss: 0.0327 - acc: 0.8978 - val_loss: 0.0422 - val_acc: 0.8302
Epoch 9/10
4708/4708 [==============================] - 1900s 403ms/step - loss: 0.0289 - acc: 0.9102 - val_loss: 0.0370 - val_acc: 0.8168
Epoch 10/10
4708/4708 [==============================] - 1900s 403ms/step - loss: 0.0262 - acc: 0.9172 - val_loss: 0.0236 - val_acc: 0.9351
<keras.callbacks.History at 0x7f0118426b38>
```

Figure III.9: résultats de l'entraînement du CapsNet

III .3 . 4 Implémentation d'un modèle via l'outil AutoML

Afin de lancer la création d'un modèle employant les stratégies liés au transfert learning nous avons opter pour l'utilisation de l'outil AutoML de la suite Firebase implémenté par Google,notamment pour des raisons de temps mais aussi afin de tester la beta de ce dernier.

La création d'un modèle ne nécessitant pas de prétraiter sois même les données ils nous a suffit de transférer celle-ci à un projet en ligne (voir figure III.9) créer via la console firebase(bien qu'il existe une alternative basé sur des lignes de commandes), puis de paramétrer le type de modèle le nombre de classes et de choisir une offre (n'ayant pas les moyens nous nous contentons de l'offre gratuite nous limitons à 1000 images et une seule heure d'entraînement au maximum).

A la fin de la phase d'entraînement il nous as ensuite été communiqué un résumé des tests effectué sur modèle ainsi qu'un accès vers une page de configuration nous permettant de favoriser des metrics au détriment d'autres.

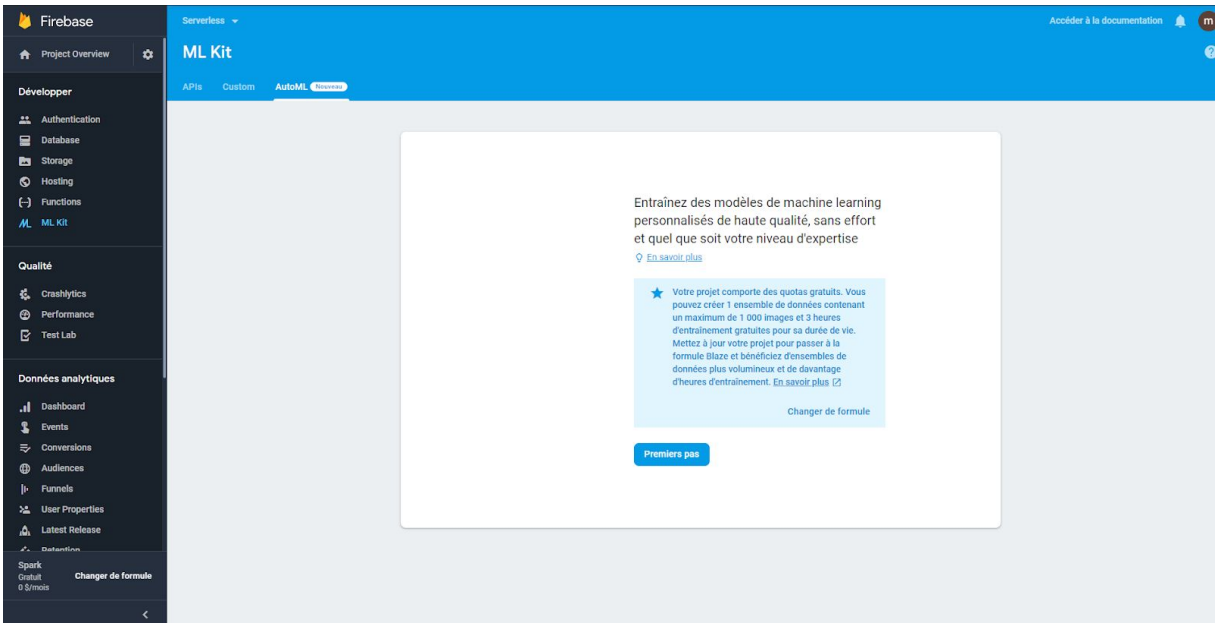

Figure III.10: interface de l'outil AutoML

III . 4 Conclusion

Nous concluons ce chapitre en rappelons qu'après avoir fait le tour des outils ,bibliothèques et autres frameworks employés lors de la phase de réalisation , et suite à notre étude détaillée de la mise en oeuvre de chacune des architecture et méthodes présentée durant le chapitre consacré à la conception , il nous est enfin possible de passer de la phase 4 du cycle de développement de modèles basés sur les réseaux de neurones profond; à savoir la phase de tests à l'issue de laquelle nous pourrons déterminer lequel de nos modèle est le plus intéressant

Chapitre IV Benchmarking

Dans ce qui suit nous tâcherons d'évaluer l'ensemble des solutions implémentées précédemment puis nous analyserons les résultats obtenus de sorte à désigner la plus convenable

IV .1 Introduction

A la suite de l'implémentation des différentes solutions précédemment introduites (voir chapitre III) à savoir une architecture basée sur les réseau de neurones convolutifs ***CNN*** (solution standard), la plus récente architecture basée sur l'utilisation de capsules ***Caps Nets*** ainsi qu'une technique tout aussi récente reposant sur le transfert de compétence ***transfer learning*** ; nous nous sommes ,dans ce qui suit , attelés à évaluer chacunes de ces alternatives à l'aide d'outils et de mesures introduits antérieurement (voir chapitre I.3.4 et chapitre II.2.3).

Nous tâcherons enfin de conclure ce chapitre sur une comparaison nous permettant d'affirmer laquelle de nos méthodes est la plus adapté à la classification d'images médicales .

IV . 2 Mesures et évaluation des différentes approches

IV. 2 . 1 Présentation des mesures employées

Afin de pouvoir évaluer au mieux chacune des approches que nous avons mis en oeuvre nous nous somme focalisés sur trois aspects clés de ce type de méthodes à savoir :

- la justesse et la valeure moyenne de la fonction de coûts

Pour rappel la justesse *Accuracy* est la mesure de la validité des prévisions effectuées par le model que nous cherchons à produire .

La mesure de la moyenne de la fonction de coûts sert quand à elle à nous renseigner sur l'écart entre les prédications de notre modèle et les réponses attendues .

Ces deux mesures pouvant être obtenues grâce à l'outil de monitoring ***Tensorboard*** (voir chapitre III.2) lors de la phase d'entraînement de notre réseau de neurones convolutifs et notre réseau de type *CapsNe*t .

Il est à noter qu'en raison de l'emplois de l'outils ***autoML*** dans le cas de la dernière approche il ne nous est pas possible de suivre l'évolution de ces mesures ainsi nous nous contenterons de la valeure finale de ces dernières .

- La durée moyenne de l'entraînement

Il s'agira ici de mesurer la durées nécessaire à l'entraînement d'un modèle ,dont les performances sont acceptables, avec l'une ou l'autre des deux architectures ou encore à l'emplois *d'autoML*.
Il est à noter que la durée de l'entraînement varie selon le hardware plus particulièrement selon que l'on emplois un CPU , un GPU ou un TPU qui eux même sont dotés de puissance de calcule différentes (nonobstant le fait qu'il reste possible d'employer des environnements mixtes)se faisant et dans un soucis de contrôle les mesures que nous fournissons dans ce documents ont été réalisé avec l'environnement d'exécution cloud fournis par **Collaboratory** .

De plus à la lumière de ces observations il apparaît comme évident de considérer le rapport entre les durées d'entraînement afin de réellement pouvoir établir une comparaison de performance plus fiable .

- La Sensibilité et la spécificité

Il s'agit de mesures standards employées dans le cadre de l'évaluations de solutions de type "diagnostic médicale automatisé" , permettant dans la cas de la sensibilité (Sensibility) d'évaluer la capacité du système à détecter et à signaler les cas suspects (c'est à dire les cas avéré comme ceux semblant l'être);

La spécificité quant à elle permet de quantifier le nombre de personnes jugées comme étant saines d'après le système à apprécier .(voir chapitre I.3.3) .

IV. 2 . 2 Résultats des différentes approches

IV. 2 . 2.1 Evaluation de notre CNN:

Dans ce qui suit nous tâcherons d'énumérer l'ensemble des résultats obtenus par notre implémentation d'un modèle basé sur un réseau de neurones convolutifs ,dans l'ordre de présentation de nos différentes mesures .

- La justesse et la valeure moyenne de la fonction de coûts

Grâce à l'utilisation de tensorboard il nous a été possible de suivre l'évolution de la précision (accuracy) (voir figure IV.1)ainsi que la valeure moyenne de notre fonction de coûts de notre modèle tout au long de son entraînement (voir figure IV.2) ; de plus de part l'emplois du paramètre d'entraînement *validation_split* il a été possible d'effectuer des tests de validation à chaque epoch nous permettant d'augmenter le nombre de nos mesures en y incluant une précision de validation (epoch_*val_acc*)(voir figure IV.3) et une moyenne de la fonction de coûts de validation(epoch_val_*loss*)(voir figure IV.4) .

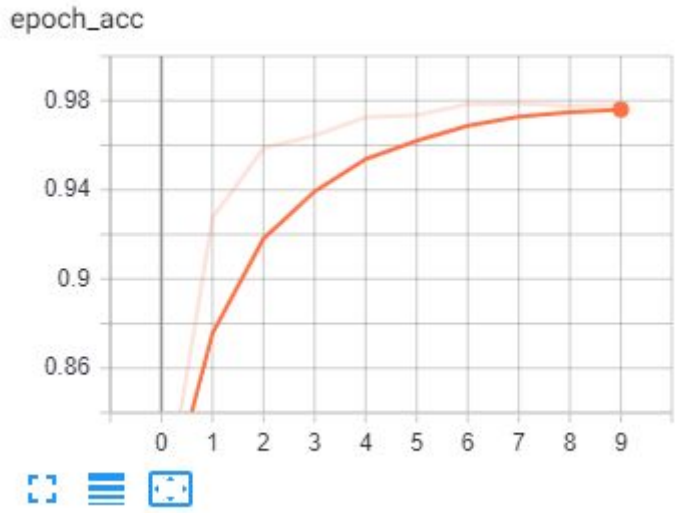

Figure IV.1 : graphe de l'évolution de la précision du CNN par epoch [f22]

L'on observe deux courbes en raison d'un traitement de lissage permettant de rendre la courbe initiale (la plus effacées) plus simple à interpréter.

Figure IV.2 : courbe de l'évolution de la moyenne de la fonction de coûts [f22]

En se focalisant sur les deux courbes précédentes l'on peut aisément observer l'évolution duale des deux mesures en effet de par la nature de celles-ci ou la précision mesure la faculté du modèle à réaliser de bonne prédications , la ou la fonction de coûts permet d'évaluer l'écart entre les prédication attendues et celles obtenues ; de fait à mesure que notre modèle s'améliore le nombre d'erreures qu'il commets s'amenuisent d'où la baisse du coûts la ou la précision augmente.

De part la nature des réseaux de neurones il est coutume de ne pas se fier au résultats obtenue en phase d'entraînement car en effet tel que vu précédemment (Chapitre I) ces derniers ont une fâcheuse tendance à la mémorisation des données d'entrainements conduisant le modèle dans les ecueilles de *l'overfitting*.

Ainsi afin de s'assurer de la validité de nos résultats il est de l'ordre de l'obligation de procéder à des testes de validation en soumettant au modèle obtenu des nouvelles données ceci pouvant se faire en fin d'entraînement ou à la suite de chaque epoch (traitement de l'ensemble du dataset d'entraînement) , dans notre cas nous avons opter pour les deux afin de suivre au mieux l'évolution de l'entraînement de notre modèle mais aussi afin de nous assurer de la validité de nos résultats.

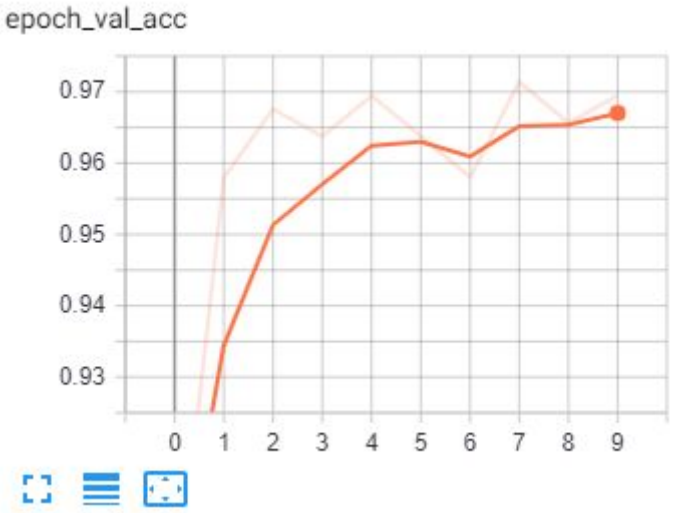

Figure IV.3: évolution de la précision sur le dataset de test par epoch

Figure IV.4 : evolution de la fonction de coûts de validation par rapport à chaque epoch

Il est à noter que cette évolution est similaire à celle de l'entraînement ce qui est une excellent chose car ceci indique en effet que notre modèle ne souffre pas du problème de *l'overfitting*

Un second test de validation est effectué mais cette fois afin d'obtenir une évaluation finale de la précision et du coûts de notre modèle (voir figure IV.5)

```
624/624 [==============================] - 8s 12ms/sample - loss: 0.1077 - acc: 0.8510
le valeur de Loss est de  0.10765756819492732
le valeur de la precision est de  0.85096157
```

Figure IV.5 : précision et coûts suite à l'évaluation

Les valeurs finales que nous avons obtenus sont relativement bonnes en effet avec une précision de 85.1 % notre modèle est relativement efficace a titre de comparaison le classement officiel de Stanford réalisé pour ce dataset vois le meilleur résultat s'élève à 91.7% notre résultat est proche de 23 ème place de ce classement.

- La durée moyenne de l'entraînement

Selon l'environnement d'exécution choisi , les temps d'entraînements ont pu fortement varier , nous tenons à rappeler que nous entendons par environnement d'exécution l'emplois d'un CPU, un GPU ou encore un TPU .

Notez bien que bien que disponible le TPU n'as hélas pas pus être exploité au maximum de son potentiel dû à l'emplois du framework **keras** dont certaines classes font appel à un ensemble d'instruction non compatibles avec ce derniers ce qui a hélas donné des résultats très décevant par rapport au capacité d'un tel hardware

Ce qui pour des raisons de lisibilité , de pertinence et de facilité nous a contraint à ne pas prendre en comptes les temps d'entraînement sur ce type de hardware .

Le hardware employé un CPU cadencé à 2.3 GHZ disposant d'un seul coeur et deux thread et un GPU de type Tesla K80 disposant de 2496 CUDA cores et de 12Go de mémoire

Temps d'exécution CPU par epoch	**Temps d'exécution GPU par epoch**
13 h	174 secondes

La comparaison ne laisse aucun doute la puissance effective d'un GPU comparé à un banale CPU ; nous poussons à ne jamais considérer l'éventualité de l'emplois d'un CPU.

- La Sensibilité et la spécificité

Sans conteste les deux mesures clé de notre batterie de tests car jusqu'alors inédites dans l'étude comparative entre les différentes solutions que nous avons implémenter. Elle apportent ici une meilleure vision de l'apport potentiel de la mise en oeuvre de ce type de solution dans le monde réel .

En effet en maximisant la sensibilité il devient à priorie moins risqué de mettre en circulation un model qui à défaut de savoir ou face à un doute agit prudemment en signalant le cas comme étant une anomalie.

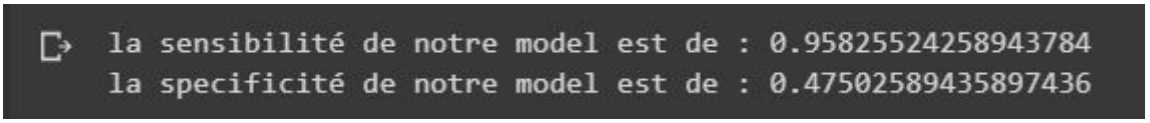

Figure IV.6 : résultat de l'évaluation de la sensibilité et de la spécificité du modèle

Comme indiqué par la figure IV.6 la mesure de sensibilité de notre modèle (qui doit tendre vers 1 celui-ci étant la valeure optimale) à un très bonne tendance à classer les cas suspects qu'il détecte comme étant des cas de personnes malades la ou d'après sa spécificité (qui elle doit tendre vers 0.5) le modèle à rarement tendance à laisser passer un cas suspects comme étant celui d'une personne en bonne santé.

IV. 2 . 2.2 Evaluation des Capsules networks:

Suivant l'ordre préétablie nous reviendrons sur chacuns des résultats de notre modèle obtenue à partir de notre réseau de neurones basé sur les capsule .

- La justesse et la valeure moyenne de la fonction de coûts

Ayant employé tensorboard une énième fois afin de suivre l'entraînement de notre modèle nous porterons notre attention sur le même type de graphes que ceux présentés précédemment.

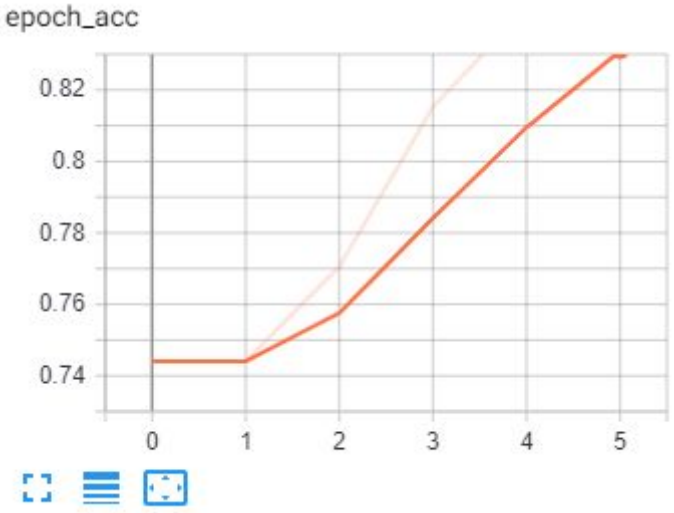

Figure IV.7 : graphe de l'évolution de la précision du CapsNets par epoch

Figure IV.8 : courbe de l'évolution de la moyenne de la fonction de coûts

Comme précédemment nous constatons une évolution duale des deux mesures lors de l'entraînement ce qui est comme expliqué plus haut un indicateur de la bonne évolution du modèle que nous entrainons .

Au vu de l'emplois du même paramètres d'entraînement "vaidation_split" pour ce modèle il nous est possible de nous focaliser sur des mesures d'avantage plus fiables

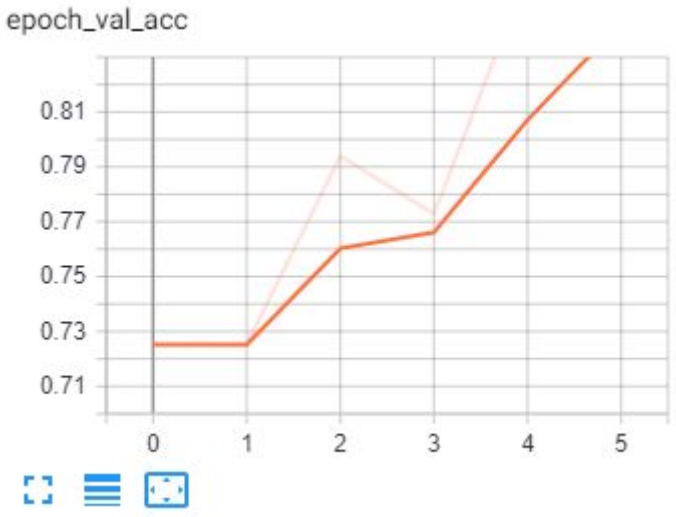

Figure IV.9 : évolution de la précision sur le dataset de test par epoch

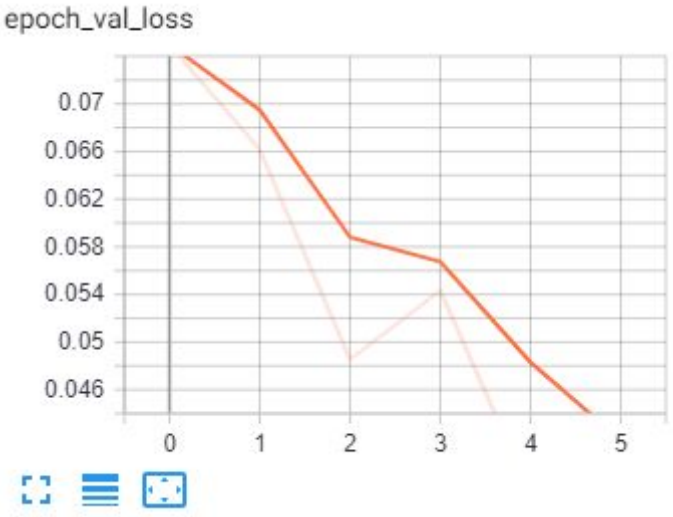

Figure IV.10 : evolution de la fonction de coûts de validation par rapport à chaque epoch

Nous pouvons observer que les deux graphes obtenue suite au testés concordent parfaitement avec l'évolution de mesures de l'entraînement ce qui indique que notre modèle n'est pas affecté par le problème de l'excès d'adaptation (overfitting).

Pour finir , il nous faut porter un intérêt particuliers aux valeurs finales obtenues après la seconde (et dernière) évaluation de la précision et du coûts (voir Figure IV.11).
nous prenons note de la très faible valeure du coûts (0.04) indiquant que notre modèle à tendance à être certain des prédications qu'il apporte pour une entrées données ainsi pour une entré x de sortie attendue 1 le modèle aura tendance à répondre 0.99 .

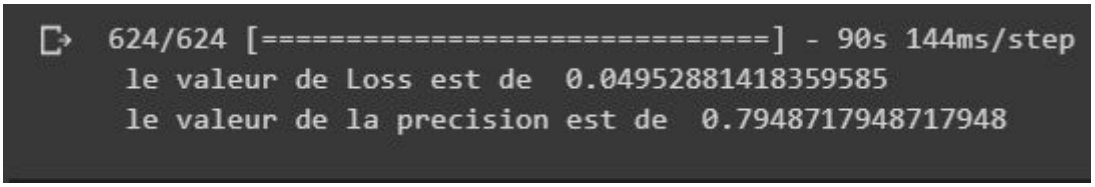

Figure IV.11 : précision et coûts suite à l'évaluation

- La durée moyenne de l'entraînement

Comme expliqué plus haut , nous ne nous focaliserons que sur les résultats offert par le GPU et le CPU de notre environnement de travail

Temps d'exécution sur CPU par epoch	**Temps d'exécution sur GPU par epoch**
le système a cracher faute de ram disponible estimé : 48 - 50 h	1 heure 27 minutes

- La Sensibilité et la spécificité

En nous référant aux résultats obtenus précédemment nous pouvons aisément voir que notre modèle est très réactif et a tendance à classer toute images suspect comme étant celle d'un malade et cela de façon suffisamment précise pour ne pas impacter sa précision .

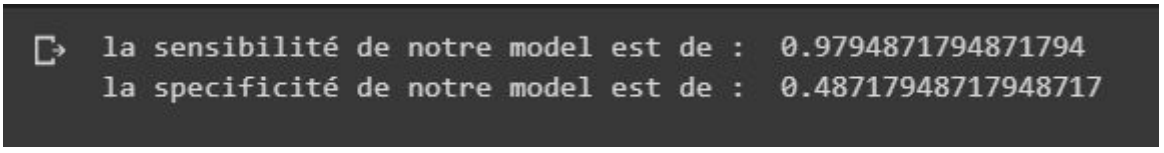

Figure IV.12 : résultat de l'évaluation de la sensibilité et de la spécificité du modèle

IV. 2 . 2.3 Evaluation de la solution implémenté via autoML:

Au vu du manque d'informations précise durant la phase d'entraînement du modèle implémenté par l'outil autoML nous nous référons aux résultats de l'évaluation système finale que le logiciel nous fournis

Figure IV.14 : résultat de l'évaluation du modèle implémenté par AutoML

Première , nous relevons que la précision de ce modèle est plus que satisfaisant surpassant même celle de nos précédentes solutions en effet elle s'élève à plus de 93 % et 97 % en moyenne (celle-ci étant calculé en se basant sur toutes les valeurs que peuvent prendre les paramètres ajustables dont on disposait afin de modifier le comportement du modèle) .
Ensuite, suite à une configuration que nous estimons optimale il nous faut nous attarder sur le rappel (équivalant à la sensibilité d'après la définition offerte par l'outil) qui avoisine les 95 % ce qui indique que le modèle est plus sûre .

Enfin , en ce qui concerne le temps d'entraînement d'un modèle créer via cet outils celui-ci est totalement dépendant de l'offre que vous avez souhaité acquérir de fait l'offre de base propose un entraînement d'une durée d'une heure sur un dataset de taille n'excédant pas le milliers d'images toutes classes comprises,pouvant selon les offres aller, pour la plus chère, à une quantité de données illimité pour une durée d'entraînement dépassant les 12 heures .

IV. 2 . 3 Comparaison et analyse

Nous tâcherons de réaliser un tableau comparatif reprenant l'ensemble des données et mesures effectuées précédemment,puis , en nous basons sur ce derniers nous tâcherons d'apporter une analyse constructive permettant de dégager la meilleur solution possible pour le type de problèmes abordé dans ce mémoire.

	Précision et Coûts	Meilleur Temps d'entraînement	Sensibilité et Spécificité
CNN	précision : 85 % Coûts : 0.1	170 secondes par epoch (en moyenne 10 minutes en tout)	Sensibilité :95% Spécificité :47%
CapsNet	précision : 80 % Coûts : 0.04	1h27 minutes par epoch (en moyenne 18 heures)	Sensibilité 98% Spécificité :48%
Transfer Learning via AutoML	précision : 93 % Coûts : non fournie	une heure	Sensibilité :95.6% Spécificité :non fournie

Figure IV.15: tableau comparatif

Grâce à la mise en perspective apportée par le tableau (figure IV.15), nous pouvons faire un certains nombre d'observations :

- Le modèle implémenté via l'outil AutoML possède la plus importante précision la ou le moins précis et le modèle de type CapsNet

- Le modèle le plus rapide à entraîner est notre CNN à l'opposé du CapsNet dont la durée d'entraînement peut facilement atteindre 18 h

- Le modèle ayant la plus grande sensibilité et la meilleure spécificité est le modèle issue des Capsule la ou le CNN reste légèrement en retrait par rapport aux deux autres .

En nous basant sur sur les mesures ainsi observées et les propriétées des différentes solutions nous pouvons déduire ceci

- Le modèle dont les prédictions traduisent le plus une forme de certitude (Coûts le plus faible) est le modèle CapsNet la ou le CNN à tendance à produire des prédictions plus incertaines (pour des réponses attendues de forme 1 ou 0 le CNN auras tendance à prédire des valeurs entre [0.2 - 0.8])

- Ayant entraîné le CNN et le CapsNet avec plus de 5000 images pour ensuite les tester avec plus de 1000 images les résultats obtenue par ces derniers sont relativement plus fiables que ceux produit par le modèle de l'outil autoML en raison de son entraînement / test effectué avec un set limité.

- L'outils AutoMl produisant des modèle via des réseaux de neurones à l'architecture inconnue il est très difficile d'évaluer la fiabilité d'un tel modèle ou de le configurer de

sorte à corriger d'éventuelles erreures

- L'outils autoML bien que fonctionnel est encore en phase beta et ne permet pas pour le moment de mettre à jour le modèle produit par ses soins

IV . 3 Conclusion

En nous basant sur l'ensemble des observations effectuées par nos soins à la suite de la batterie de tests que nous avons effectuer sur chacuns des modèles implémentés selon les différentes approches abordées par notre étude , il se dégage de nos résultats le constat suivant ,

la solution la plus fiable pour le cas de la classification d'images médicales reste encore à ce jours l'emplois de réseaux de neurones convolutifs au vu de la simplicité de la mise en oeuvre de cette solution , ses résultats plus que acceptables , sa maintenabilité là ou le concurrent CapsNet bien qu'ayant des résultats prometteurs (un coûts plus faible , la plus grande sensibilité) reste encore aujourd'hui beaucoups trops coûteux en ressources pour des résultats approchant ceux d'un CNN.

Enfin , l'outil AutoMl peut selon les cas constituer une solution viable mais sa jeunesse son opacité et son statut de projet en version bêta nous empêche au vu de la nature de notre problématique de le considérer en tant que tel.

Conclusion Générale

Ce mémoire avait pour ambition de déterminer la meilleure approche issue du *deep learning* quand à l'aide au diagnostique médicale de la pneumonie via la classification de radiographies du thorax via l'implémentation de plusieurs de ses approches à savoire une première reposant sur les réseaux de neurones convolutifs ,une seconde plus récente inspirée des travaux du chercheur Hinton basée quant à elle sur les capsules et le routage d'information par agrément et enfin un modèle issue des techniques de transfert de connaissance implémenté via l'outil autoML.

Pour ce faire nous nous sommes plongés dans l'immense domaine qu'est le Deep Learning et avant acquis les compétences nécessaire à l'étude de chacune de ces approches et par la suite fort de notre savoir nous avons implémenter chacunes de ces dernières . en parallèle de quoi, nous avons introduits des outils de mesures propre au domaine de la problématique à savoire la sensibilité et la spécificité que nous avons mesuré au même titre que des valeurs plus classiques tel que la justesse et l'écart type calculé par la fonction de coûts.

Fort de ces observations nous avons conclus à l'issue de notre chapitre consacré au benchmarking de ces solutions que la plus courante des trois reste suffisamment fiable et peu contraignante pour le moment afin de la considérer comme étant la plus intéressante des trois.

Nous pensons qu'au terme de ce projet nous avons réussi à atteindre les objectifs que nous sommes fixés que ce soit sur le plan pratique ou même personnel dus à l'expérience acquise durant la mise en oeuvre de ce mémoire .De fait à la lumière de ces compétences nouvelles nous prévoyons d'explorer davantage cette problématique en perfectionnant les modèles produits de sorte à offrire un diagnostic plus nuancé, explorer d'autres approches et architectures tel que l'architecture TECAV de Google annoncé en Mai de cette année voir encore perfectionner nos tests en y incluant de nouvelles mesures .

Bibliographie

Webographie

[C1] : livre de : Nicholas Locascio et Nikhil Buduma , Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithms;lhttps://books.google.dz

[C2] : Claude Touzet , LES RESEAUX DE NEURONES ARTIFICIELS INTRODUCTION AU CONNEXIONNISME
http://www.touzet.org/Claude/Web-Fac-Claude/Les_reseaux_de_neurones_artificiels.pdf

[C3] : Chloé-Agathe Azencott, Architecture d'un perceptron;
https://openclassrooms.com/fr/courses/4470406-utilisez-des-modeles-supervises-non-lineaires/4730716-entrainez-un-reseau-de-neurones-simple

[C4] : wikiversity, avantages des réseaux de neurone;
https://fr.wikiversity.org/wiki/R%C3%A9seaux_de_neurones/Avantages_et_possibilit%C3%A9s
[C5] : wikiversity, Les inconvénients des réseaux de neurone;l
https://fr.wikiversity.org/wiki/R%C3%A9seaux_de_neurones/Points_faibles_et_limites

[C6] : wikiversity, domaine d'application;
https://fr.wikiversity.org/wiki/R%C3%A9seaux_de_neurones/Applications_des_r%C3%A9seaux_de_neurones

[C7] : Pascal Monasse et kimia Nadjahi,lassifiez les images à l'aide de réseaux de neurones convolutifs;https://openclassrooms.com/fr/courses/4470531-classez-et-segmentez-des-donnees-visuelles/5082166-quest-ce-quun-reseau-de-neurones-convolutif-ou-cnn

[C8] : Aryan Misra ,capsule networks ;
https://towardsdatascience.com/capsule-networks-the-new-deep-learning-network-bd917e6818e8?fbclid=IwAR3qq6QQHq6nQM3grufCu8FuKuxyAkU5YbLCeKBWBienpOVTXSn4m6Bjs2s
[i1] pilote automatique de tesla https://www.tesla.com/autopilot véhicules autonomes de Google https://waymo.com/

[i2] plateforme d'aide au diagnostique du cancer du sein https://www.analyticsinsight.net/artificial-intelligence-aids-breast-cancer-detection-and-saves-time/
[i3] division medicale de Google https://ai.google/healthcare/
[i4] initiative e-santé en GB https://www.ehidc.org/resources/artificial-intelligence-healthcare
[i5] rapport sur l'ethique et limites judiciaire de l'emplois du DL dans le domaine medicale en europe https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6206380/
[CI.1] Anish Singh Walia , Activation Functions and its types , which is better ? https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f
[CI.2] goerge seif ,https://towardsdatascience.com/understanding-the-3-most-common-loss-functions-for-machine-learning-regression-23e0ef3e14d3
[CI.3] Michael Nielson , http://neuralnetworksanddeeplearning.com/chap2.html

- Le framework tensorflow https://www.tensorflow.org/

- L'api kears https://keras.io/models/about-keras-models/

- L'outil automl https://firebase.google.com/

- La bibliothéque numpyhttps://www.numpy.org/

- La bibliothéque numpy https://pandas.pydata.org/

- lien github vers l'implémentation de CapsNets https://github.com/bojone/Capsule/

Ouvrages et thèses

- N.Frosst.S.Sabour,G.Hinton, Dynamique routing between capsules, https://arxiv.org/abs/1710.09829

- Aurélien Geron ,Hands-On Machine Learning with Scikit-Learn and TensorFlow ,O'Reilly , 2017

- Dr. Adrian Rosebrock ,Deep Learning for Computer Vision with Python Starter Bundle 1st Edition (1.1.0)

- Nikhil Buduma with contributions by Nicholas Locascio,Fundamentals of Deep Learning DESIGNING NEXT-GENERATION MACHINE INTELLIGENCE ALGORITHMS ,

O'Reilly ,2017

- CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning Pranav Rajpurkar Jeremy Irvin Kaylie Zhu Brandon Yang Hershel Mehta Tony Duan Daisy Ding Aarti Bagul Robyn L. Ball Curtis Langlotz Katie Shpanskaya Matthew P. Lungren Andrew Y. Ng

- Transfer Learning Lisa Torrey and Jude Shavlik University of Wisconsin, Madison WI, USA

- CapsNet comparative performance evaluation for image classification Rinat Mukhometzianov and Juan Carrillo University of Waterloo, ON, Canada

Les Sources des figures

[f1]: réalisé via google draw

[f2] : Nicholas Locascio et Nikhil Buduma , Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithms;l:https://books.google.dz

[f3] :https://bit.ly/2S7tBZ0

[f4] Claude Touzet , LES RESEAUX DE NEURONES ARTIFICIELS INTRODUCTION AU CONNEXIONNISME;:
http://www.touzet.org/Claude/Web-Fac-Claude/Les_reseaux_de_neurones_artificiels.pdf

[f5]:https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f

[f6]:https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f

[f7]:http://neuralnetworksanddeeplearning.com/chap2.html

[f9]: ChestXRay , Stanford Dataset

[f10]:https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

[f11] Understanding of Convolutional Neural Network (CNN) ; https://www.freecodecamp.org/news/understanding-capsule-networks-ais-alluring-new-architecture-bdb228173ddc/?fbclid=IwAR1ilXf7Gf1kx3dWlcnOLYyDgz--9Zi__zWfh6UZQSz_p6Pxq6XzQYWmoto

[f12] Capsule Networks; https://towardsdatascience.com/capsule-networks-the-new-deep-learning-network-bd917e6818e8?fbclid=IwAR3qq6QQHq6nQM3grufCu8FuKuxyAkU5YbLCeKBWBienpOVTXSn4m6Bjs2s

[f13]https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a

[f14] hands on machine learning with scikit-learn and tensorflow de Aurélien Géron , O'reilly 2017

[f15] https://www.revmed.ch/RMS/2017/RMS-N-583/L-echographie-pleuro-pulmonaire-pour-le-pneumologue

[f16]https://imm.fr/fiche-info-patient/bilan-preoperatoire-avant-une-chirurgie-dexerese-de-tumeur-pulmonaire