CentraleSupélec

OPENCLASSROOMS

Parcours Data Scientist : projet n° 8

Rapport de veille thématique

Réseaux de neurones dans le domaine de la classification d'images

---

Christophe Coudé

Novembre 2017

Table des matières

I. Introduction....3
II. La brique de base, le neurone....4
III. Historique sommaire de quelques modèles....5
IV. Historique et description des premiers modèles....7
A. Principes initiateurs de l'architecture RNA, TLU/MCP, le premier neurone formel....7
B. Le Perceptron, une évolution du TLU/MCP....9
C. Le réseau de Hopfield et les réseaux récurrents....10
D. Perceptron multicouche et rétropropagation....11
V. Modèles pour la reconnaissance d'images....12
A. Réseaux convolutifs, spécialistes de la reconnaissance d'images....12
B. Réseaux VGG....14
C. Réseaux ResNet....14
D. Réseaux à module Inception....15
E. Un exemple de développement très récent, les Capsules de Hinton....15
VI. Etude de cas....17
A. Base de données....17
B. Transfer Learning....18
C. Le modèle Inception V3....19
D. Résultats....21
E. Erreurs de classification....22
VII. Conclusion....23
VIII. Annexes....23
A. Fonctions d'activation....23
B. Algorithme d'apprentissage....26
C. Types de connexions entre couches....27
D. Domaines d'utilisation des réseaux neuronaux....28

I. Introduction

Les réseaux de neurones artificiels (RNA) sont des réseaux informatiques qui tentent de simuler d'une façon sommaire les réseaux de cellules nerveuses du système central nerveux biologique et s'inspire de la connaissance neurophysiologique des neurones biologiques. Cela diffère des machines informatiques conventionnelles qui sont utilisées pour dépasser les capacités calculatoires des cerveaux humains mais sans considération organisationnelle. La différence est d'importance car en simulant les réseaux biologiques, c'est une toute autre architecture informatique ainsi qu'une nouvelle architecture d'algorithmes qui ont été envisagées. Auparavant, pour résoudre des questions complexes, mathématiquement mal définis ou des problèmes non linéaires ou stochastiques, on utilisait des algorithmes conventionnels recourant à des ensembles d'équations complexes, *ad hoc*, adaptées spécifiquement au problème. Avec les RNA, il n'est alors plus question que d'opérations informatiques simples (addition, multiplication, opérations logiques) ainsi que d'algorithmes dont la plasticité permet de répondre à une gamme très large de problèmes. La structure des RNA répond à un maximum de simplicité et d'auto-organisation et son potentiel à utiliser des calculs parallèles la rend bien plus performante que le système séquentiel des machines numériques conventionnelles (une erreur dans une séquence et tout s'arrête, alors la mort de centaines de neurones chaque année n'impacte pas le fonctionnement du cerveau outre mesure). L'introduction des réseaux artificiels a donc ouvert la voie à une nouvelle compréhension de la façon de simplifier la programmation et l'algorithmie [1].

Ils sont utilisés de nos jours dans un très grand nombre de domaines différents, de la finance à l'aérospatiale, en passant par le médical. En reconnaissance audio, on utilise des réseaux neuronaux récurrents (RNN) comme le Long Short-Term Memory (LSTM) et le Gated Recurrent Units (GRU) car les RNN permettent de garder en mémoire des informations [38] [39] [43]. D'autres réseaux sont utilisés en déchiffrage de signes et études de textes [40] [41]. Dans le monde de l'entreprise et de la recherche, il faut bien sûr trouver et produire, mais il faut surtout être les premiers à trouver et à développer de nouvelles idées. La concurrence provoque une émulation qui doit pousser à posséder et à utiliser l'information avant les autres. Et en cela, la veille thématique, et dans notre contexte, la veille technologique, est un ingrédient essentiel de l'innovation car elle constitue la recherche perpétuelle des informations les plus critiques. Bien sûr, elle ne doit pas étendre son domaine de surveillance à des domaines trop larges, il faut choisir le bon « learning rate » de la veille, une sorte de « watch span » qui doit se restreindre en priorité aux besoins de l'entreprise. J'ai donc choisi comme thème un cas d'utilisation précis des réseaux de neurones, celui de la classification d'images.

Une définition

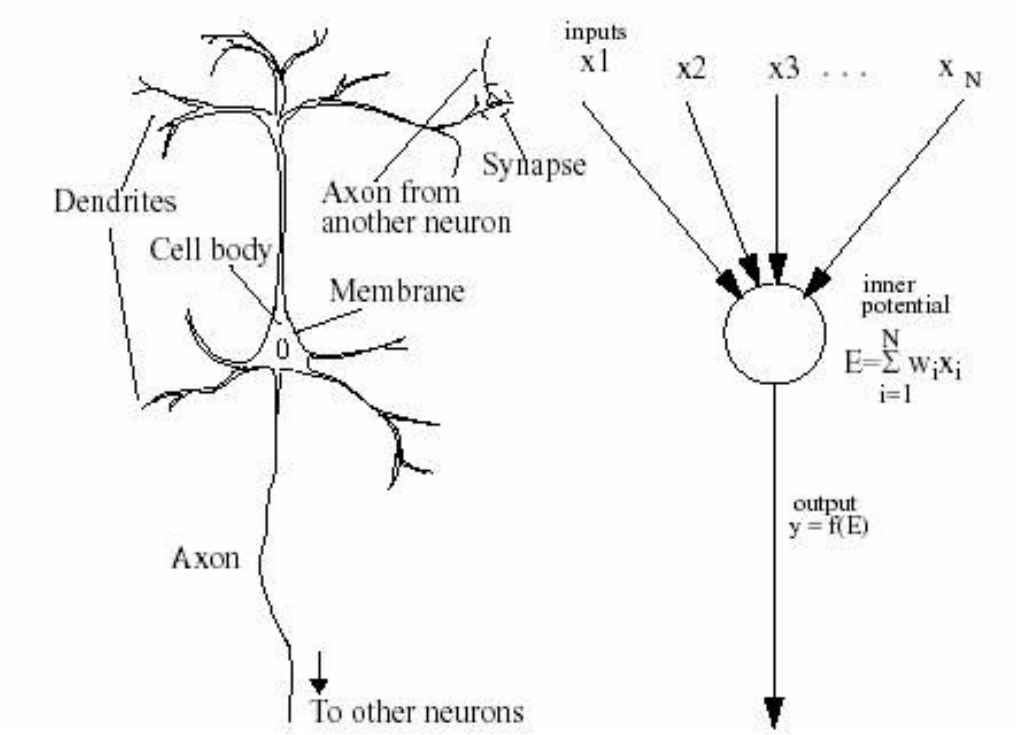
(N.B. les anglo-saxons utilisent le terme de « unit », donc d'unité pour différencier le neurone artificiel du neurone physiologique.)

« Un réseau de neurones est une assemblée interconnectée d'éléments simples de transformation, unités ou nœuds, dont la fonctionnalité est approximativement basée sur le neurone animal. La capacité de transformation du réseau est stockée dans les forces de connexion entre les unités, c'est-à-dire les poids, obtenus par un processus d'adaptation à un ensemble de schémas d'entraînement, ou apprenant de ces schémas. » [2]

L'unité de base d'un réseau de neurones est donc… le neurone. Avant de considérer l'histoire et l'évolution des réseaux de neurones, il est essentiel de décrire ce qu'est un neurone et comment il fonctionne.

II. La brique de base, le neurone

Le neurone mathématique a été conçu par analogie avec le neurone biologique. S'il ne modélise pas entièrement toutes ses fonctions, il en a adopté un certain nombre.



Schématiquement :

(1) Un neurone biologique est connecté à ses homologues, formant réseau.

(2) Il existe un sens de transport de l'information (différence de potentiel). Celle-ci entre dans le neurone par ses dendrites et son corps cellulaire (ou soma) et en ressort via son axone.

(3) La connexion entre neurones se fait entre les synapses, respectivement axodendritiques et axosomatiques, des terminaisons axoniques d'un neurone, et la membrane, respectivement des dendrites et du corps d'un autre neurone.

(4) La connexion entre ces neurones n'est pas figée, la force synaptique dépend des activités antérieures des neurones et de leur « utilisation » de cette connexion.

(5) La plus grande partie de la transformation de l'information se passe dans le corps cellulaire du neurone (contenant noyau, nucléole, mitochondries etc.).

(6) En réponse à la somme des influx reçus, et si cette somme dépasse un certain seuil, alors le neurone génère un potentiel d'action au niveau du segment initial (région où l'axone quitte le soma) qui se transmet sans atténuation le long de l'axone.

On retrouve « globalement » le même schéma pour les neurones artificiels avec toutefois des différences (par exemple, dans le neurone biologique, l'axone peut également recevoir des influx). Un neurone artificiel peut recevoir des inputs (1) de nombreux autres neurones (en biologie, le neurone peut être connecté à plus de mille neurones). Il existe également un sens de transfert de l'information avec une polarisation valeurs input/output du neurone (2). Le message entrant dans un neurone est constitué des outputs « x_i » des neurones auxquels il est connecté (3), modulés par la force des différentes synapses et qu'on appelle poids « w_i » (4) (puisqu'un poids négatif aura tendance à inhiber un neurone comme la diminution de la quantité de neurotransmetteurs transmis alors qu'un poids positif participera à l'excitation du neurone). Dans le « corps » du neurone artificiel s'effectue la somme de ces produits (5) et une fonction d'activation permet de déterminer la valeur du potentiel d'action et du seuil à dépasser afin que celui-ci soit généré et transmis en sortie du neurone (6).

Il est à noter que le fonctionnement d'un réseau de neurones biologiques est bien plus complexe que les quelques fonctions décrites précédemment et donc que les réseaux de neurones artificiels n'ont pas comme objectif de calquer parfaitement leurs homologues biologiques. Mais l'essentiel est bien retenu, utiliser la plasticité synaptique afin de trouver les « équilibres » (donc les valeurs optimales) permettant de répondre au mieux à un problème. Par exemple dans le cas d'un problème de classification supervisée d'images, l'entraînement permettra de sonder les meilleures configurations de connexions entre neurones, affaiblissant les ponts entre neurones qui ne participent pas efficacement au résultat escompté, et au contraire, favorisant ceux qui permettent d'aboutir au bon résultat.

Remarque : dans tous les réseaux de neurones qui seront présentés ci-après, l'activation d'un neurone correspond toujours à la somme des flux entrants pondérés. Mais il existe d'autres types de réseaux de neurones où cela peut être différent. On notera par exemple les MNN ou Morphological Neural Network où au lieu de faire la somme des produits input * weight, le résultat est déterminé par la valeur maximale des sommes input + weight.

III. Historique sommaire de quelques modèles

(Remarque : les dates varient souvent d'un auteur à l'autre…, mais cela ne se joue qu'à un ou deux ans près, cela dépend si la date retenue correspond à la période où les modèles sont discutés, par exemple dans des congrès ou des relations épistolaires, ou à des dates de parution d'articles officiels ou de livres qui sont nécessairement un peu décalées.)

Mise en place des principes et premiers succès :

(1) Warren McCulloch et Walter Pitts ont concrétisé leurs principes par un modèle formel (que l'on qualifierait de nos jours comme) très simple, abstraction du neurone physiologique, le TLU (Threshold Logic Unit) en 1943. Ils veulent ainsi démontrer que le cerveau est équivalent à une machine de Turing, la pensée devenant alors un ensemble de mécanismes matériels et logiques.

(2) Le Perceptron, inspiré du système visuel, est le plus ancien des RNA après le TLU, et a été proposé en 1958 par le psychologue Frank Rosenblatt [3]. Rosenblatt construit le premier neuro-ordinateur basé sur ce modèle et l'applique à la reconnaissance de formes [4]. Il possède deux couches de neurones : une couche de perception et une couche liée à la prise de décision. C'est le premier système artificiel capable d'apprendre par expérience.

(3) Le Artron, dû à R. J. Lee en 1959 [5]

(4) En 1960, Bernard Widrow et Ted Hoff développèrent un neurone particulier l'Adaptive Linear Neuron (Adaline) ou Adaptive Linear Combiner (ALC) [6]. En 1988, Widrow et Rodney Winter utilisent l'Adaline pour créer un réseau, le Madaline (Many Adaline) [7].

Période d'ombre :

(5) 1969, M. Minsky et S. Papert mettent en exergue les limitations théoriques du perceptron, même si elles sont déjà connues. En effet, le perceptron ne peut alors pas traiter des problèmes non linéaires, ce qui correspond à la majorité des problèmes. En étendant implicitement ces limitations à tous modèles de réseaux de neurones artificiels, ils favorisent l'abandon financier des recherches dans le domaine, les chercheurs se tournant principalement vers l'IA et les systèmes à bases de règles [4].

(6) 1967-1982, Même si les recherches ne sont pas toutes abandonnées, les RNA connaissent une période difficile. Elles perdurent toutefois dans des domaines spécialisés comme le traitement adaptatif du signal, la modélisation en

neurobiologie etc. dans lesquels par exemple Stephen Grossberg et Teuvo Kohonen ont travaillé.

Renaissance :

(7) 1982, le physicien John J. Hopfield présente une théorie du fonctionnement et des possibilités des réseaux de neurones [8]. Il démontre l'intérêt d'utiliser des réseaux de neurones récurrents pour la compréhension et la modélisation des fonctions de mémorisation. Si les précédents auteurs proposaient d'abord une structure et une loi d'apprentissage, n'étudiant qu'après les propriétés émergentes, J. J. Hopfield a d'abord fixé le comportement souhaité pour son modèle, puis à partir de là, construit la structure et la loi d'apprentissage correspondant au résultat escompté. Son modèle est actuellement utilisé pour des problèmes d'optimisation. A cette époque, l'IA n'ayant pas répondu à toutes les attentes et ayant montré certaines limitations délicates, et même si les limitations du Perceptron n'ont également pas été levées par le modèle d'Hopfield, les recherches sur les RNA ont quand même été relancées [4].

(8) 1983, les limitations du perceptron disparaissent avec la Machine de Boltzmann. Toutefois, son utilisation pratique reste difficile car la convergence de l'algorithme est extrêmement longue [9].

(9) 1986, les psychologues David E. Rumelhart, Geoffrey. E. Hinton et l'informaticien Ronald. J. Williams proposent un réseau basé sur la rétropropagation de gradient, un RNA basé sur un Perceptron multicouches et donnant une solution élégante à l'apprentissage des couches cachées [10]. La rétropropagation a été découverte par trois équipes de chercheurs indépendantes. Grâce à cette méthode, il a été possible de réaliser une fonction non linéaire d'entrée/sortie sur un réseau en décomposant cette fonction en une suite d'étages linéairement séparables. Actuellement, les réseaux multicouches avec la rétropropagation de gradient restent le modèle le plus productif au niveau des applications.

(10) 1987, Robert Hecht-Nielsen propose le Counter-Propagation Network (CPN) qui combine l'apprentissage de Kohonen et celui de Grossberg pour faciliter l'apprentissage non supervisé [11].

Les réseaux développés ultérieurement comme ART, le Cognitron, LAMSTAR etc. incorporent certains des éléments de ces réseaux fondamentaux ou les utilisent comme des briques combinées à d'autres éléments de décision.

Approfondissement des réseaux (reconnaissance d'images) :

(11) Des réseaux spécifiques aux images sont créés avec la famille des réseaux neuronaux convolutifs, par exemple le LeNet dans les années 1990 [12]

(12) Avec les concours d'ImageNet, les ILSVRC (ImageNet Large Scale Visual Recognition Challenge), toute une gamme de réseaux de plus en plus profonds et complexes sont développés afin de gagner ce concours chaque année avec par exemple :

(13) 2012, l'AlexNet d'Alex Krizhevsky [13] est une version plus profonde et plus large du LeNet.

(14) 2013, le ZFNet de Matthew Zeiler et Rob Fergus [14] est une amélioration de l'AlexNet via une modification des hyperparamètres du modèle.

(15) 2014, le GoogLeNet de Christian Szegedy et al. [15] invente le module Inception qui permet de réduire drastiquement le nombre de paramètres.

(16) 2014, le VGGNet du Visual Geometry Group d'Oxford [16] ont montré que la profondeur des réseaux (le nombre de couches) était un composant critique d'une bonne performance.

(17) 2015, le Residual Network de Kaiming He et al. [17] a utilisé une communication entre des couches éloignées et non plus seulement entre couches adjacentes.

(18) 2016, le DenseNet de Gao Huang et al. [18] est un réseau où toutes les couches sont connectées à toutes les autres couches.

(19) 2017, CapsNet et les Capsules de Hinton de Geoffrey E. Hinton [19].

IV. Historique et description des premiers modèles

A. Principes initiateurs de l'architecture RNA, TLU/MCP, le premier neurone formel

Les fondations de ce qui a commencé à être connu comme « calcul neural », « réseau neural » ou « connexionnisme », ont été posées en 1943 par le neurophysiologiste Warren McCulloch et le mathématicien Walter Pitts [20]. Afin de décrire comment les éléments de calcul basique du cerveau pourraient fonctionner, McCulloch et Pitts ont montré comment de simples circuits électriques connectant des groupes de « fonctions linéaires à seuil » pouvaient calculer de nombreuses fonctions logiques [21].

Dans leur article *A logical calculus of ideas immanent in nervous activity*, McCulloch et Pitts ont listé cinq hypothèses gouvernant les opérations neurales et définissant ce qu'on appelle maintenant le neurone McCulloch-Pitts (MCP) :

(1) Le neurone est un système binaire possédant deux états, « excité / actif » (firing) et « au repos / non actif » (non firing).

(2) Chaque neurone a un seuil fixé.

(3) Pour qu'un neurone soit excité, il faut qu'un certain nombre fixé de synapses plus grandes que 1 soient excitées.

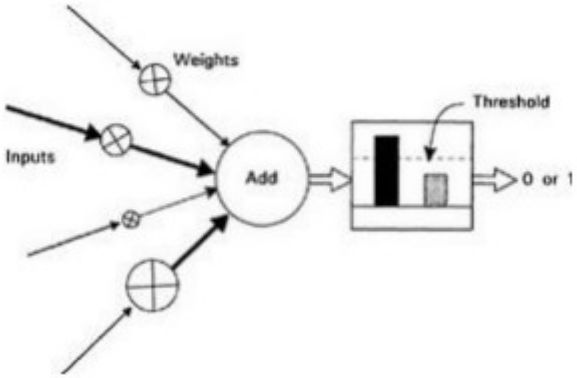
(4) Le neurone peut recevoir un input de la part de synapses inhibitrices dont l'action est « absolue » ; l'arrivée d'une quelconque de ces synapses empêche le neurone de se déclencher.

(5) Il y a un laps de temps dans lequel toutes les entrées synaptiques doivent être intégrées par le neurone.

En résumé, l'opération neuronale s'effectue de la manière suivante : pendant un laps de temps déterminé, s'il n'y a pas de flux sur une synapse inhibitrice, le nombre de flux excitants est sommé et si celle somme égale ou dépasse un certain seuil, le neurone déclenche une réponse, sinon il reste inactif.

Le résultat principal de l'article de McCulloch et Pitts a été de montrer que des tables de vérités logiques d'une quelconque complexité pouvaient être construites à partir des neurones MCP, c'est-à-dire que toute expression logique finie pouvait être exprimée par n réseau adapté de neurones McCulloch-Pitts.

En permettant d'effectuer des calculs très complexes, McCulloch et Pitts ont ouvert la voie à une nouvelle forme de calcul basée sur le design de réseaux de neurones artificiels plutôt que sur des machines de programmation de Turing.



A partir de ce neurone de McCulloch et Pitts, il a donc été construit des réseaux de neurones artificiels (RNA, ou ANN en anglais) comme groupes connectés de neurones MCP. Le calcul neuronal a donc fondamentalement défini un mode de calcul cherchant à intégrer le style de « calcul » utilisé par le cerveau, un fonctionnement basé sur l'apprentissage par l'expérience et non plus par des méthodes et algorithmes classiques spécifiquement définis.

Depuis que McCulloch et Pitts ont démontré qu'un réseau de MCP était Turing-complet (ayant une puissance de calcul au moins équivalente à celle des machines de Turing et toutes les fonctions calculables au sens de Turing le sont par ce réseau), les chercheurs ont cherché à appliquer les réseaux de neurones à des tâches mathématiques qui, si elles étaient effectuées par des humains, seraient dites impliquer de l'intelligence.

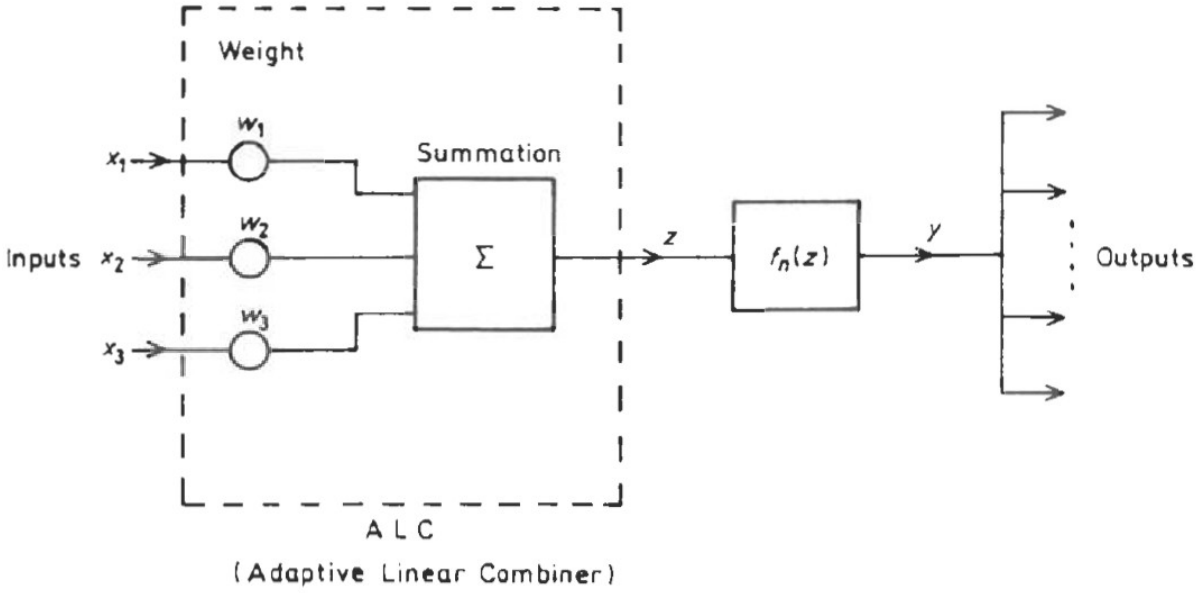
Les RNA ont été utilisés pour modéliser trois classes importantes de fonctions complexes :

- Association de vecteurs : un réseau de neurones associatif mappe un vecteur en entrée à un vecteur donné en sortie.
- Classification de vecteurs : mapping d'un vecteur d'entrée à un ensemble de classes
- Approximation de fonctions : par exemple T = f(X, Y, Z) où X, Y et Z sont des vecteurs d'entrée et T un vecteur en sortie.

B. Le Perceptron, une évolution du TLU/MCP

En 1958, Rosenblatt, un ancien camarade de Minsky (celui-là même qui devait onze ans plus tard mettre en exergue les limites du Perceptron), a publié une première description de son neurone calculateur, le Perceptron, neurone utilisant celui de McCulloch et Pitts.

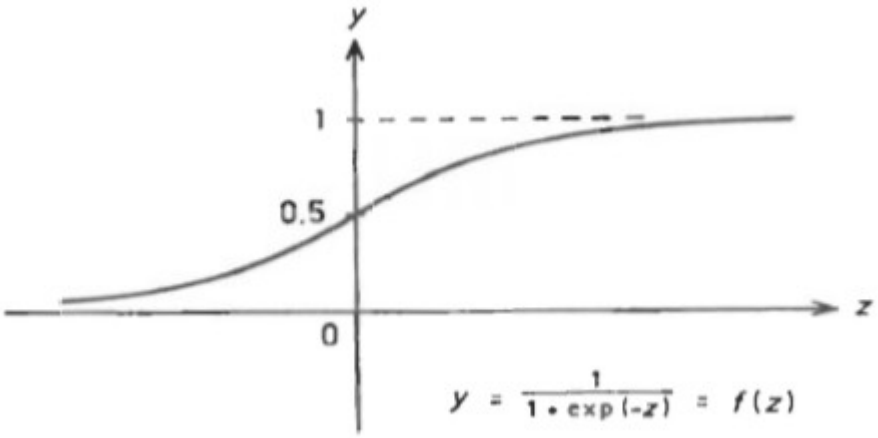
Schéma général :



Comme on peut le comparer avec les précédents schémas, le Perceptron ressemble fort au MCP/TLU mais en diffère en plusieurs points :

- Si pour le TLU, la fonction d'activation est fixée, $f(x) = 1_{[\theta, +\infty[}(x)$, pour le Perceptron, on peut la choisir suivant différents critères.
- La sortie n'est plus binaire mais peut prendre tout un panel de valeurs

A l'époque, la fonction d'activation utilisée était souvent la fonction sigmoïde :



D'autres fonctions d'activation seront introduites au fur et à mesure comme la tangente hyperbolique (tanh), l'unité de rectification linéaire (ReLU) etc. (voir l'annexe A pour la description des différentes fonctions d'activation et de leur choix).

En 1961, Rosenblatt a formulé le théorème d'apprentissage du Perceptron affirmant que le Perceptron peut apprendre tout ce qu'il peut simuler. (En réalité, cela n'est vrai que pour un réseau à au moins deux couches de Perceptrons et non pour un Perceptron isolé.) Il développa la PCP ou Perceptron Convergence Procedure qui fut perçue comme une avancée importante par rapport à la règle de Hebb [22] sur l'apprentissage des poids dans un réseau complexe.

J'ai mis en annexe B un exemple d'algorithme d'apprentissage pour un Perceptron.

Limitations :

E.B. Crane en 1965 puis Minsky et Papert en 1969 ont mis en exergue les sérieuses limitations des capacités du Perceptron en montrant par exemple qu'il était incapable de résoudre un problème logique à deux états Exclusive-Or (XOR) : [23]

$$[(x_1 \cup x_2) \cap (\bar{x}_1 \cup \bar{x}_2)]$$

En fait, c'est tout large gamme de problèmes que des classifieurs à une seule couche ne pouvaient pas résoudre, surtout si le nombre d'input grandissait.

Minsky et Papert ont montré qu'un RNA monocouche pouvait résoudre des problèmes de classification de points qui se trouvent dans une partie ouverte et convexe, ou fermée et convexe. Mais il n'y avait pas de méthodes pour des non convexes. Ce n'est qu'en 1986 qu'entres autres Rumelhart montra qu'un RNA à deux couches pouvait résoudre des problèmes non convexes, y compris le problème XOR. Mais dans les années 60 et 70, aucun outil ne permettait de calculer facilement et rapidement les poids d'un réseau multicouches (la méthode de rétropropagation n'a été découverte qu'en 1986).

J'ai mis en annexe E, le code python d'un simple Perceptron à 2 couches simulant cette fonction XOR. L'ajout de la couche cachée et la rétropropagation du gradient de l'erreur permet sa résolution.

Toutefois, il existait quand même des réseaux multicouches qui étaient, dans une certaine mesure, entraînés, le Madaline, basé sur le neurone Adaline de Widow. Mais il était très lent et pas assez rigoureux pour le problème général du multicouche.

Le réseau de Hopfield et les réseaux récurrents

Comme je l'ai indiqué précédemment, à la fin des années 1960, après les travaux pionniers de McCulloch, Pitts, Rosenblatt, Widrow et Hoff, deux chercheurs, Minsky et Papert, ont virtuellement provoqué à eux-seuls l'arrêt des recherches dans ce domaine naissant du calcul neuronal. A la publication de leur livre et avec le début du succès de l'intelligence artificielle symbolique, approche privilégiée par Minsky et Papert, il devint assez impopulaire de continuer des recherches dans le domaine de l'apprentissage des réseaux neuronaux. On est alors entré dans une période dite de « l'hiver du connexionnisme ». Toutefois, certains chercheurs comme Bledsoe et Browning (1959), Aleksander (1970), Hopfield, Grossberg et Kohonen ont continué leurs travaux dans ce domaine [24].

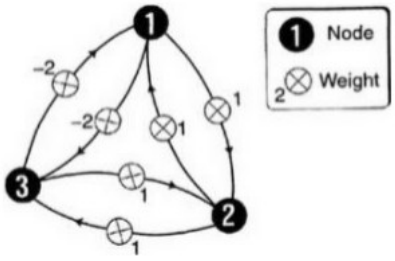
Ainsi, le physicien John Hopfield, en 1982, utilisa des idées importées de la physique statistique pour analyser le comportement d'un simple réseau binaire de cellules de type MCP dans lequel la sortie de chaque neurone était connectée aux entrées de tous les autres neurones du réseau. Pour Hopfield, à tout moment, la sortie de tous les neurones pouvait constituer un état transitoire du réseau. Hopfield montra qu'en prenant un vecteur initial arbitraire, et après une série d'états temporaires, un tel réseau pouvait s'arrêter sur un schéma stable de déclenchements de neurones après un nombre fini d'itérations. Ce réseau fonctionnait alors comme un pattern d'associations bi-directionnelles, associant un vecteur initial arbitraire à un vecteur de sortie. L'apprentissage du réseau de Hopfield revient donc à calculer les paramètres du réseau de façon à ce que les codes des informations que l'on souhaite mémoriser soient des états stables du réseau.

Le réseau de Hopfield fut utilisé pour résoudre des problèmes d'optimisation difficiles et de satisfaction de contraintes, et a obtenu de bons résultats [25].

Jusqu'à présent, nous avons vu des réseaux dont l'information circulait vers l'avant (feed-forward), de la couche d'entrée vers la couche de sortie. Nous avons ici affaire à un autre type de réseaux, un réseau de type récurrent, et plus encore, un réseau à connexions complètes. J'ai décrit en annexe C quelques-uns des différents types de connexions possibles entre les couches de neurones.

Par exemple, le schéma ci-dessous montre un réseau de Hopfield avec 3 TLU connectés les uns aux autres (mais pas à eux-mêmes) avec des poids symétriques. Il y a donc un feedback dans le réseau qui le rend récurrent, c'est-à-dire que les neurones peuvent être utilisés plusieurs fois pour transformer l'information.

[26]



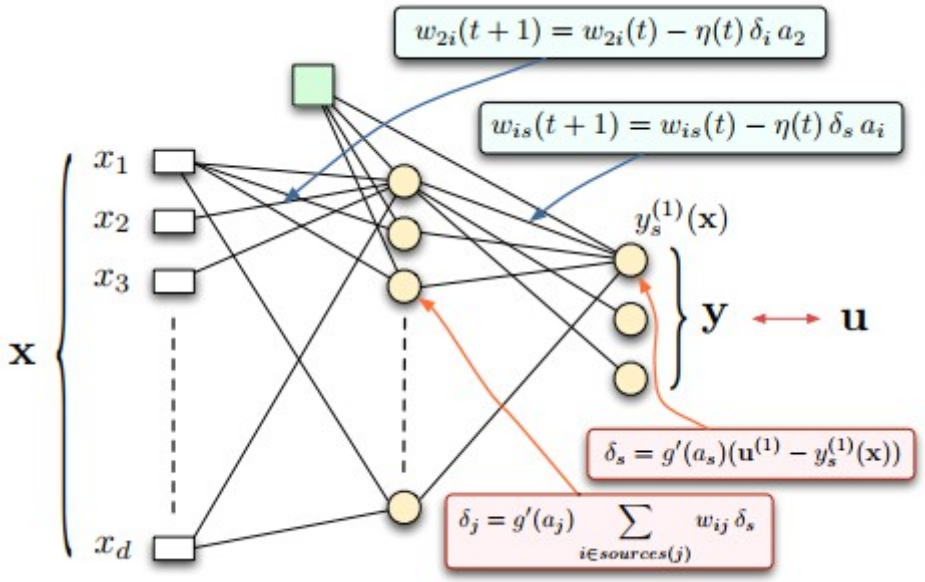
Perceptron multicouche et rétropropagation

Un des moments clef dans la recherche des réseaux neuronaux qui a mené à une « renaissance » dans ce domaine, a été le développement d'une règle d'apprentissage pouvant être appliquée à des réseaux neuronaux à plusieurs couches. Cette règle qui a été développée en même temps pour les MLP par plusieurs équipes différentes mais qui a été popularisée par Rumerhart, Hinton et Williams en 1986, a été appelée « rétropropagation » (backpropagation) ou « règle delta généralisée ». (Il faut toutefois noter que cet algorithme est bien plus ancien et remonte aux années 1960.)

Cette règle permet de propager un terme d'erreur « en arrière » à travers le réseau, des neurones de sortie vers les neurones d'entrée, modifiant les poids de façon à ce que l'erreur totale de prévision soit minimisée.

La règle delta généralisée fonctionne en effectuant une descente de gradient dans cet espace des poids/erreurs, c'est une généralisation de la règle delta standard (règle de Widrow-Hoff). En utilisant la rétropropagation, après que chaque élément d'entraînement soit passé à travers les couches en feedforward, l'erreur de prévision du réseau est calculée en la comparant avec la valeur attendue, puis en revenant dans le sens inverse, chaque poids est du réseau est modifié en diminuant le gradient de l'erreur. La descente de gradient implique de changer chaque poids en proportion de la négation de la dérivée de l'erreur.

Schéma du modèle de rétropropagation de l'erreur :



Modèles pour la reconnaissance d'images

Avec le développement des Perceptrons multicouches et la large adoption des règles d'apprentissage de la rétropropagation, quelques-unes des critiques clefs des perceptrons monocouches par Minsky et Papert ont été écartées et s'en est suivi au cours des décennies suivantes un grand nombre d'articles détaillant l'application des technologies de réseaux neuraux dans différents domaines comme par exemple les systèmes d'identification et de contrôle (Irwin, 1995), les jeux et la prise de décision (Tesauro, 1989), la reconnaissance de motifs (Aleksander et Stonham, 1979), la reconnaissance de séquences (gestes, discours, textes écris etc.), la localisation de robots mobiles (Katevas, 1997, Bishop, 1998), le data mining, filtres de spam etc. (J'ai mis en annexe D une liste, non exhaustive, de quelques domaines d'utilisation des réseaux neuronaux.) La reconnaissance des images est depuis quelques années un sujet très important, un concours a même été créé pour faire rivaliser les modèles, l'ILSVRC (http://www.image-net.org/challenges/LSVRC/).

Les Perceptrons multicouches ont un peu de mal à gérer les images de grande taille car on est passé par exemple d'un total de 269 322 paramètres à entraîner pour le MLP appliqué aux images du MNIST à 38 603 786 paramètres pour les images d'ImageNet. Comme dans les Perceptrons, les couches sont toutes entièrement connectées, le nombre de paramètres augmente énormément. Une piste de recherche s'est alors portée sur les réseaux convolutifs dont les neurones d'une couche ne sont pas connectés à tous les neurones de la couche précédente.

Réseaux convolutifs, spécialistes de la reconnaissance d'images

Les réseaux convolutifs ne sont pas récents, ils apparaissent dès les années 1980 avec la famille des Cognitron, Neocognitron par Fukushima, puis par exemple avec le HMAX de Riesenhuber, M. & T. Poggio en 1999 et le LeNet de LuCun en 1998.

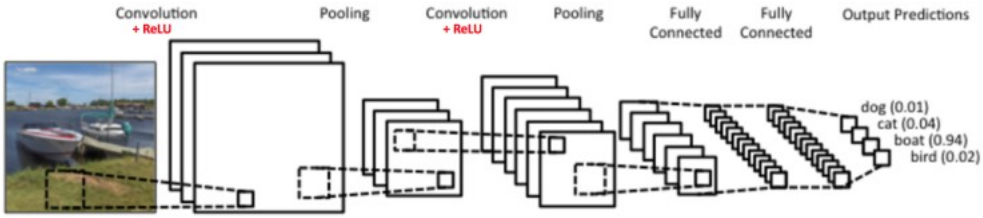
Pour résumer rapidement le principe d'un réseau de neurones convolutif (ou CNN / ConvNets en anglais), son composant principal est la couche convolutive. Son travail est de détecter les caractéristiques importantes des pixels de l'image. Les couches les plus profondes (les plus proches de l'entrée du réseau) apprennent à détecter des caractéristiques simples comme des bords ou des gradients de couleur, alors que les

couches les plus élevées combinent les caractéristiques simples en des caractéristiques plus complexes. Les dernières couches entièrement connectées au sommet du réseau combinent alors les caractéristiques de plus hauts niveaux pour produire les prévisions de classification.

Avantages du CNN :

- Localité : le champ récepteur permettant de limiter les entrées à une zone locale de l'image, les filtres modélisent mieux des motifs particuliers de l'image que lorsqu'ils doivent traiter l'image entière. Cela réduit en outre la mémoire nécessaire.
- Invariance spatiale : dans une fibre (ou depth column, ensemble des neurones en profondeur pointant tous vers le même receptive field), le paramétrage des neurones est le même, donc également dans un même noyau, ce qui fait que le motif filtré par ce noyau (et cette fibre) ne dépend pas de la localisation spatiale au sein de l'image.

Description du CNN : [27]

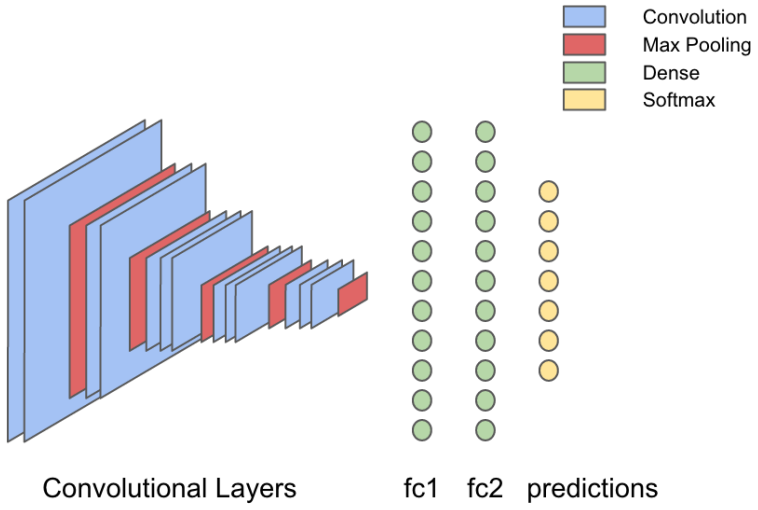


- Les couches CONV et FC effectuent des transformations qui sont des fonctions des activations dans le volume d'entrée mais aussi des paramètres (poids et biais des neurones).
- Les couches de POOLING ont pour fonction de réduire progressivement la taille spatiale (hauteur / largeur, mais pas profondeur) de la représentation afin de diminuer le nombre de paramètres, réduisant les temps de calculs ainsi que l'éventuel overfitting (surapprentissage) du modèle.
- Les « couches » ReLU correspondent en réalité à la partie dévolue à la fonction d'activation qui n'est plus une sigmoïde comme dans le Percpetron mais la fonction max (0, x) permettant d'introduire une non linéarité.
- Les couches DROPOUT permettent d'éviter une part d'overfitting en désactivant de façon aléatoire un certain nombre de neurones. Il existe tout un ensemble de méthodes de réduction de l'overfitting en plus du pooling et du dropout, comme le DropConnect, ou l'augmentation du nombre de données d'entraînement.
- Les couches FC (Fully-Connected / Dense) des hauteurs du réseau (top-layers of the stack) servent à la classification, les couches précédentes ayant pour but d'extraire les caractéristiques de l'image. Si l'on dispose parfois plusieurs couches FC, c'est pour mettre en exergue les combinaisons non linéaires de ces caractéristiques.
- Récemment, il a été ajouté la possibilité d'incorporer des couches de BatchNormalization qui permettent de normaliser les activations de la couche précédente pour chaque batch, ce qui améliore les performances du réseau.

C. Réseaux VGG

Visual Geometry Group a créé deux principaux réseaux, les VGG16 et VGG19 qui sont des CNN avec respectivement 16 et 19 couches. Ils contiennent essentiellement des groupes de couches de convolution 3x3 entre lesquels s'intercalent des couches de pooling 2x2, et se termine par 3 FC. Le VGG16 semble avoir été le premier à surpasser l'être humain dans la reconnaissance des images d'ImageNet. C'était un modèle qui au début utilisait beaucoup de paramètres, toutefois il s'est avéré qu'on pouvait retirer les FC sans grande baisse de performance.

Schéma du VGG16 : [28]



Il arriva deuxième au ILSVRC de 2014 derrière GoogLeNet.

D. Réseaux ResNet

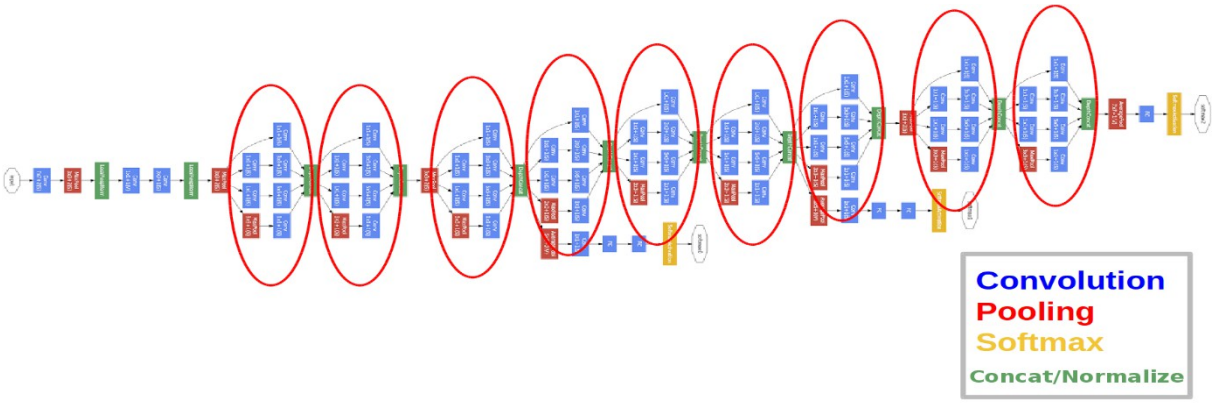
RESNET (resnet pour Residual Network) a pour origine du residual learning. Il est le gagnant du ILSVRC 2015. Au fur et à mesure des recherches, avec l'approfondissement des réseaux, l'entraînement est devenu de plus en plus difficile et l'accuracy a commencé à saturer et à se dégrader. On pouvait penser qu'en ajoutant toujours plus de couches, on obtiendrait toujours de meilleurs résultats, ou du moins égaux. Ce n'est pas le cas. Le residual learning a tenté de résoudre ces problèmes. Ses auteurs sont partis de l'hypothèse que les mappings directs entre deux entités étaient difficiles à apprendre. Ils ont donc proposé, plutôt que d'apprendre les liens sous-jacents, de tenter d'apprendre les différences, c'est-à-dire les résidus. Il ne reste plus alors qu'à ajouter ces résidus à l'entité initiale pour obtenir l'autre. En residual learning, le modèle effectue des raccourcis en connectant directement les input d'une couche plus basse (ex. la n-ième) à une couche plus élevée (ex. la (n+x)-ième). Il a été prouvé qu'entraîner ce genre de réseaux était plus facile qu'entraîner un simple CNN et que cela résolvait le problème de dégradation de l'accuracy.

Il a été développé plusieurs modèles, le ResNet50, ResNet101 et ResNet152 où les nombres indiquent le nombre de couches. Toutefois, même si le nombre de couches est bien plus élevé que pour les VGG, la complexité des ResNet est inférieure à celle des VGG.

Ce ne sont que quelques modèles, bien d'autres existent.

E. Réseaux à module Inception

Les réseaux à module Inception sont une piste différente pour améliorer les performances. Ils feront l'objet d'une étude de cas en chapitre 7.



F. Un exemple de développement très récent, les Capsules de Hinton

Il y a quelques semaines, Geoffrey Hinton et son équipe ont publié deux articles [19] [36] (dont l'un est encore actuellement en train d'être vérifié) introduisant un type complètement nouveau de réseaux de neurones basés sur les dénommées « capsules » ainsi qu'un algorithme « dynamic routing between capsules » permettant d'entraîner un tel réseau.

Comme on peut le voir dans les références et en parcourant la littérature, Geoffrey Hinton est un des fondateurs du Deep Learning et l'inventeur de nombreux modèles et algorithmes utilisés de nos jours (il fut un des chercheurs à introduire entre autres la rétropropagation).

On a vu que dans les CNN, la couche de Max Pooling permettait de réduire le nombre de paramètres et de limiter l'overfitting. L'expérience montre que l'utilisation de ces couches améliore les performances du modèle, ce qui semble paradoxal puisqu'il y a une perte d'information. Or pour G. Hinton, l'opération de pooling est une grande erreur et le fait que cela marche si bien est un désastre. Le problème est que la représentation des données internes d'un CNN ne prend pas en compte les hiérarchies spatiales entre les objets complexes et les objets simples. Le pooling était supposer introduire des invariances de position, d'orientation et de proportion. Mais la méthode utilisée étant trop grossière, cela a eu au contraire l'effet d'ajouter toute sorte d'invariances de position.

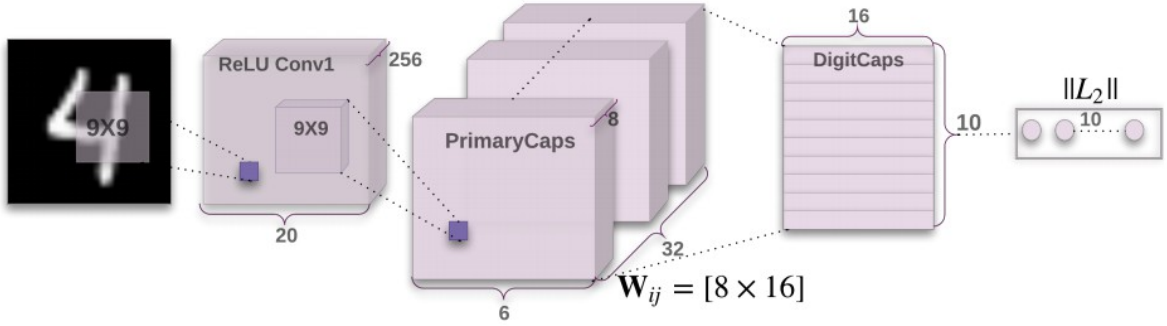
Dans l'opération de « rendering » d'un ordinateur, un software prend la représentation d'une image stockée en mémoire et la convertit en une image à l'écran. Cette représentation prend en compte les positions relatives des objets via des principes internes de hiérarchies de formes géométriques, et est gardée en mémoire dans des tableaux d'objets géométriques et des matrices. Pour Hinton, le cerveau fait le contraire. A partir des images reçues par les yeux, le cerveau décompose le monde en une représentation hiérarchique et essaie de la faire correspondre à des schémas déjà appris stockés dans le cerveau. L'idée maîtresse est que ces représentations dans le cerveau ne dépendent pas de l'angle de vue selon lequel on voit un objet. (En effet, quand on voit un objet, on peut la plupart du temps le reconnaître quel que soit l'angle sous lequel on le regarde. Ce n'est pas le cas pour un CNN.) En graphisme en 3D, les relations entre les objets 3D peuvent être représentées par ce qu'on appelle une « pose » (translations et rotations).

Pour Hinton, afin de classer et de reconnaître correctement les objets, il est important de préserver les « relations de pose » entre les parties des objets. La capsule de Hinton intègre cette notion de relativité entre les parties des objets et les objets et les représente numériquement dans une matrice 4D de pose. Quand ces relations sont construites dans une représentation interne, il devient facile pour le modèle de comprendre que la chose qu'il voit n'est qu'une autre vue de quelque chose qu'il a déjà vu et non quelque chose d'autre. L'angle de vue n'est plus alors plus un problème.

Pour un CNN normal, le fait de ne pas intégrer cette compréhension interne de l'espace en 3D rend difficile la reconnaissance de points de vue différents d'un même objet. Les invariances des CNN les rend tolérant aux petits changements dans le point de vue, mais pas aux changements plus importants. Ce n'est pas le cas pour le réseau CapsNet (réseau développé par Hinton et utilisant ces capsules) car il intègre cette notion. Selon [36] (*capsules reduce the number of test errors by 45% compared to the state-of-the-art*), les capsules permettent une diminution très importante du nombre d'erreurs de reconnaissance. Un autre avantage est que la capsule de Hinton est capable d'apprendre au même niveau que les meilleurs modèles mais en n'utilisant qu'une fraction des données qu'un CNN aurait utilisées. En cela, les capsules se rapprochent plus du fonctionnement du cerveau qui n'a pas besoin d'autant de données d'entraînement que les CNN.

Hinton avait ces idées depuis des décennies, mais il lui manquait des ordinateurs suffisamment puissants pour les mettre en pratique et il n'y avait pas encore d'algorithme pour les faire fonctionner. Avec les nouvelles GPU et le nouvel algorithme « dynamic routing between capsules », il a enfin pu les développer. Celui-ci permet aux capsules de communiquer entre elles et de créer les représentations adéquates.

Il faudra néanmoins attendre de voir ce que donnent les tests dans divers domaines car les implémentations actuelles sont plus lentes que les autres modèles les plus performants, même si sur le dataset du MNIST, les résultats sont excellents.



Plus techniquement, une capsule est un ensemble de couches de neurones enchâssées. Dans un réseau habituel, on a l'habitude d'ajouter des couches de neurones. Dans CapsNet, on ajoute des couches à l'intérieur d'une couche. L'état des neurones à l'intérieur d'une capsule capture les propriétés d'un objet à l'intérieur d'une image. Le vecteur en sortie de capsule représente la probabilité que l'objet soit présent dans le domaine limité d'observation et un ensemble de paramètres d'instanciation qui peuvent inclure la pose, l'éclairage ou la déformation de cet objet relativement à une version canonique de cet objet déjà défini.

L'algorithme développé par Hinton introduit un mécanisme itératif de « routage par accord » selon lequel une capsule de bas niveau préfère envoyer sa sortie à des capsules de plus haut niveau dont les vecteurs d'activité ont un produit scalaire plus important avec la prédiction provenant de la capsule de plus bas niveau. Ce routage dynamique assure que la sortie de capsule soit envoyée à un parent plus approprié de la couche supérieure. Il peut être vu comme un mécanisme parallèle d'attention qui permet à une capsule d'un niveau de s'occuper des capsules actives du niveau inférieure et d'ignorer les autres.

Dans le schéma précédent, on observe un CapsNet simple avec 3 couches qui donne des résultats comparables aux CNN profonds. La longueur du vecteur de sortie de chaque capsule dans la couche DigitCaps indique la présence d'une instance de chaque classe et est utilisée pour calculer la perte de la classification.

Il sera intéressant de comparer ces nouveaux développements des capsules avec les Spatial Transformer Networks [37] qui notent également les problèmes d'invariances spatiales des CNN et qui tentent d'y remédier en implémentant un nouveau module d'apprentissage, le *Spatial Transformer*, qui permet de reconnaître des transformations spatiales en les intégrant explicitement à l'intérieur du réseau. De la même manière que les

capsules enchâssées, ces modules peuvent être insérés dans une architecture convolutive existante.

V. Etude de cas

Il peut être intéressant de faire une étude de cas car cela permet de prendre un contexte particulier, un sujet de recherche ou de développement en entreprise par exemple, et d'évaluer et comparer les différents modèles existants afin de faire ressortir un « state of the art », c'est-à-dire les plus hautes connaissances acquises dans un domaine (le terme « état de l'art » fait trop référence à un tout de la connaissance et non à son apogée). L'étude de cas est donc naturellement un élément préalable et essentiel d'une veille technologique.

A. Base de données

Je me suis donc orienté vers une étude de cas dans le domaine de la reconnaissance d'image, domaine très prisé puisqu'il a donné lieu depuis 2010 à une compétition annuelle où des équipes de recherche évaluent leurs algorithmes de traitement d'images sur un jeu de données, ImageNet. Le but est d'obtenir la meilleure précision sur plusieurs tâches de vision par ordinateur. Pour cette étude, j'ai rapatrié en local la partie dévolue aux chiens à partir du site http://vision.stanford.edu/aditya86/ImageNetDogs/ qui sont des images provenant d'ImageNet (http://www.image-net.org/). On peut télécharger également la description de ces images comprenant 120 races de chiens dont 12000 images d'entraînement et 8580 images de test. La taille des images est très variable, cela va d'environ 100x100 pixels à 3200x2500 pixels. Cependant, les chiens à l'intérieur des images ont dans la grande majorité des cas une taille qui ne dépasse pas 500x500 pixels.



Comme on peut le voir dans ces exemples, la grande diversité des images ne rend pas la tâche facile aux modèles. Ceux-ci doivent être capables de bien différencier la race d'un chien quand un ou des humains sont présents, quand il y a plusieurs chiens sur une même photo, et quand le chien n'est pas dans une position de portrait.

Nous avons vu dans le chapitre précédent un certain nombre de modèles assez récents. J'ai choisi le modèle Inception V3 pour l'expliquer en détail, un modèle très récent qui date de décembre 2015 [42] car il me semble représenter l'architecture la plus proche de la notion d'apprentissage.

Dans les tests des premiers modèles, Perceptrons, CNN simple, j'ai été limité dans mes tests par les capacités de mon ordinateur qui n'a que 4 CPU. J'ai donc restreint le nombre de

races de chien de 120 à 10, ce qui fait en tout 1000 images d'entraînement réparties donc en 10 catégories et 707 images de test.

B. Transfer Learning

Pour les modèles les plus récents, ResNet50, InceptionV3 et Xception, j'ai utilisé des modèles pré-entraînés afin de gagner en temps d'exécution et obtenir des résultats qui soient plus représentatifs et un peu plus comparables entre eux. Le pré-entraînement des poids sur une base d'entraînement donnée est un outil très utile car il arrive souvent que, même avec des GPU, les modèles les plus complexes prennent deux à trois semaines pour entraîner les CNN sur toute la base d'ImageNet. Or, lorsqu'un particulier réutilise les mêmes images et les mêmes modèles, il est inutile qu'il recommence un travail déjà effectué puisque les résultats de ce travail ont été mis à disposition du public, en l'occurrence, les paramètres entraînés des modèles. Cela est d'autant plus profitable quand un particulier n'utilise pas exactement les mêmes images, mais des images assez proches (les chihuahuas hors ImageNet seront toujours assez proches des chihuahuas d'ImageNet), surtout s'il ne dispose pas d'une base de données aussi développée que celle d'ImageNet. Il est alors intéressant d'utiliser le transfer learning auquel correspond l'apprentissage de ces modèles pré-entraînés afin de palier le problème d'une base de données non suffisamment fournie pour entraîner ses propres modèles. La base de ce transfer learning consiste donc à utiliser ces modèles pré-entraînés comme modèles de base d'une pile de modèles où serait agencé au-dessus un modèle « local », adapté à son propre problème. Il faut bien sûr utiliser des modèles pré-entraînés qui soient compatibles avec les données locales, ce qui est bien le cas ici.

Il existe plusieurs façons de procéder [44].

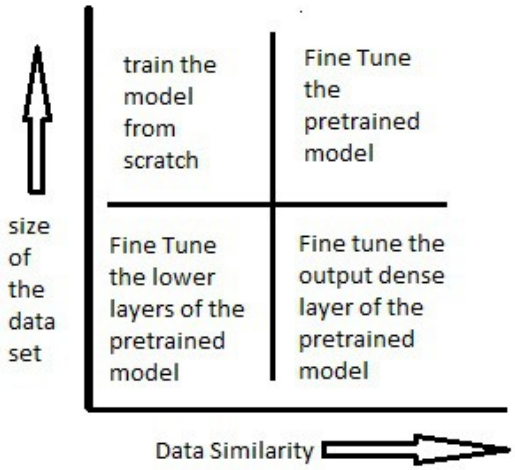
Un modèle de classification possède deux fonctions qui se distinguent au niveau de l'empilement des couches, l'extraction de caractéristiques qui composent les premières couches, et la classification de ces caractéristiques qui constitue les dernières couches dont la dernière est la couche de sortie qui fournit les probabilités.

1- On peut utiliser le modèle pré-entraîné comme mécanisme d'extraction des caractéristiques et supprimer sa couche de sortie puisqu'elle est basée sur le classement des 1000 classes d'ImageNet. Dans notre cas, on n'a que 10 classes. Cette dernière couche n'est donc pas adaptée.

2- On peut entraîner certaines couches et en geler certaines autres de façon à n'entraîner le modèle que partiellement. Par exemple, geler les premières couches et n'entraîner que les dernières couches.

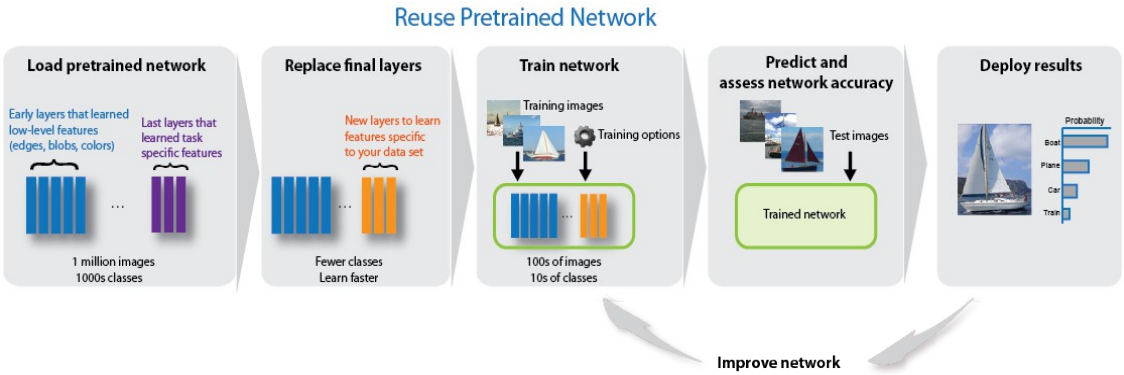
3- On peut également utiliser l'architecture du modèle pré-entraîné en initialisant tous les poids de façon aléatoire et entraîner le modèle sur notre dataset.

En fait, on a la même technique de gel et de réutilisation, mais utilisée à des degrés différents. Tout dépend de deux paramètres, la taille de notre dataset d'entraînement et la similitude de nos données avec les données de pré-entraînement.



Dans notre étude de cas, nous sommes dans le scénario où la similarité des données est forte mais avec peu de données. On peut donc utiliser les modèles pré-entraînés comme extracteurs de caractéristiques et enlever la couche de sortie

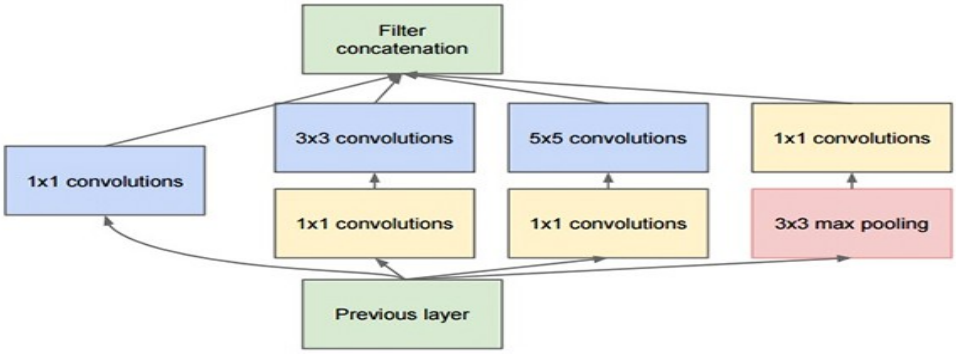
[45]



C. Le modèle Inception V3

Les auteurs du module Inception ont cherché à savoir si au lieu d'approfondir toujours plus les modèles, on ne pouvait pas les élargir sans pour autant diminuer les performances. Ils se sont demandé quels types de convolution on devait choisir pour chaque couche. Doit-on prendre des convolutions 1x1, 3x3, 5x5, etc. voire du pooling ? Et pourquoi ne pas toutes les choisir et laisser le modèle décider ce qui est le mieux ? Cela permettrait en outre au modèle de trouver des caractéristiques locales à l'aide des petites convolutions et des caractéristiques de plus haute abstraction avec des convolutions plus larges.

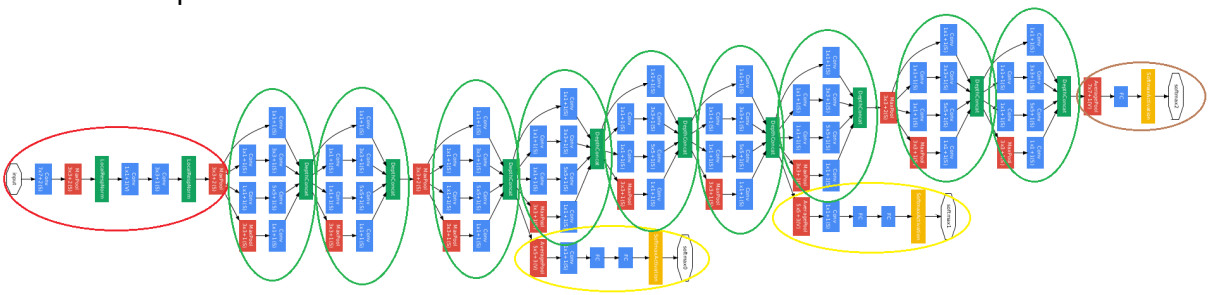
Cela ressemble fort à un apprentissage orienté de sa propre structure. Le modèle commence à choisir quelle structure prendre pour obtenir les meilleurs résultats. Le ResNet était également intéressant car il a commencé à proposer des courts-circuits, une communication ne se faisant pas uniquement entre couches adjacentes, suivi en cela et généralisé par DenseNet. Et si l'on suit la même logique que pour Inception, on devrait pouvoir proposer au réseau un certain nombre de connexions inter-couches afin qu'il garde les plus efficaces.



Les convolutions de tailles différentes permettent d'extraire des inputs, des caractéristiques d'abstraction plus ou moins fines. Cela permet de ne pas se focaliser sur les mêmes genres de détails ou de traits généraux et offre donc une certaine diversité, ce qui améliore la performance du modèle.

Une autre amélioration par rapport aux précédents modèles est qu'il utilise des convolutions 1x1. Auparavant, les modèles effectuaient des convolutions sur les dimensions spatiales et les canaux, en même temps. Avec les convolutions 1x1, le module Inception récupère les corrélations qui ont lieu à travers les canaux, indépendamment des dimensions spatiales. S'en suit après les corrélations à la fois spatiales et à travers les canaux grâce aux convolutions 3x3 et 5x5.

En 2014, Christian Szegedy et son équipe ont donc utilisé ce module Inception dans un modèle, GoogLeNet, dont on peut voir le schéma ci-dessous. Les ellipses représentent les modules Inception :



Ce schéma représente GoogLeNet, c'est-à-dire la version V1 de la famille des réseaux Inception avec laquelle ils ont gagné le concours ILSVRC en 2014. D'autres versions, les V2, V3 et V4 ont suivi depuis. La V2 améliore la V1 grâce à un système de factorisation. La V3 a rajouté des couches de BatchNormalization et a fait disparaître les deux classifieurs auxiliaires qu'on peut voir dans les ellipses jaunes.

La structure de la version V3 est donc un peu différente de celle de la V1. Je ne l'ai pas mise en détail car elle prendrait plusieurs pages. Elle est par contre présente dans le notebook des prototypes.

J'y ai dénombré :

- 94 couches de convolution + activation
- 94 BatchNormalization
- 4 MaxPooling + 9 AveragePooling + 1 GlobalAveragePooling2D
- 15 concaténations
- FC (Dense)

Un module Inception correspond à 6 convolutions + 1 AveragePooling + 1 concaténation, et il y a donc 9 modules.

Les couches de convolution s'occupent des « calculs », sommation des flux entrants pondérés + fonction d'activation. Les BatchNormalization normalisent les sorties des couches de convolution, cela améliore la performance et l'accuracy du modèle. Les concaténations correspondent aux concaténations à l'intérieur des modules Inception. Et enfin, la FC est la dernière couche entièrement connectée qui traite de la classification.

Il s'agit donc d'un modèle avec énormément de couches, et pourtant peu de paramètres, comparé au Perceptron 10C et au CNN du tableau ci-dessous… Généralement, les paramètres se trouvent en grand nombre dans les couches Fully-Connected. On peut le voir avec le Perceptron à 10 FC qui a près de 155 millions de paramètres. Inception V3 n'a qu'une seule couche FC, la dernière de classification.

On peut remarquer également qu'il n'y a aucun dropout.

D. Résultats

Type	Modèle	Paramètres	Loss	Accuracy (%)
Train	Perceptron 2C	19 269 002	14.96	7.21
Train	Perceptron 10C	155 889 226	14.96	7.21
Train	CNN 10C	102 831 658	13.11	18.67
Train Transfer Learning	ResNet50	24 641 930	1.41	54.03
Train T.L.	Inception V3	22 857 002	2.12	25.18
Train T.L.	Xception	21 915 698	2.23	19.52
Classification	ResNet50	25 636 712		72.84
Classification	Inception V3	23 851 784		86.28
Classification	Xception	22 910 480		87.27

J'ai donc utilisé sur le même jeu de données de test, 3 modèles « simples », Perceptron 2 couches, Perceptron 10 couches, et un CNN 10 couches que j'ai entraîné avec les images d'entraînement des 10 catégories choisies au préalable.

Puis j'ai utilisé 3 modèles pré-entraînés (sans top) sur lesquels j'ai greffé un MLP pour l'entraînement aux données locales de l'étude, ResNet50, Inception V3 et Xception.

Enfin, j'ai utilisé directement les modèles pré-entraînés pour classer les données de test sans passer par un entraînement des données locales.

On observe donc des résultats similaires pour les deux perceptrons et une légère amélioration avec le CNN, ce qui est cohérent avec l'historique de ces modèles. (Cela doit être cependant relativisé avec le fait que j'ai utilisé très peu d'images d'entraînement…)

(Il est à noter qu'un perceptron avec quelques couches donne un excellent résultat, 96.16%, avec les petites images du MNIST.)

Grâce au transfer learning, j'ai pu augmenter assez sensiblement l'accuracy et diminuer la perte par rapport aux premiers modèles. On observe que sur ces quelques données, ResNet50 donne de meilleurs résultats que les Inception. Or, c'est le contraire quand je n'utilise pas de transfer learning et que j'utilise directement les modèles pré-entrainés pour classer les données de test. On obtient alors de bien meilleurs résultats, plus conformes à ce qu'on trouve dans la littérature en ce qui concerne la hiérarchie, c'est-à-dire que Xception est meilleur que Inception V3 qui est meilleur que ResNet50 sur les données d'ImageNet.

Il est à noter qu'il est normal d'avoir de moins bons résultats avec la technique de transfer learning que par rapport à la classification car malgré tout, l'entraînement des couches top s'est fait sur peu de données. On devait donc s'attendre à des scores intermédiaires, supérieurs aux premiers modèles, mais inférieurs à ceux de la classification puisque les données de test sont des images d'ImageNet. La classification à partir des modèles pré-

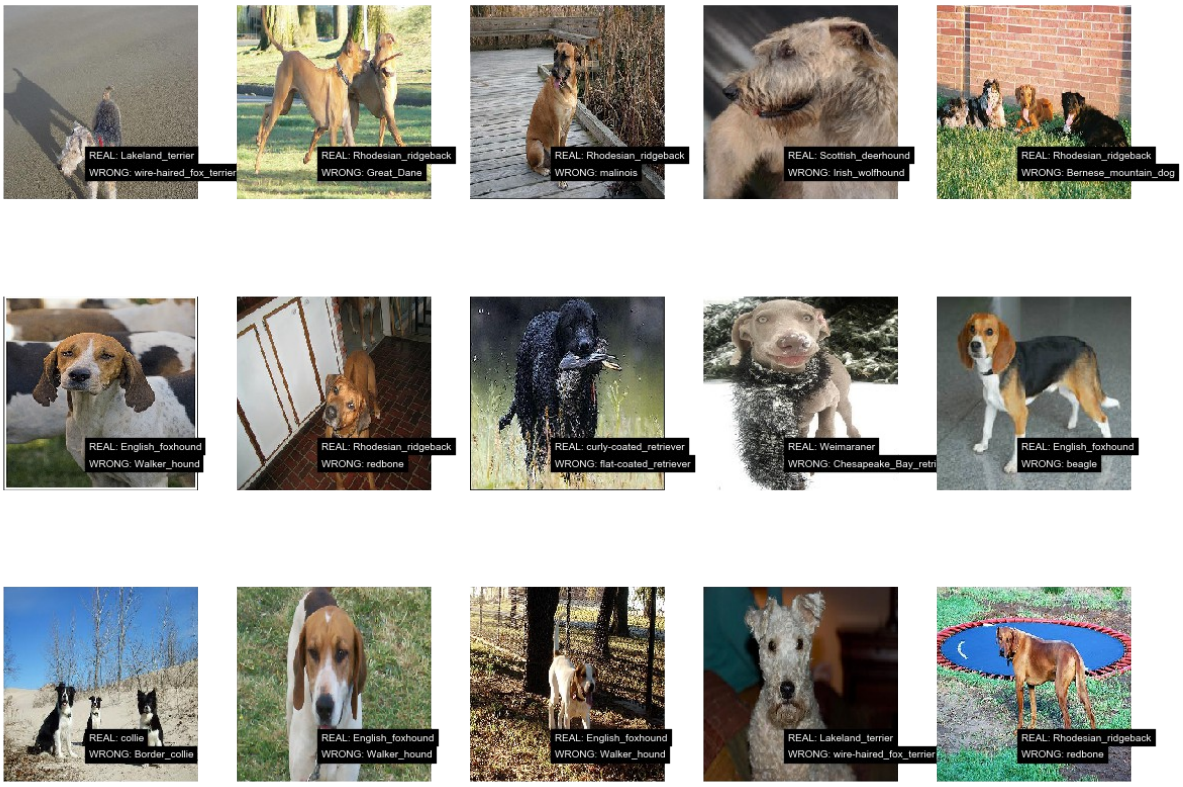
entraînés ne pouvait donner que de meilleurs résultats lorsqu'on compare les tailles des datasets d'entraînement respectifs.

Pour améliorer les résultats, il faudrait bien sûr nettement augmenter le nombre de d'images d'entraînement, et donc disposer aussi de ressources informatiques plus puissantes.

On pourrait également cibler l'augmentation du nombre d'images dans les catégories qui posent problème, en étudiant les erreurs de classification.

E. Erreurs de classification

Pour Inception V3, j'ai illustré les erreurs de classification par 15 chiens mal classés :



Juste pour ces quelques exemples, on voit qu'à 3 reprises, le modèle confond un English Foxhound avec un Walker Hound. Et franchement, moi aussi… ce sont vraiment deux races très proches… On peut remarquer aussi 4 de ces images, il y a plusieurs chiens en même temps. Sur d'autres non présentées ici, on voit souvent la présence d'humains.

Ces exemples sont juste illustratifs, mais il faudrait établir une liste de caractéristiques des images (présence d'humains, présence de plusieurs chiens, taille des chiens etc.) et les croiser dans une matrice de confusion afin de mettre en exergue quelles caractéristiques posent le plus problème aux différents modèles. Une fois ces caractéristiques extraites, il faudrait alors trouver d'autres photos possédant ces caractéristiques afin d'aider les modèles à s'entraîner dessus. Cela devrait améliorer leur performance.

VI. Conclusion

L'accroissement de la complexité technologique au fil des décennies a fourni aux chercheurs des problèmes de plus en plus complexes à résoudre. Avec l'amélioration des connaissances en neurophysiologie, certains chercheurs ont eu l'idée de s'inspirer de la

« machine » la plus performante alors connue, le cerveau. Les sept dernières décennies ont ainsi vu s'enchaîner des avancées tellement importantes dans les technologies artificielles que depuis quelques années les artifices humains ont dépassé les capacités biologiques de leurs créateurs comme cela fut le cas en reconnaissance visuelle d'images ou dans certains jeux de stratégie. Cela ne s'est pas fait de façon linéaire car ces technologies nécessitent des compétences pointues dans un grand nombre de domaines et allient sciences et techniques. Ainsi que nous l'avons vu, sans l'algorithme mathématique et conceptuel de rétropropagation du gradient de l'erreur, les travaux sur les réseaux de neurones ont été sérieusement ralentis pendant près de vingt ans. Et sans la technologie des GPU, des idées comme les capsules de Hinton étaient en latence, faute de disposer d'ordinateurs suffisamment puissants pour les mettre en pratique. Depuis une vingtaine d'années toutefois, les chercheurs disposent d'un arsenal théorique et technique suffisant pour aboutir à d'excellents résultats et les réseaux de neurones ont commencé à être utilisés dans un très large éventail de disciplines, que ce soit en informatique bien sûr mais aussi dans des domaines plus inattendus comme la finance et l'écologie. Leur capacité à dépasser la problématique d'utiliser des fonctions mathématiques littérales comme solutions de problèmes complexes et à traiter des problèmes non linéaires, leur a ouvert la porte à un nombre de plus en plus important d'applications. Les structures rigides des débuts ont laissé la place à une réflexion de plus en plus poussée sur la communication entre couches au fur et à mesure que certaines limites semblaient atteintes. Le terme de réseau prend désormais tout son sens tant sa structure évolue et, si on peut s'exprimer ainsi, apprend. Avec le module Inception, le réseau a en partie appris à choisir la structure, ou du moins les composants qui lui étaient les plus profitables. Et s'ils possèdent une formidable capacité à reconnaître des entités fixes (reconnaissance de textes, d'images, de signaux et de structures en général), les réseaux de neurones pourront peut-être bientôt apprendre à s'auto-gérer et à s'auto-créer, toutefois lorsque cela reste dans les limites d'un apprentissage supervisé.

VII. Annexes

A. Fonctions d'activation

Liste de différentes fonctions d'activation [29]

Identité :

Les fonctions d'activation linéaires comme l'identité font que l'output du neurone est proportionnel à l'activation (somme des flux entrants pondérés). L'output n'est donc pas nécessairement borné.

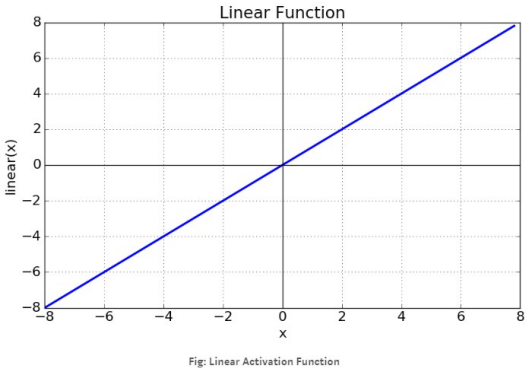


Fig: Linear Activation Function

Sigmoïd :

Les données dans la réalité sont rarement linéaires. Ces fonctions d'activation ne sont en général pas utiles pour ce genre de données. Il est préférable de privilégier les fonctions d'activation non linéaires, comme la fonction logistique, dite aussi sigmoïde.

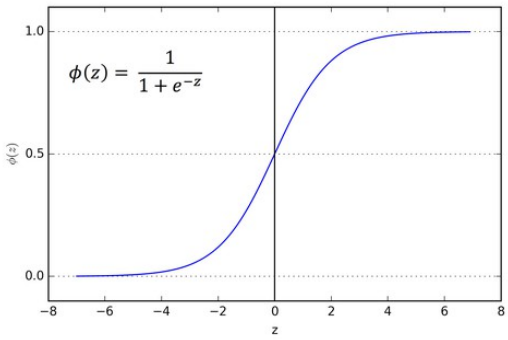


Fig: Sigmoid Function

Cette fonction est utile car elle permet de borner l'output entre 0 et 1, à l'instar des probabilités. Elle est utilisée lors de classifications binaires dans des modèles de régression logistique.

Softmax :

Quand on utilise plus de deux catégories, il est alors préférable d'utiliser la fonction d'activation Softmax qui est la généralisation à plusieurs catégories du classifieur de régression logistique binaire. Comme la somme des scores vaut 1, Softmax permet de mieux visualiser la hiérarchie des scores par rapport à SVM par exemple dont les scores ne sont pas bornés par 1.

Tanh :

De même que Sigmoïd, la fonction de tangente hyperbolique Tanh est sigmoïdale (elle possède une forme en S). La différence réside dans ses outputs qui vont de 0 à 1 pour la sigmoïde mais de -1 à 1 pour Tanh. Il s'ensuit que les inputs négatifs seront mappés de façon plus négative qu'avec la Sigmoïde.

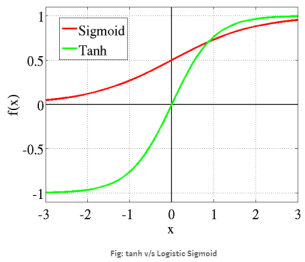


Fig: tanh v/s Logistic Sigmoid

ReLU :

Il est toutefois préférable, d'après [30], d'utiliser la fonction d'activation ReLU plutôt que Tanh ou Sigmoid car elle accélère grandement la convergence de la descente de gradient du fait de sa forme linéaire et non-saturable

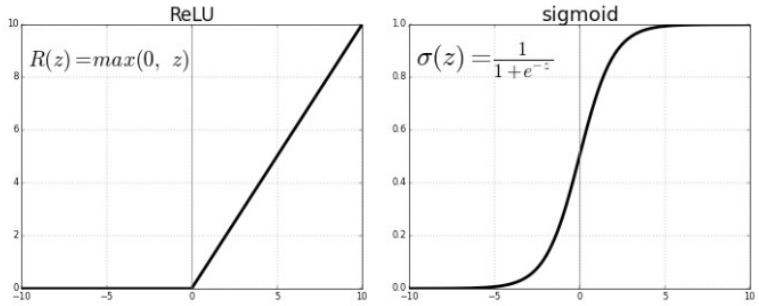


Fig: ReLU v/s Logistic Sigmoid

Le problème est que toutes les valeurs négatives sont annulées, ce qui diminue la capacité du modèle à s'entraîner correctement avec les données puisque les valeurs négatives ne sont pas prises en compte de façon fine. Afin d'y remédier, il a été essayé plusieurs autres fonctions, comme la Leaky ReLU.

Leaky-ReLU, PReLU, Maxout etc. :

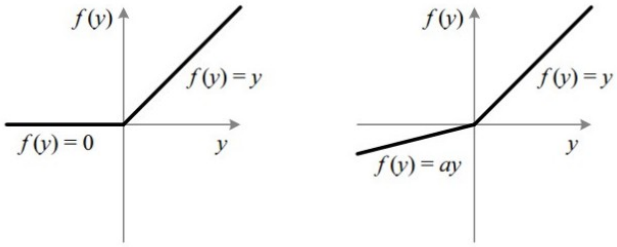


Fig : ReLU v/s Leaky ReLU

En augmentant les outputs possibles sur les valeurs négatives, on prend mieux en compte ces valeurs d'inputs. Leaky ReLU permet ainsi d'avoir un petit gradient (a vaut 0.01) quand le neurone n'est pas actif.

On peut également décliner a, pour en faire un paramètre entraînable. On obtient alors une fonction d'activation appelée Parametric ReLU :

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ ax & \text{otherwise} \end{cases}$$

Quand a ≤ 1, f est équivalent à max(x, ax), ce qui correspond à la fonction Maxout [31].

PReLU et Maxout n'ont pas les inconvénients de ReLU en ce qui concerne la "mort des neurones", qui est évitable en ne prenant pas un learning rate trop élevé.

B. Algorithme d'apprentissage

Illustration d'un algorithme d'apprentissage avec un Perceptron monocouche [32].

Pour les perceptrons multicouches où il existe des couches cachées, il existe des algorithmes plus sophistiqués tels que la rétropropagation. On peut également utiliser la règle du delta (delta rule) pour une fonction non linéaire et dérivable.

Données :

z, vecteur d'entrée du Perceptron

y, vecteur de sortie du Perceptron tel que y = f(z)

D = {(x_1, d_1), …, (x_s, d_s)} est l'ensemble d'entraînement à s éléments où

- x_j est un vecteur d'entrée à n dimensions
- d_j est la valeur de sortie désirée pour cet input x_j

$x_{j,i}$ est la valeur de la i-ème dimension du j-ème vecteur d'entrée d'entraînement

$x_{j,0} = 1$

w_i est la i-ème valeur du vecteur des poids, à multiplier avec la valeur de la i-ème dimension du vecteur d'entrée.

Du fait que $x_{j,0} = 1$, w_0 est un biais utilisé à la place du biais constant b.

$w_i(t)$ est la valeur du poids i à l'instant t.

Etapes :

1- Initialisation des poids et du seuil (correspondant aujourd'hui au biais). Les poids peuvent être initialisés à 0 ou à une petite valeur aléatoire, négative ou positive.

2- Pour chaque élément d'entraînement, étapes suivantes a et b

a- Calculer la sortie

$$y_j(t) = f[w(t) \cdot x_j]$$

$$= f[w_0(t) x_{j,0} + w_1(t) x_{j,1} + \ldots + w_n(t) x_{j,n}]$$

b- Mettre à jour les poids

$$W_i(t+1) = w_i(t) + (d_j - y_j(t)) x_{j,i}$$ pour chaque dimension $0 \leq i \leq n$

1- L'étape 2 peut être répétée jusqu'à ce que l'erreur

$\frac{1}{s}\sum_{j=1}^{s} |d_j - y_j(t)|$ soit inférieure à un pallier initialement déterminé ou jusqu'à un certain nombre d'itérations ait été effectuée.

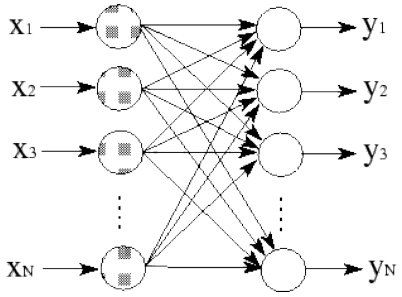
L'algorithme met à jour les poids après les étapes 2a et 2b. Ces poids sont immédiatement modifiés plutôt que d'attendre que tous les éléments d'entraînement aient passé les étapes.

C. Types de connexions entre couches

Une part de la topologie d'un réseau réside dans la communication entre couches. A l'origine, avec un TLU ou un Perceptron, en supposant que parler de réseau ait eu un sens dans ce contexte, on pouvait considérer qu'on avait affaire à un réseau à une couche composée d'un unique neurone. Puis, en ajoutant des neurones au sein d'une même couche et en multipliant les couches, on a alors construit des réseaux de plus en plus complexes. On a pu ainsi voir une certaine évolution au fil du temps dont certaines structures ont été réutilisées comme des sortes de blocs.

Réseau monocouche :

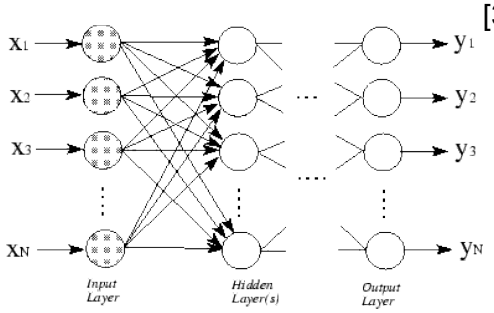
[33]



La couche d'entrée n'est pas comptabilisée dans le nombre de couches d'un réseau. On a donc ici une unique couche, la couche de sortie où sont fait les calculs. Dans ce cas particulier de type de réseau, tous les inputs sont connectés à tous les neurones de la couche de sortie et aucun neurone de sortie ne communique avec les autres neurones de sa couche.

Réseau multicouche :

[34]



Ajouter des couches devient nécessaire quand on veut approcher de fonctions de plus en plus complexes et non linéaires. Ce type de réseaux se distingue par l'ajout de couches cachées (hidden layer).

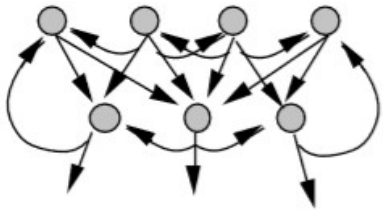
Toutefois, tous les neurones d'une couche ne sont pas nécessairement connectés à tous les neurones de la couche précédente.

Réseau multicouche à connexions locales : [35]

Si pour les couches « Fully-Connected » (FC), tous les neurones d'une couche sont bien connectés à tous les neurones de la couche précédente, ce n'est pas le cas pour les couches de Convolution où un neurone d'une couche n'est connecté qu'à quelques-uns des neurones de la couche précédente. Il s'agit d'une évolution car au début, il n'y avait que des couches entièrement connectées.

Réseau multicouche à connexions récurrentes :

Pour les réseaux, il y a un transport « en arrière » de l'information par rapport au sens de propagation « en avant » défini dans la plupart des réseaux neuronaux.

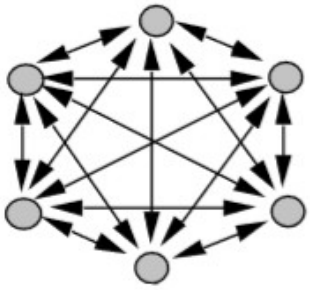


Réseau à connexions récurrentes

On peut noter les connexions entre les neurones d'une même couche.

Réseau multicouche à connexions complètes :

C'est le réseau le plus général possible, tous les neurones sont connectés à tous les autres neurones du réseau ainsi qu'à lui-même.



Réseau à connexions complètes

D. Domaines d'utilisation des réseaux neuronaux

- Aérospatial : pilotage automatique, simulation de vol…
- Défense : guidage de missiles et suivi de cibles, reconnaissance du visage, radar, suppression du bruit…
- Electronique : prédiction de la séquence d'un code, vision machine, synthétiseur vocal…
- Finance : prévision du coût de la vie
- Secteur médical : analyse EEG et ECG
- Télécommunications : compression des données…
- Etc.

Références

[1] Daniel Graupe, *Principles of artificial neural networks*, 2007, ADVANCED SERIES IN CIRCUITS AND SYSTEMS – Vol. 6, World Scientific.

[2] Kevin Gurney, *An introduction to Neural Networks*, 1997, CRC Press, p. 13

[3] Frank Rosenblatt, *The perceptron, a probabilistic model for information storage and organization in the brain*, 1958, Psychol. Rev. 65, pp. 386–408.

[4] Claude Touzet, *LES RESEAUX DE NEURONES ARTIFICIELS, INTRODUCTION AU CONNEXIONNISME : COURS, EXERCICES ET TRAVAUX PRATIQUES. EC2*, 1992, Collection de l'EERIE, N. Giambiasi. Disponible sur : https://hal-amu.archives-ouvertes.fr/hal-01338010 (consulté le 29/11/2017)

[5] R. J. Lee, *Generalization of learning in a machine*, 1959, Proc. 14th ACM National Meeting, S8ptember.

[6] Bernard Widrow & Marcian E. "Ted" Hoff, *Adaptive switching circuits*, 1960, Proc. IRE WESCON Conf., New York, pp. 96–104.

[7] Bernard Widrow & Rodney Winter, *Neural nets for adaptive filtering and adaptive pattern recognition*, 1988, Computer 21, pp. 25-39.

[8] John J. Hopfield, *Neural networks and physical systems with emergent collective computational abilities*, 1982, Proceedings of the National Academy of Sciences 79, pp. 2554–2558.

[9] http://william.arrouy.free.fr/neural/neu6.html

[10] David E. Rumelhart, Geoffrey. E. Hinton et Ronald. J. Williams, *Learning internal representations by error propagation*, 1986, pp. 318–362 in Parallel Distributed Processing: Explorations in the Microstructures of Cognition, eds. Rumelhart, D. E. and McClelland, J. L. MIT Press, Cambridge, MA.

[11] Robert Hecht-Nielsen, *Counter propagation networks*, 1987, Applied Optics 26, pp. 4979–4984.

[12] Yann LeCun et al., *Gradient-Based Learning Applied to Document Recognition*, 1998, http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

[13] Alex Krizhevsky , *ImageNet Classification with Deep Convolutional Neural Networks*, 2012, https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[14] Matthew D. Zeiler, Rob Fergus, *Visualizing and Understanding Convolutional Networks*, 2013, https://arxiv.org/abs/1311.2901

[15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, *Going Deeper with Convolutions*, 2014, https://arxiv.org/abs/1409.4842

[16] Karen Simonyan, Andrew Zisserman, *Very Deep Convolutional Networks For Large-Scale Image Recognition*, 2015, Visual Geometry Group, Department of Engineering Science, University of Oxford https://arxiv.org/pdf/1409.1556.pdf

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, *Deep Residual Learning for Image Recognition*, 2015, https://arxiv.org/abs/1512.03385

[18] Gao Huang, Zhuang Liu, Kilian Q. Weinberger, Laurens van der Maaten, *Densely Connected Convolutional Networks*, 2016, https://arxiv.org/abs/1608.06993

[19] Sara Sabour, Nicholas Frosst, Geoffrey E. Hinton, *Dynamic Routing Between Capsules*, 2017, https://arxiv.org/abs/1710.09829

[20] Warren McCulloch & Walter Pitts, *A Logical Calculus of Ideas Immanent in Nervous Activity*, 1943, Bulletin of Mathematical Biophysics 5: pp. 115-133.

[21] Hisao Ishibuchi, *Computational Intelligence - Volume I*, 2015, EOLSS Publications, p. 31

[22] Donald Hebb, *The Organization of Behavior*, 1949, John Wiley, New York.

[23] Daniel Graupe, *Ibid.*, pp. 23-24

[24] Hisao Ishibuchi, *Ibid.* pp. 41-42

[25] John J. Hopfield, David W. Tank, *Neural Computation of Decisions in Optimization Problems*, 1985, Biological Cybernetics, 52, 141-152.

[26] Kevin Gurney, *Ibid.* p. 140

[27] Andrej Karpathy, notes sur les cours de Stanford, 2017, disponible sur : http://cs231n.github.io/convolutional-networks/

[28] https://blog.cloudera.com/blog/2017/06/deep-learning-on-apache-spark-and-hadoop-with-deeplearning4j/

[29] https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

[30] Andrej Karpathy, notes sur les cours de Stanford, 2017, disponible sur : http://cs231n.github.io/neural-networks-1/

[31] https://en.wikipedia.org/wiki/Rectifier_(neural_networks)

[32] https://en.wikipedia.org/wiki/Perceptron

[33] http://www.ece.utep.edu/research/webfuzzy/docs/kk-thesis/kk-thesis-html/node15.html

[34] http://www.ece.utep.edu/research/webfuzzy/docs/kk-thesis/kk-thesis-html/node16.html

[35] Claude Touzet, *Ibid.* pp. 24-25

[36] « Anonymous authors » (Under review as a conference paper at ICLR 2018) https://openreview.net/pdf?id=HJWLfGWRb

[37] Max Jaderberg, Karen Simonyan, Andrew Zisserman, Koray Kavukcuoglu, *Spatial Transformer Networks*, 2015, https://arxiv.org/abs/1506.02025

[38] Dario Amodei, Rishita Anubhai, Eric Battenberg *et al.*, *Deep Speech 2: End-to-End Speech Recognition in English and Mandarin*, 2015, https://arxiv.org/abs/1512.02595

[39] Eric Battenberg, Jitong Chen, Rewon Child, *Exploring Neural Transducers for End-to-End Speech Recognition*, 2017, https://arxiv.org/abs/1707.07413

[40] Batuhan Balci, Dan Saadati, Dan Shiferaw, *Handwritten Text Recognition using Deep Learning*, 2017, http://cs231n.stanford.edu/reports/2017/pdfs/810.pdf

[41] Christian Bartz, Haojin Yang, Christoph Meinel, *STN-OCR: A single Neural Network for Text Detection and Text Recognition*, 2017, https://arxiv.org/abs/1707.08831

[42] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna, *Rethinking the Inception Architecture for Computer Vision*, 2015, https://arxiv.org/abs/1512.00567

[43] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, Yoshua Bengio, *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*, 2014, https://arxiv.org/abs/1412.3555

[44] Dishashree Gupta, *Transfer learning & The art of using Pre-trained Models in Deep Learning*, 2017, https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-trained-model/

(en fait, l'auteur a copié le code de la documentation de Keras : https://keras.io/applications/)

[45] https://fr.mathworks.com/help/nnet/ug/pretrained-convolutional-neural-networks.html?requestedDomain=www.mathworks.com#mw_6101bf28-05dc-4d23-bc48-1d2ac270efee