الجـــــــــــــــــــمـهـوريـــــة الجـــــــــــــزائـريـة الديمـــــــقـراطيـة الـشـــــــــعبيـة

Republique Algerienne Democratique Et Populaire

وزارة التـــــــــــــــــــــــــــعـليـم العـــــــالي و البـــــــــــحث العـــــلـمـــي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جـــــــــــــــامعة غـردايـــــة

Université de Ghardaia

كـليـة العـلـوم والتـكـنـلـوجيـا

Faculté des Sciences et de Technologie

قسـم الرياضيات و الاعلام الآلـي

Département des Mathématiques et Informatique

Mémoire

Présenté pour l'obtention du **diplôme** de **MASTER**

En : Informatique

Spécialité : Systèmes Intelligents pour l'Extraction de Connaissances (SIEC)

Par : Hind FIHAKHIR

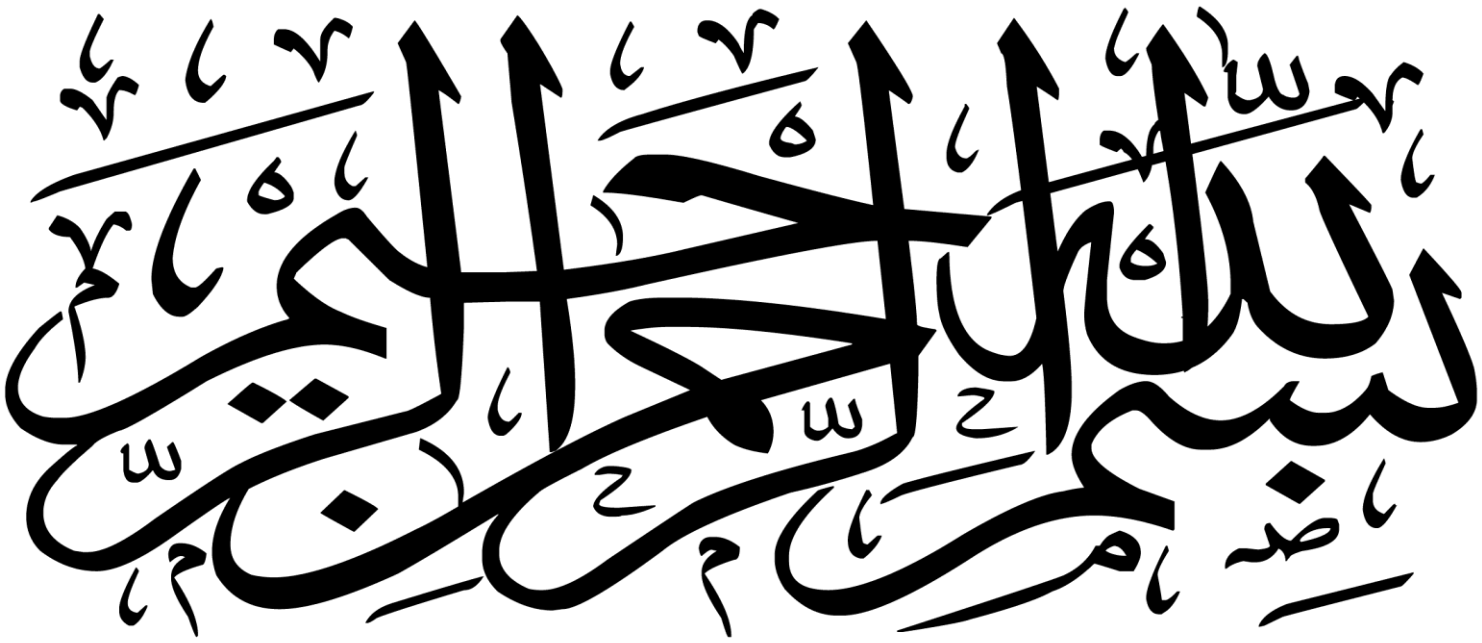
Sujet

Classification du texte avec deep learning

Devant le jury composé de :

Dr. Slimane OULED NAOUI	Maitre-assistant	Univ. Ghardaia	Président
Dr. Slimane BELLAOUAR	Maitre-assistant	Univ. Ghardaia	Encadreur
M. Youcef MAHDJOUB	Maitre-assistant	Univ. Ghardaia	Examinateur
Prof. Djeloul ZIADI	professeur	Univ. Ghardaia	Examinateur

Année Universitaire 2017/2018


بسم الله الرحمن الرحيم

Résumé

Résumé

L'apprentissage automatique s'est proposé comme une solution au défi résultant du phénomène de l'explosion des données.

Le deep learning (l'apprentissage profond), est un sous-domaine de l'apprentissage automatique qui exploite les réseaux de neurones artificiels.

Les réseaux de neurones convolutionnels et récurrents sont deux types de réseaux de neurones artificiels. Ces réseaux ont la capacité d'extraire les descripteurs à partir des données brutes.

Dans ce mémoire, on s'intéresse à la classification de texte avec le deep learning. Notre travail s'est focalisé sur l'étude de classification de texte en utilisant les architectures convolutionnels (Convolutional Neural Network, CNN) et récurrente (Recurrent Convolutional Neural Network, RCNN).

Enfin, nous avons conduit une étude empirique comparative des architectures CNN et RCNN en utilisant le dataset IMDb. Les résultats révèlent l'impact visible du deep learning sur la tâche de classification de texte.

***MOTS-CLÉS** : Apprentissage automatique, Deep learning, Classification du texte, Réseau de neurones convolutionnel (CNN), Réseau de neurones récurrent (RNN).*

Abstract

Machine learning is proposed as a solution to the resulting challenge of the data explosion phenomenon.

Deep learning is a sub-field of machine learning that exploits artificial neural networks.

Convolutional and recurrent neural networks are two types of artificial neural networks. These networks have the ability to extract features from raw data.

Our task is to focus on studding text classification using covolutional neural networks (CNN) and recurrent convolutional neural network (RCNN).

Finally, we conduct an empiric comparative study of CNN and RCNN, architecture using the IMDb dataset. The results reveal a visible impact of deep learning on text classification.

KEYWORDS: Machine learning, Deep learning, Text classification, Convolutional neural network (CNN), Recurrent neural networks (RNN).

ملخص

تم اقتراح التعلم الآلي كحل لظاهرة انفجار البيانات.

التعلم العميق، فئة من التعلم الآلي الذي يستغل شبكات العصبونات الاصطناعية.

الشبكات العصبية التلافيفية والمتكررة هي نوعان من الشبكات العصبية الاصطناعية، هذه الشبكات لديها القدرة على استخراج الميزات من البيانات الخام.

في هذه المذكرة، نحن مهتمون بتصنيف النص مع التعلم العميق. ركز عملنا على دراسة تصنيف النص باستخدام البنيتين التلافيفية والمتكررة.

وأخيرًا، أجرينا دراسة تجريبية مقارنة على بنيتين CNN و RCNN باستخدام مجموعة بيانات IMDb . تكشف النتائج التأثير المرئي للتعلم العميق في مهمة تصنيف النص.

الكلمات المفتاحية : *التعلم الآلي، التعلم العميق، تصنيف النص، الشبكات العصبية التلافيفية (CNN)، الشبكات العصبية المتكررة (RNN).*

Table des matières

Liste des Figures

Liste des Tables

Remerciement

Au premier temps, nous remercions Allah qui nous a aidé à réaliser ce travail, mes parents, et qui a été avec nous en tout le moment.

Nous remercions également notre encadreur Monsieur Slimane BELLAOUAR pour l'aide et de ses conseils pour compléter ce mémoire dans son intégralité.

Nous tenons à remercier les jurys qui ont suivi ce travail, et les responsables de la formation Système Intelligent pour Extraction des Connaissance « SIEC », pour avoir assuré cette formation, et toute l'équipe pédagogique de Département des Mathématiques et Informatique de notre université.

Introduction générale

Introduction générale

Notre ère de révolution numérique est estampé par le phénomène du déluge informationnel, notamment, le type de données texte. Ainsi , le traitement manuels de ce type de données devient coûteux, voire impossible. Le recours à des méthodes intelligentes d'analyse de données s'avère indispensables.

La classification du texte est une technique plus répandue dans l'apprentissage automatique dont sa vocation est multiple. Elle peut être utilisée dans la recherche sur le Web, le filtrage d'informations et l'analyse des sentiments.

Le deep learning est un sous-domaine de l'apprentissage automatique, et l'apprentissage automatique est à son tour un sous-domaine de l'intélligence artificielle. Ces trois domaines sont comme un ensemble de poupées Russes imbriquées les unes dans les autres.

Dans notre mémoire, nous nous intéressons a le problème de classification du texte en utilisant la nouvelle technique du deep learning. Nous commençons par dévoiler le secret du deep learning en se focalisant principalement sur les architectures convolutionnelles (convolutional neural network, CNN) et les architectures récurrentes (recurrent convoutional neural network, RCNN).

Par la suite, nous montrons comment utiliser ces architectures (CNN et RCNN) dans la classification du texte. Enfin, nous conduisons une étude empirique comparative impliquant les architectures CNN et RCNN et leur impact sur la tâche de classification du texte.

Ce mémoire est composé de quatre chapitres :

Le chapitre 1 commence par une définition de l'apprentissage automatique ainsi que ses types. Ensuite, il introduit le concept du deep learning. Enfin, les réseaux de neurones sont présentés. Ils sont considérés comme l'approche la plus réussie du deep learning pour le moment.

Le chapitre 2 s'intéresse à la présentation des deux types de réseaux de neurones, à savoir convolutionnels (Convolutional Neural Network, CNN) et récurrents (Recurrent Neural Network, RNN).

Le chapitre 3 se veut une revue de la littérature de la classification du texte avec le deep learning. Après l'introduction de quelques concepts préliminaires, nous présentons deux

travaux connexes utilisant les CNN (Kim, 2014) et les RCNN (Siwei, et al., 2015) dans la classification du texte.

Le chapitre 4 se présente sous forme d'une étude empirique comparative des deux travaux de classification du texte abordés dans le chapitre 3. Le dataset IMDb est utilisé comme des données d'application.

La conclusion, vient pour clôturer notre travail avec un bilan général de notre travail.

Chapitre 1

Apprentissage automatique et deep learning

1.1 Introduction

Le deep learning est un ensemble de méthodes d'apprentissage automatique développées pour les machines. Le deep learning est un sous-ensemble d'algorithmes d'apprentissage automatique qui reconnaît très bien les modèles. Pour bien comprendre le deep learning, il faut avoir une solide compréhension des principes de base de l'apprentissage automatique (Deng & Yu, 2014). Le deep learning excelle dans la reconnaissance des objets tels qu'ils sont mis en œuvre en utilisant 3 couches ou plus de réseaux de neurones artificiels où chaque couche est responsable de l'extraction d'une ou plusieurs caractéristiques de l'entrée.

Dans ce chapitre, nous définissons à la fois l'apprentissage automatique et ses types, le deep learning et les réseaux neurones artificiels qui sont à la base de le deep learning.

1.2 Apprentissage automatique

Dans cette section, nous présentons l'apprentissage automatique et ses types.

1.2.1. Définition

En 1959 **Arthur Samuel** a défini l'apprentissage automatique comme suit :

« Domaine d'études qui permet aux ordinateurs d'apprendre sans être explicitement programmés »

Bien que l'apprentissage automatique soit un domaine de l'informatique, il diffère des approches informatiques traditionnelles. En effet dans cette dernière, les algorithmes sont des ensembles d'instructions explicitement programmées utilisées par les ordinateurs pour calculer ou résoudre des problèmes. Les algorithmes d'apprentissage automatique permettent aux ordinateurs de s'entraîner sur les données d'entrées et utilisent l'analyse statistique pour produire des valeurs qui se situent dans une plage spécifique.

1.2.2 Types de l'apprentissage automatique

Dans l'apprentissage automatique, deux types d'apprentissage automatique les plus largement adoptées sont apprentissage supervisé qui forme des algorithmes basés sur des données d'entrée et de sortie étiquetées par l'homme et l'apprentissage non supervisé qui ne fournit pas à l'algorithme des données étiquetées.

1.2.2.1 Apprentissage supervisé

Dans l'apprentissage supervisé, l'ordinateur est fourni avec des exemples d'entrées qui sont étiquetés avec les sorties souhaitées. Le but de cette méthode est que l'algorithme puisse «apprendre» en comparant sa sortie réelle avec les sorties «apprises» pour trouver des erreurs et modifier le modèle en conséquence. L'apprentissage supervisé utilise donc des modèles pour prédire les valeurs d'étiquettes sur des données non étiquetées (Ceri, et al., 1998).

1.2.2.2 Apprentissage non supervisé

Dans l'apprentissage non supervisé, les données sont non étiquetées, de sorte que l'algorithme d'apprentissage trouve tout seul des points communs parmi ses données d'entrée. Les données non étiquetées étant plus abondantes que les données étiquetées, les méthodes d'apprentissage automatique qui facilitent l'apprentissage non supervisé sont particulièrement utiles.

L'objectif de l'apprentissage non supervisé peut être aussi simple que de découvrir des modèles cachés dans un ensemble de données.

1.3 Deep learning

Le deep learning a été introduit par la première fois par Dechter & Pearl, (1986). Le deep learning est une classe de techniques d'apprentissage automatique qui exploitent les réseaux de neurones artificiels (Bengio, 2009; Deng & Dong, 2014). Un modèle de deep learning a la capacité d'extraire des descripteurs à partir des données brutes grâce aux multiples couches de traitement composé les multiples transformations linéaires et non linéaires(Schmidhuber, 2014).

1.4 Réseaux de neurones artificiels

Bien que le deep learning puisse théoriquement s'appliquer à n'importe quel type de modèle, presque toutes les approches de le deep learning jusqu'à présent reposent sur des réseaux neurones artificiels (Arnold, 2013).

Les réseaux de neurones artificiels (Bishop & Christopher, 1995) regroupent une grande variété de modèles qui utilisent les neurones comme unité de calcul élémentaire. Bien que cette classe de modèles ait été historiquement inspirée des processus biologiques, elle fait maintenant partie intégrante du cadre mathématique d'apprentissage automatique. Les réseaux de neurones peuvent être utilisés dans des contextes supervisés pour la classification de la

régression, dans des contextes non supervisés pour la réduction de la dimensionnalité et les représentations d'apprentissage, et peuvent être interprétés dans une perspective probabiliste.

1.4.1 Neurone artificiel

Avant de commencer à définir le neurone artificiel qui est la base de tous les réseaux de neurones artificiels. D'abord se connaître la structure de neurone biologique.

1.4.1.1. Inspiration biologique

Les systèmes biologiques sont capables d'effectuer des calculs très complexes pour survivre dans leur environnement, trouver des aliments ou échapper aux prédateurs. Ces comportements complexes sont contrôlés par un système nerveux composé de cellules nerveuses ou de neurones. L'une des propriétés les plus remarquables de ces systèmes est leur évolutivité de quelques centaines de neurones à des milliards de neurones (environ 85 milliards dans le cerveau humain). Pour être plus précis, il semble que les neurones puissent être combinés de sorte qu'une augmentation du nombre de neurones entraîne une augmentation des capacités cognitives (Huxley, et al., 1952; Herculano-Houzel, 2009).

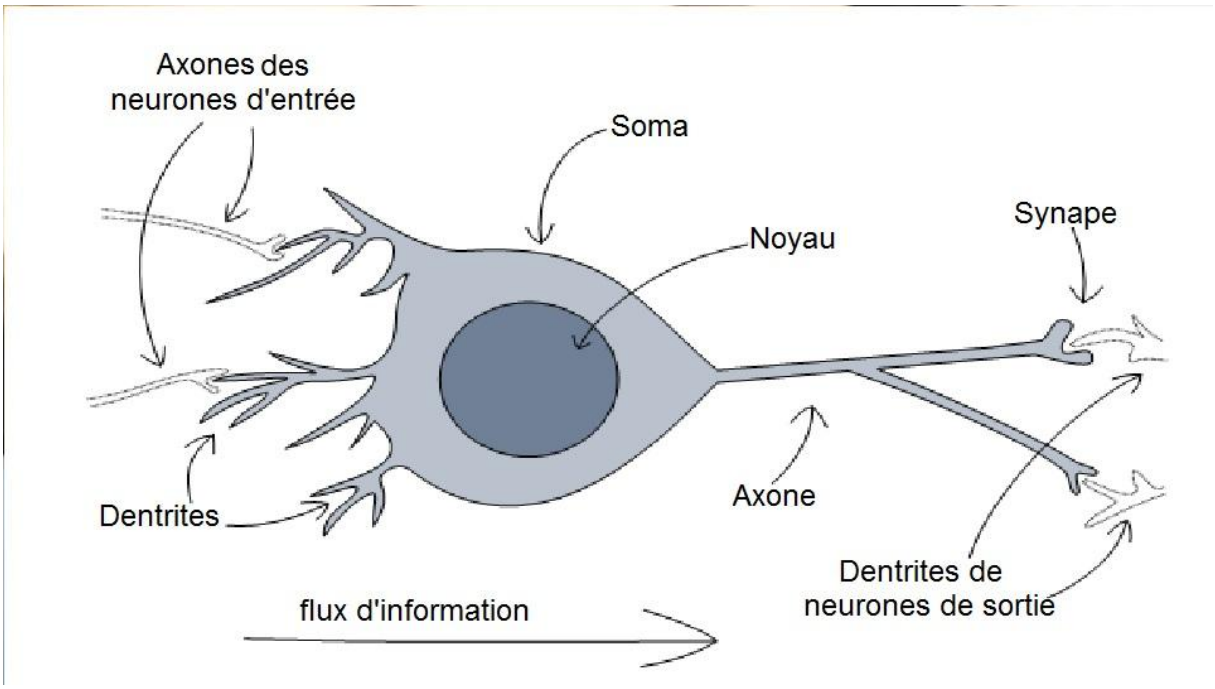


Figure 1.1 : La structure d'un neurone biologique extrait de (Arnold, 2013).

La Figure 1.1 décrit l'architecture générale d'un neurone biologique. L'information provient des neurones d'entrée sous la forme de potentiels d'action. Si le neurone reçoit suffisamment

de potentiels d'action de ses neurones présynaptiques, il émet un pic, envoyant un potentiel d'action à travers son axone aux neurones post-synaptiques.

1.4.1.2 Modèle des neurones artificiels

Afin de transmettre l'information, les neurones artificiels ont une valeur d'activation[1].

Aux fins du calcul, un neurone artificiel (la Figure 1.2) a des connexions pondérées à un ensemble de neurones d'entrée. Les neurones d'entrée peuvent être vus comme un vecteur $X = x_1, x_2, \dots, x_D$ où X est un vecteur de dimension D, et x_i correspond à l'activation D i-ème neurone d'entrée. y est la valeur d'activation d'un neurone peut être calculée étant donné les activations d'entrée x_i et les poids de connexion w_i selon

$$y = \varphi\left(b + \sum_i w_i . x_i\right)$$

Où φ s'appelle **la fonction d'activation** et b est appelé le biais du neurone y (voir section 1.4.3).

Le terme $b + \sum_i w_i . x_i$ pris en entrée de φ s'appelle la pré-activation. Le biais peut être considéré comme un poids normal $w_0$connecté à un neurone d'entrée x_0 tel que $x_0 = 1$. Par conséquent, un neurone effectue un produit scalaire entre l'extension.

[1] Cette valeur d'activation peut être considérée comme jouant un rôle similaire à la fréquence de la libération d'un neurone biologique.

$$y = \varphi\left(b + \sum_{i=1}^{n} w_i . x_i\right)$$

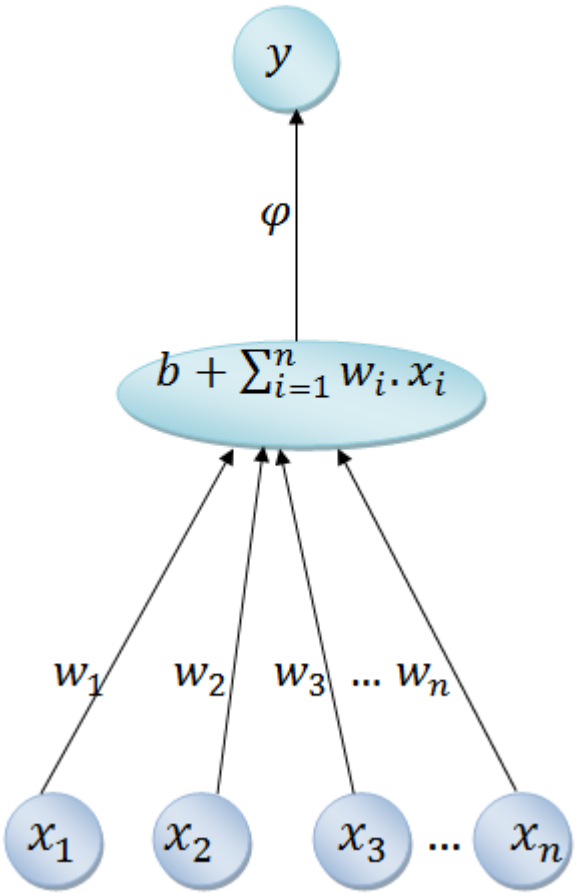


Figure 1.2 : Propriétés de calcul d'un neurone artificiel.

La Figure 1.2 illustre Propriétés de calcul d'un neurone artificiel. L'activation d'un neurone est calculée comme une somme pondérée des activations des neurones d'entrée, transformées par une fonction d'activation φ Les poids des connexions déterminent l'influence d'un neurone d'entrée sur le neurone de sortie.

1.4.2. Réseaux de neurones Feed-forward

Les neurones artificiels ne peuvent faire qu'un simple calcul par eux-mêmes. Cependant, ils peuvent être disposés dans des réseaux de neurones pour effectuer des opérations plus complexes. Ces réseaux peuvent ensuite être utilisés pour calculer les activations d'un ensemble de neurones de sortie $Y = (y_1, y_2, \ldots y_D)$ étant donné les activations des neurones

d'entrée $X = (x_1, x_2, \dots, x_D)$. Le calcul implique généralement un ensemble de neurones cachés h qui effectuent des calculs intermédiaires (Rojas, 1996).

Bien que les neurones puissent en théorie être arrangés de façon assez arbitraire, ils sont souvent disposés en pratique dans un graphe acyclique, ce qui signifie que l'entrée d'un neurone ne dépend pas de sa sortie, même indirectement. Les réseaux neuronaux organisés avec une telle topologie sont appelés réseaux de neurones feed-forward car les activations peuvent être propagées vers l'avant dans le réseau. En revanche, les réseaux de neurones récurrents peuvent contenir des connexions cycliques. Les réseaux de neurones récurrents sont potentiellement meilleurs pour modéliser les systèmes dynamiques, mais la présence de cycles rend l'entraînement beaucoup plus difficile. La Figure 1.3 donne des exemples pour un réseau de neurones récurrent général et un réseau feed-forward(Arnold, 2013).

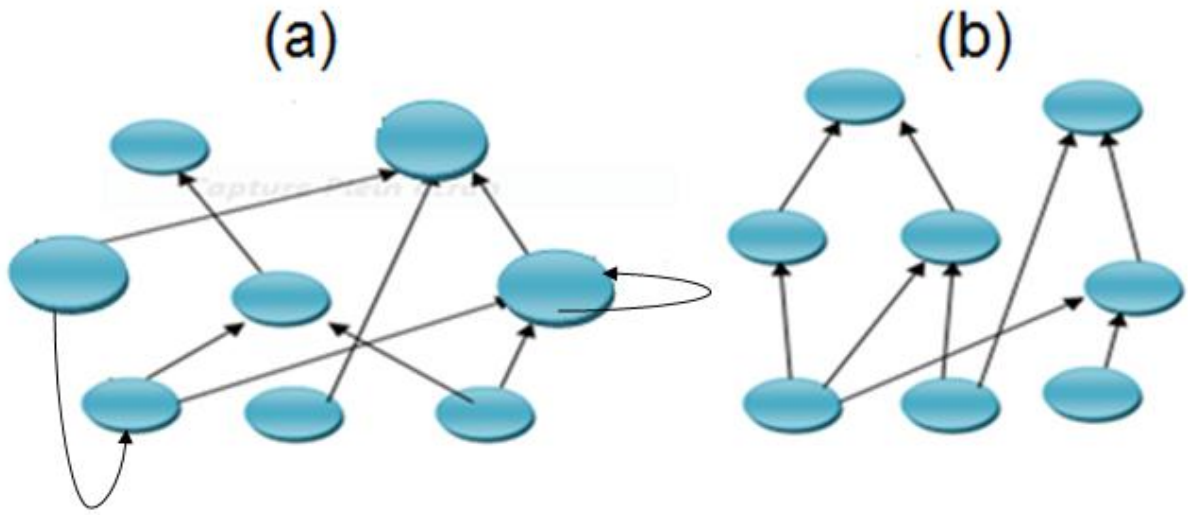


Figure 1.3: Différentes topologies des réseaux de neurones: un réseau de neurones récurrent (a) et un réseau de neurones feed-forward. (b)

Considérons maintenant les réseaux feed-forward où les neurones sont organisés en couches. Dans cette terminologie, le vecteur d'entrée x est la couche d'entrée l_0. Les couches suivantes l_k regroupent alors les neurones qui ne reçoivent que l'entrée des neurones de la couche précédente l_{k-1}. Ceci rend possible le calcul d'activations neurales de manière anticipée, par couches. Les poids utilisés pour calculer l'activation d'une couche l_k à partir des activations d'une couche l_{k-1} forment alors une matrice W où w_{ij} donne le poids de connexion du $i^{ième}$ neurone de la couche l_{k-1} au $j^{ième}$ neurone de la couche l_k. Pour deux couches suivantes x et y, la règle de calcul devient :

$$\forall j, Yj = \varphi\left(b_j + \sum_i w_{ij}.x_i\right)$$

Où la fonction φ est la fonction d'activation.

Lorsqu'un réseau de neurones implique plus de couches que les couches d'entrée et de sortie, le réseau est appelé un réseau de neurones à multi-couches. Les neurones cachés sont alors disposés en plusieurs couches cachées, comme le montre la Figure 1.4.

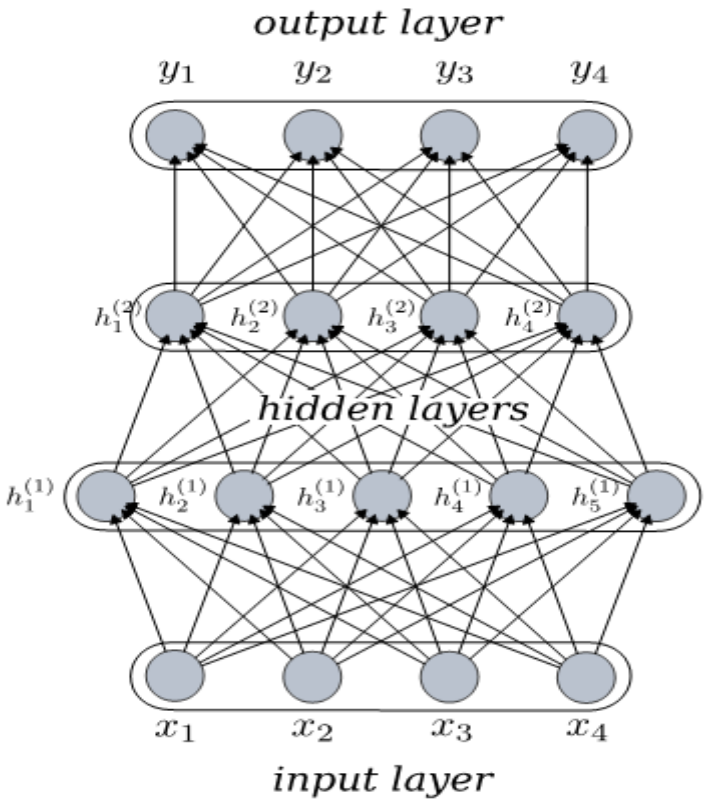


Figure 1.4: Un réseau de neurones multicouche. Les activations peuvent être propagées couche par couche de la couche d'entrée x vers la couche de sortie y.

1.4.3 Fonctions d'activation

Les fonctions d'activation son utilisées pour déterminer la sortie du réseau de neurones il existe plusieurs choix communs de fonction d'activation φ, chacun adapté à différentes applications. Jusqu'à présent, nous avons considéré une activation indéfinie φ (Kriesel, 2005).

Les fonctions d'activation peuvent être fondamentalement divisées en deux types:

- Fonction d'activation linéaire

- Fonctions d'activation non-linéaires.

1.4.3.1 Fonction d'activation linéaire

Activation linéaire Une première possibilité consiste à utiliser la fonction d'identité $\varphi(x) = x$ comme fonction d'activation. Une couche effectue ensuite une opération linéaire $Wx + b$ sur le vecteur d'entrée x.

Comme vous pouvez le voir la fonction est une ligne ou linéaire. Par conséquent, la sortie des fonctions ne sera pas confinée entre une gamme.

1.4.3.2 Fonctions d'activation non-linéaires

Fonction Heaviside Step : est définie comme suit

$$h(x) = \begin{cases} 0 & if\ x < 0 \\ 1 & if\ x \geq 0 \end{cases}$$

Dans un réseau de neurones, le résultat est alors 1 ou 0 selon le signe de la pré-activation $Wx + b$. Cela correspond à une partition de l'espace d'entrée $\mathbb{R}^D$ selon un hyperplan de séparation: un neurone y a la valeur 0 si x est d'un côté de l'hyperplan, 1 s'il est de l'autre côté. Cela rend la fonction d'étape adaptée à des problèmes de classification linéairement séparables (Minsky & Papert, 1969). Cependant il est maintenant rarement utilisé car il n'est pas différentiable ce qui rend l'optimisation difficile. Une fonction sigmoïde différentiable est souvent préférable pour contourner ce problème (Figure 1.5) (Arnold, 2013; Kriesel, 2005).

Sigmoid activation (Fonction d'activation logistique): est définie comme suit

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

Les fonctions logistiques peuvent être vues comme des approximations continuellement différentiables de la Fonction Heaviside Step. Les réseaux de neurones avec une seule couche cachée sigmoïdale sont en fait des approximateurs universels (Cybenko, 1989; Hornik, 1989), c'est-à-dire qu'ils peuvent représenter n'importe quelle fonction avec un nombre suffisant de neurones (Figure 1.5).

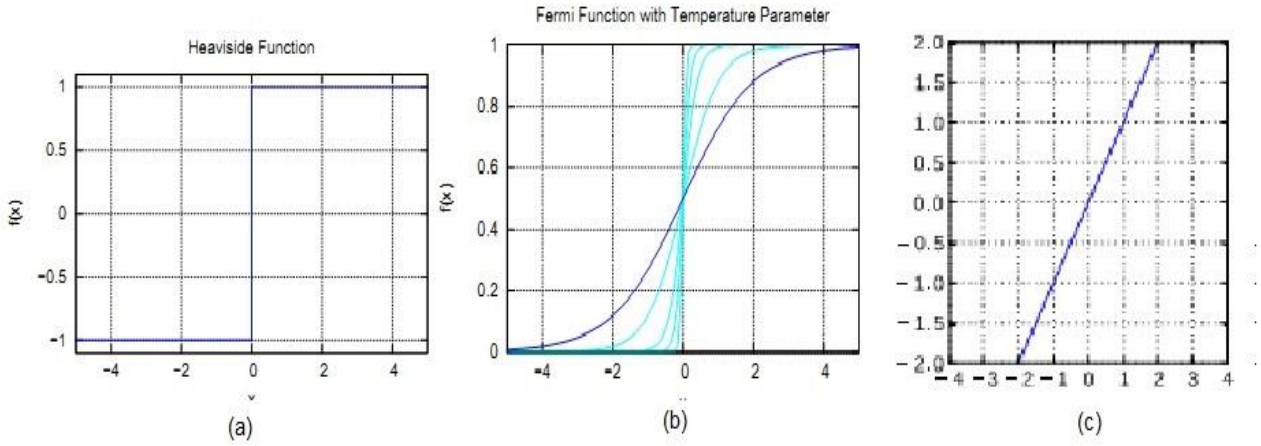


Figure 1.5: Trois fonctions d'activation communes: Fonction Heaviside Step (a), la fonction logistique (b), l'activation linéaire (c) (Kriesel, 2005)

1.4.4 Apprentissage avec l'algorithme rétropropagation (back-propagation algorithm) :

Dans un cadre supervisé, l'apprentissage des réseaux de neurones feed-forward se fait généralement avec une forme de descente en gradient. Ceci nécessite le calcul de dérivées partielles de la fonction d'erreur R/F les poids. Dans un réseau de feed-forward, ces dérivées partielles peuvent être calculées avec l'algorithme dit de rétropropagation (back-propagation algorithm)(Hinton, et al., 1986; LeCun, et al., 1998).

Dans cette section, nous considérons un ensemble de données $D = \{(x_1, t_1),\ (x_2, t_2), \dots, (x_N, t_N)\}$ de sorte que les sorties du réseau de neurones y ne soient pas confondues avec la variable cible.

La back-propagation peut être appliquée lorsque les fonctions d'erreur et les fonctions d'activation sont différentiables (Arnold, 2013). Il est alors possible d'utiliser la règle de chaîne pour obtenir:

$$\frac{\partial E(x)}{\partial w_{ij}} = \frac{\partial E(x)}{\partial a_j} \cdot \frac{\partial a_j}{\partial w_{ij}}$$

Où $a_j = b_j + \sum_i w_{ij}.z_i$ est la pré-activation d'un neurone de sortie en fonction des activations z_i des neurones dans la couche précédente. La fonction d'erreur E est une erreur par échantillon qui prend en compte le modèle. Par exemple, si nous utilisons l'erreur quadratique moyenne, $E(x) = \frac{1}{2}.(t-y)^2$ avec t la cible et $y = f(x)$ est la sortie du réseau de neurones (Arnold, 2013).

Il est noté r $\delta_j = \frac{\partial E(x)}{\partial a_j}$. Aditionellement, $\frac{\partial a_j}{\partial w_{ij}} = z_i$ et donc

$$\frac{\partial E(x)}{\partial w_{ij}} = \delta_j . z_i$$

Ainsi, le dérivé R/F w_{ij} est le produit de l'activation du neurone z_i avec δ_j . Pour la couche de sortie, les termes ${\delta_j}^{output}$ peuvent être calculés par une autre application de la règle de chaîne, à savoir.

$${\delta_j}^{output} = \frac{\partial E(x)}{\partial y_j} . \frac{\partial y_j}{\partial a_j} = \frac{\partial E(x)}{\partial y_j} . \dot{\varphi}(a_j)$$

Dans le cas du l'erreur quadratique moyenne, nous avon $\frac{\partial E(x)}{\partial y_j} = y_j - t_j$ Et ainsi ${\delta_j}^{output} = (y_j - t_j)\dot{\varphi}(a_j)$.

Pour les couches cachées, les erreurs δ_j peuvent être propagées vers l'arrière de manière récursive à travers le réseau, c'est-à-dire pour deux couches l_m et l_{m+1}:

$${\delta_j}^{m} = \frac{\partial E(x)}{\partial a_j} = \sum_k \frac{\partial E(x)}{\partial a_k} . \frac{\partial a_k}{\partial a_j}$$

Où nous somme sur toutes les unités k qui ont l'unité j comme entrée. Cela conduit à la règle de back-propagation

$${\delta_j}^{m} = \dot{\varphi}(a_j) \sum_k w_{kj} {\delta_k}^{m+1}$$

Cette règle peut ensuite être appliquée de manière itérative pour réduire les couches cachés. Les dérivées partielles résultantes peuvent ensuite être utilisées dans le cadre d'une procédure de gradient pour minimiser l'erreur sur un ensemble de données(Arnold, 2013).

Pour diminuer l'erreur, nous soustrayons ensuite cette valeur du poids actuel (éventuellement multiplié par un taux d'apprentissage η (Rojas, 1996) :

$${w_{ij}}^{new} = w_{ij} - \eta * \frac{\partial E(x)}{\partial w_{ij}}$$

L'algorithme de back-propagation a une complexité en $O(\dim \theta)$ où θ regroupe tous les paramètres du modèle et est donc très efficace.

Chapitre 2
Deep learning et réseaux de neurones

2.1 Introduction

En deep learning, nous pouvons distinguer deux types de réseaux de neurones, à savoir convolutionnels (CNN) et récurrents (RNN). La différence entre ces deux types réside dans le type de connexions entre les neurones. Dans le réseau convolutionnel la connexions entre les neurones est vers l'avant (feed-forward), par contre les connexions dans le réseau récurrent les connexions sont cycliques.

Dans ce chapitre, avant de décrire les réseaux CNN et RNN et leur constructions, nous présentons les architectures shallow et profondes.

2.2 Architectures shallow et profondes

Un réseau shallow est un réseau de neurones a une couche cachée, et un réseau profond en a plus d'un. Des couches cachées multiples permettent aux réseaux neurones profonds d'apprendre les descripteurs des données, car des descripteurs simples se recombinent d'une couche à l'autre pour former des descripteurs plus complexes (LeCun & Bengio, 2007). Les réseaux profond qui comportant de nombreuses couches transmettent des données d'entrée par les d'opérations mathématiques plus que les réseaux comportant peu de couches (les réseau shallow), peuvent conduire à des représentations beaucoup plus efficaces tout en étant des approximateurs universels(Hinton, et al., 2008). Par exemple, la fonction de parité dans les dimensions D nécessite que les paramètres $O(2^D)$ soient représentés par un SVM (Gaussian Support Vector Machine), $O(D^2)$ pour un réseau de neurones avec une couche cachée et $O(D)$ pour un réseau avec $O(\log_2 D)$ des couches (Bengio, et al., 2007). Parce qu'ils peuvent nécessiter moins de paramètres, les architectures profondes ont le potentiel à la fois d'améliorer la généralisation et de réduire les coûts de calcul.

2.3 Réseau de neurones convolutionnel

Dans cette section, nous décrivons d'une manière générale les architectures réseau de neurones convolutionnel (convolutional neural networks, CNN) ainsi que les différentes couches (LeCun, et al., 1998; Le Cun, et al., 1990).

2.3.1. Présentation générale

Les CNN sont un type spécial de réseaux de neurones qui utilisent d'autre manière de présentation uniquement d'autres couches (convolution, ReLU et pooling).

Les CNN sont fortement utilisés dans les tâches de classification, détection des objets et segmentation. Plus particulièrement, ils sont utilisés dans la reconnaissance des images et la classification des textes.

Historiquement, Les premiers CNN ayant eu du succès ont été inventés en 1989 par LeCun dans (LeCun, et al., 1989) et appliqués à la reconnaissance d'écriture manuscrite. Ces réseaux composés de couches de convolutions suivi par des couches de type MLP (multi layers Perceptron en anglais) sont appelés réseaux de type LeNet.

2.3.2 Différentes couches

Le CNN est une séquence de couches de convolution dénotés par $C^{(i)}$ et de pooling dénotés par $P^{(i)}$, $C^{(1)}P^{(1)}, C^{(2)}P^{(2)}, .., C^{(m)}P^{(m)}$, inspiré par l'organisation cellulaire du cortex visuel (Fukushima, 1980). Les connexions sont redirigées de sorte que toutes les entrées d'une couche de convolution se trouvent dans la couche de pooling précédente, et toutes les entrées d'une couche de pooling se trouvent dans la couche convolution précédente. La Figure 2.1 illustre l'architecture CNN.

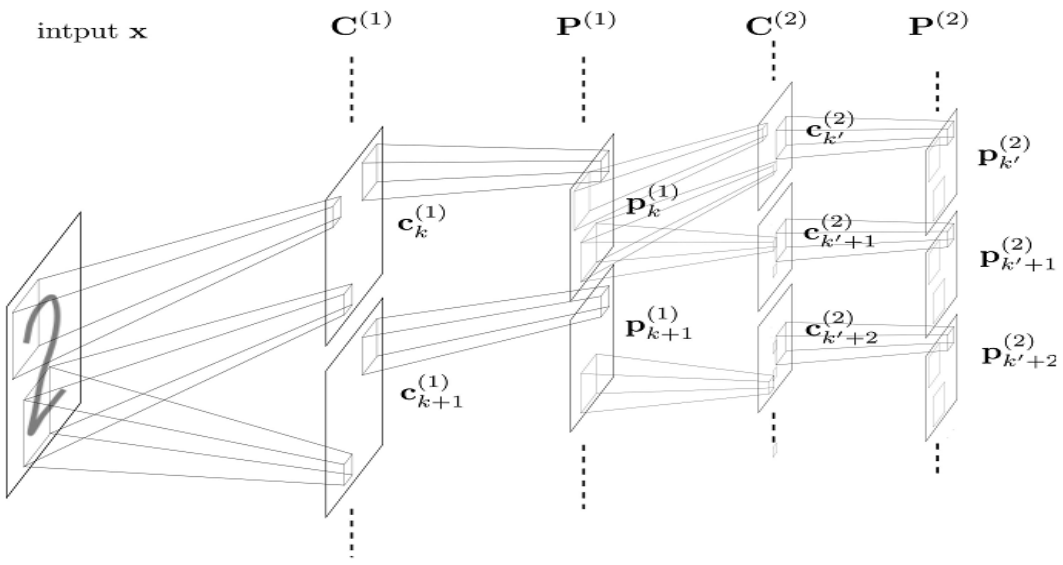


Figure 2.1 : Couches convolution et de pooling d'un réseau convolution extrait de (Arnold, 2013)

Nous allonsd écrire plus en détails les différentes couches ci-dessous.

2.3.2.1 Couche de convolution

Chaque couche de convolution $C^{(m)}$ est composée de plusieurs plans descripteurs $(C_1{}^{(m)}, C_2{}^{(m)}, \dots, C_K{}^{(m)}, \dots)$ qui calculent les convolution avec l'entrée x et un ensemble de descripteurs $(f_1{}^{(m)}, f_2{}^{(m)}, \dots, f_K{}^{(m)}, \dots)$.

$$C_K{}^{(m)} = x * f_K{}^{(m)}$$

Un exemple détaillé est traité dans la section 2.3.3

2.3.2.2 Couche pooling

Chaque couche de pooling $P^{(m)}$ est placé entre la couche de ReLU et convolution. Elle reçoit en entrée plusieurs plans de descripteurs et applique l'opération pooling qui correspond au calcul du maximum de cellules régulière. On utilise souvent des cellules carrées de petite taille pour ne pas perdre trop d'informations. Les choix les plus communs sont des cellules adjacentes de taille 2 × 2 qui ne se chevauchent pas. Le but de la couche de pooling est de réduire le nombre de paramètres et de calculs dans le réseau. On améliore ainsi l'efficacité du réseau et on évite le sur-apprentissage. Voir la figure 2.2 pour une représentation de la couche de pooling(LeCun, et al., 1989; Le Cun, et al., 1990).

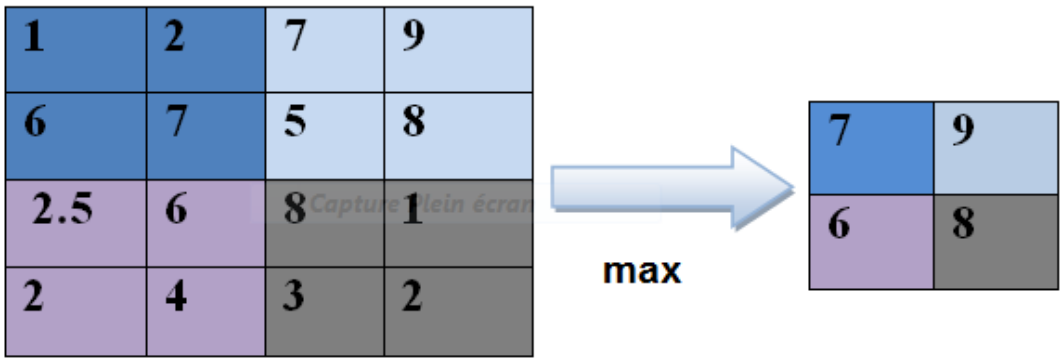


Figure 2.2 : Illustration illustre opération de pooling.

2.3.2.3 Couches de correction (ReLU)

Cette couche est utilisée éventuellement pour corriger des erreurs et ainsi améliorer l'efficacité du traitement entre les couches de convolutions et pooling. Cette couche va opérer comme une fonction d'activation sur les sorties.

La fonction ReLU (Rectified Linear Units) :

$$f(x) = \begin{cases} 0\ si & x < 0 \\ x\ si & x \geq 0 \end{cases}$$

Cette fonction force les neurones à retourner des valeurs positives (Krizhevsky, et al., 2012).

2.3.2.4 Couche entièrement connectée

Après plusieurs couches de convolution et de pooling, viennent la couche entièrement connectée ce qui transforme le réseau neurone convolutionnel en un réseau neurone (RN) normal (le réseau neurone se fait via des couches totalement connectées). Les neurones dans cette couche ont des connexions avec toutes les sorties de la couche précédente, par conséquent, nous pouvons calculés leur fonctions d'activations pour aboutir aux sorties souhaitées (Krizhevsky, et al., 2012; Le Cun, et al., 1990).

2.3.2.5 Couche d'apprentissage

La couche d'apprentissage détermine comment l'entraînement en réseau affecte la différence entre la sortie attendu et la sortie réelle. C'est généralement la dernière couche du réseau. Différentes fonctions d'apprentissage peuvent être utilisées pour différentes fonctions. La fonction "Softmax" est essentiellement utilisé (mais pas uniquement) pour des tâches de classification. Permet de construire des réseaux de neurones avec plusieurs sorties normalisées ce qui le rend particulièrement adapté à la création de classifications par les réseaux de neurones avec des sorties probabilistes.

Mathématiquement, la fonction softmax est représentée ci-dessous, où x est un vecteur des entrées de la couche de sortie. Telle que $x = (x_1, x_2, \dots, x_k)$:

$$\varphi(x)_j = \frac{\exp(x_j)}{\sum_{j=1}^{k} \exp(x_j)}$$

Pour tout $j \in \{1, \dots, k\}$.

2.3.3 Exemple

Pour nous aider à comprendre le fonctionnement de CNN, nous allons nous aider d'un exemple simplifié et chercher à déterminer si une image représente un X ou un O (Figure 2.3).

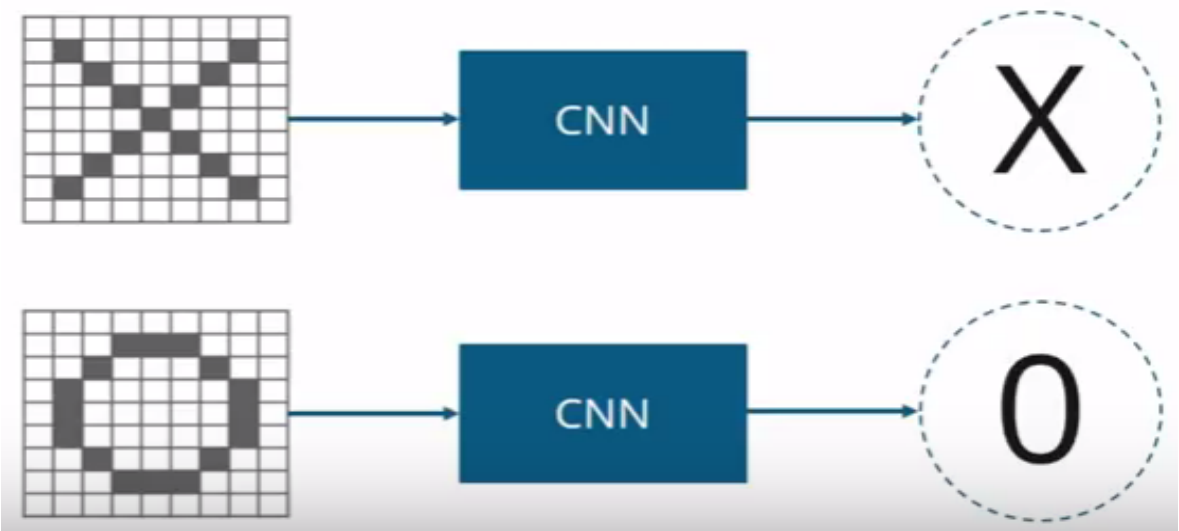


Figure 2.3 : Chercher si une image représente un X ou un O.

Cas plus difficile:

- ici, nous aurons quelques problèmes car l'image X et O n'aura pas toujours la même image il peut y avoir certaines déformations. Considérez la Figure 2.4 ci-dessous :

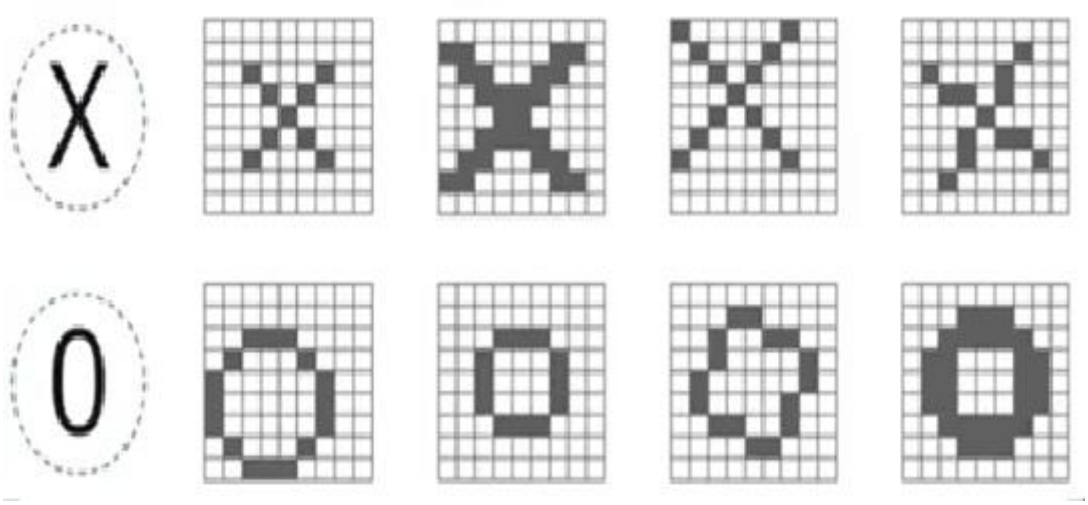


Figure 2.4 : Certaines déformations sont situées sur le X et O.

Notre CNN n'a qu'une seule tâche à réaliser: chaque fois qu'on lui présente une image, il doit décider si cette image représente un X ou un O (Figure2.5).

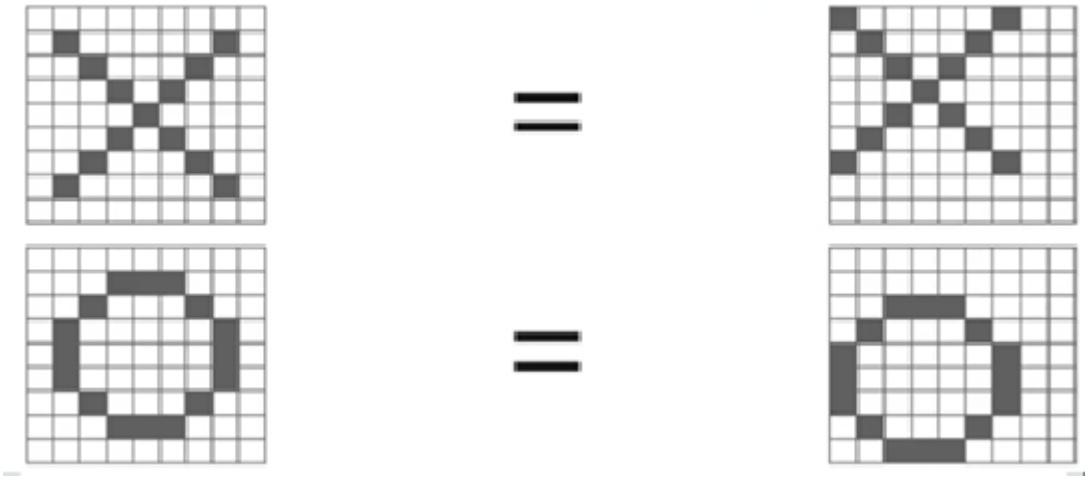


Figure 2.5 : Comparaison de chaque nouvelle image à ces deux images de X et O.

Un ordinateur comprend une image en utilisant un nombre à chaque pixel. Dans notre exemple, nous avons considéré qu'un pixel noir aura la valeur 1 et que le pixel blanc aura -1 valeur (Figure 2.6).

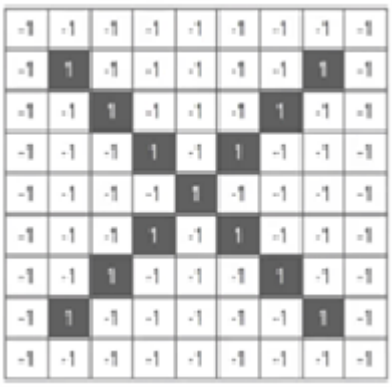


Figure 2.6 : Les pixels de l'image d'entré.

En utilisant des techniques normales, les ordinateurs comparent ces images comme la Figure 2.7:

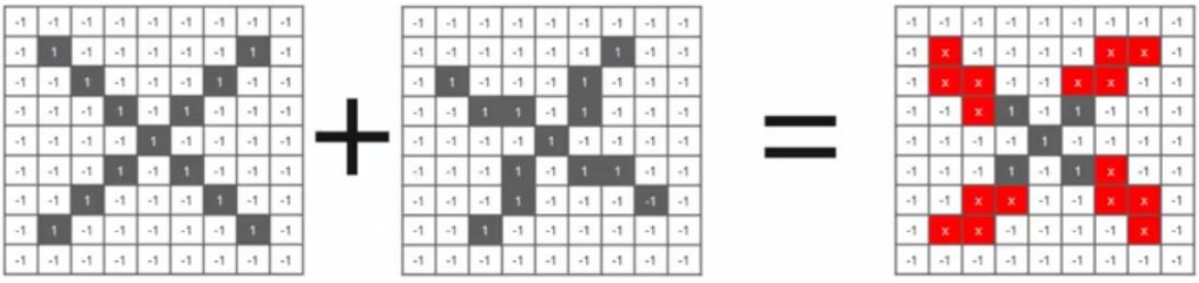


Figure 2.7 : Comparaison entre les deux X différent.

CNN compare les images fragment par fragment. La pièce qu'il recherche s'appelle des descripteurs.

En trouvant des descripteurs correspondances, à peu près la même position dans deux images, CNN obtient beaucoup mieux à voir la similitude que les schémas d'appariement d'images entières (Figure 2.8).

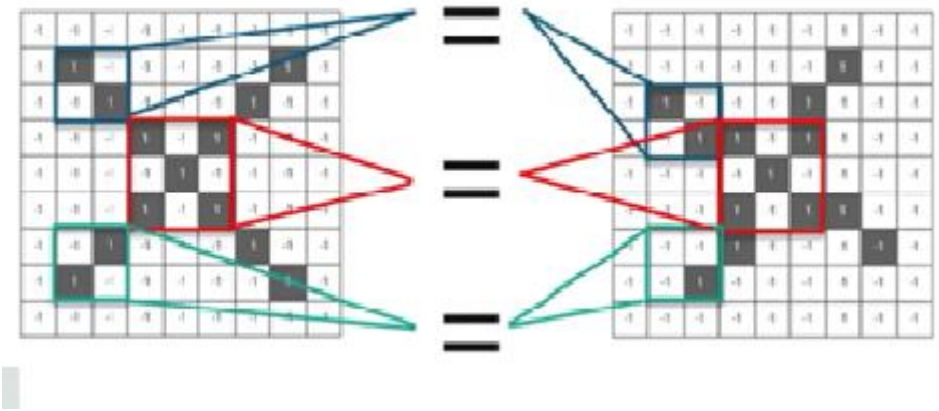


Figure 2.8 : Comparaison entre deux image pour recherche les descripteurs.

Nous allons prendre des descripteurs ou des filtres, comme indiqué dans Figure 2.9 ci-dessous:

Figure 2.9 : Les descripteurs différents.

Couche de convolution :

Ici, nous allons déplacer le descripteur (filtre) à toutes les positions possibles sur l'image (Figure 2.10)

- ✓ étape 1 : alignez le descripteur et l'image
- ✓ étape 2: multiplier chaque pixel d'image par le pixel descripteur correspondant.

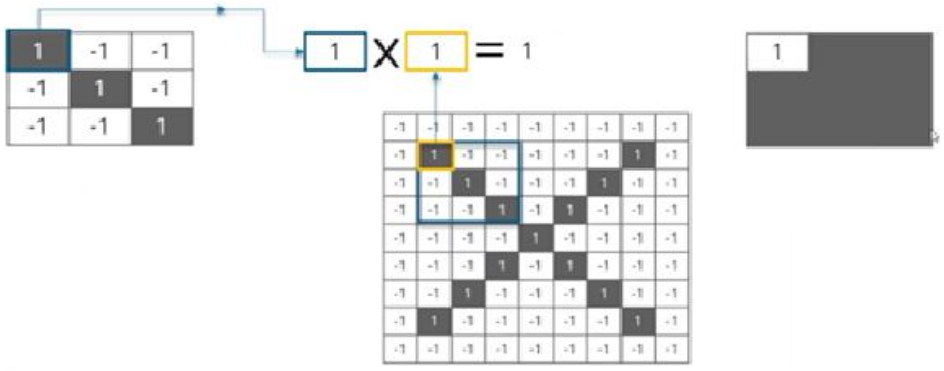


Figure 2.10 : La multiplication entre le descripteur et l'image.

- ✓ étape 3: additionnez-les.
- ✓ étape 4: diviser par le nombre total de pixels dans le descripteur (Figure 2.11).

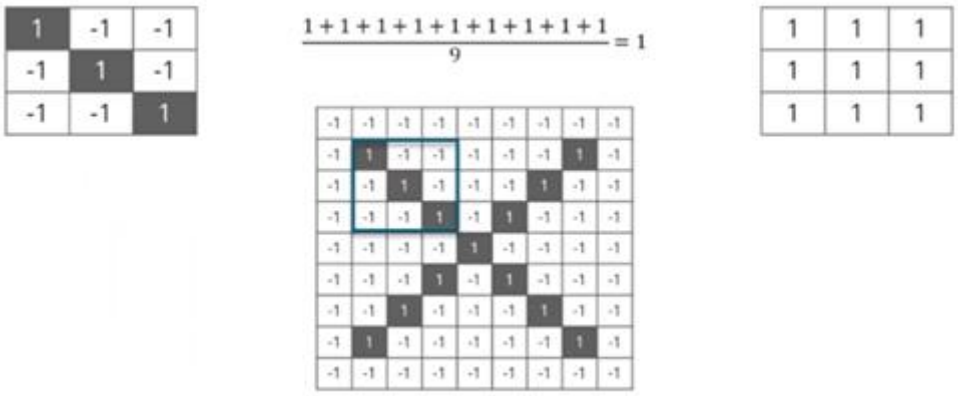


Figure 2.11 : Figure illustre l'opération addition et division.

Maintenant, pour garder une trace de l'endroit de ce descripteur était, nous créons un plan et mettons la valeur du filtre à cette place (Figure 2.12).

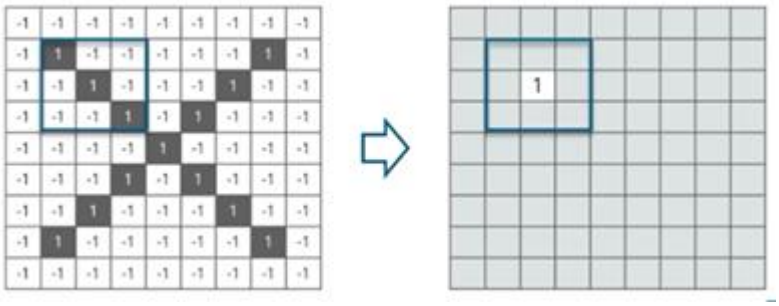


Figure 2.12 : Création un plan de descripteurs.

Maintenant, en utilisant le même descripteur et déplacez-le à un autre endroit (glisser le filtre dans toute l'image)

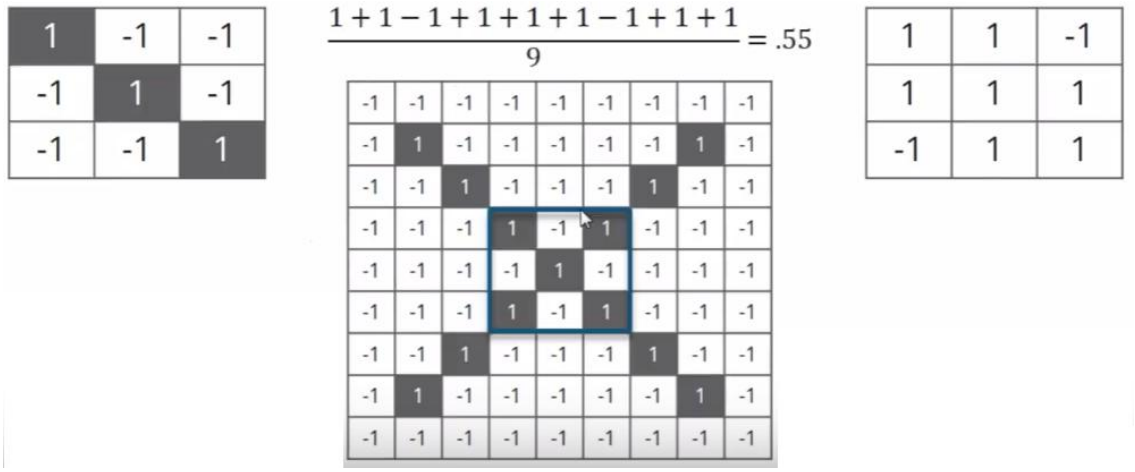


Figure 2.13 : Le déplacement du descripteur à toute l'image.

Sortie de la couche de convolution:

De même, nous allons déplacer le descripteur à chaque autre position de l'image et nous regardons comment le descripteur correspond à la zone. Enfin nous aurons une sortie comme la Figure 2.14:

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

Figure 2.14 : la sortie d'un plan descripteur.

Nous effectuons la même convolution avec tous les autres filtres, nous obtenons la couche de convolution (Figure 2.15)

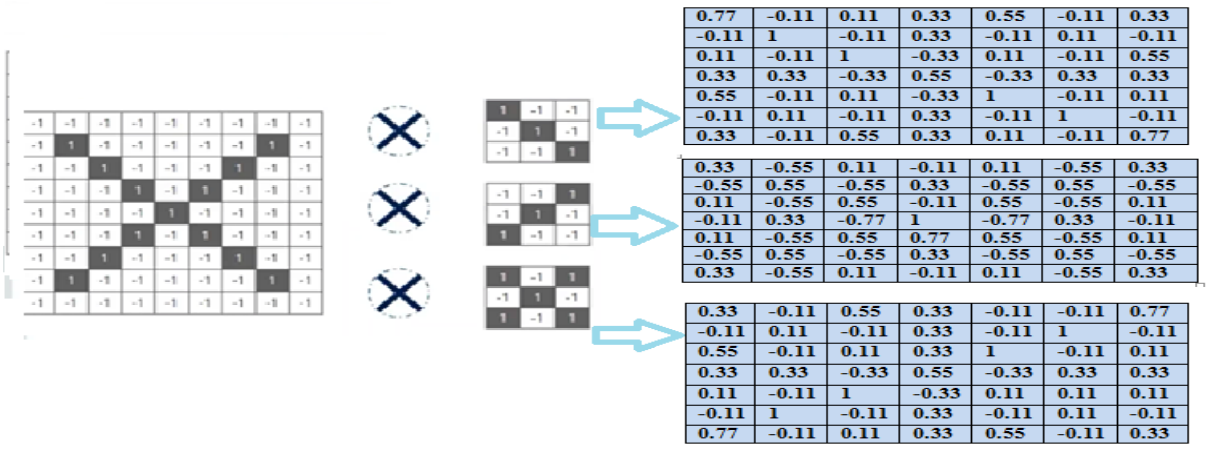


Figure 2.15 : La sortie de la couche convolution.

Couche de ReLU :

dans cette couche supprimer toutes les valeurs négatives de l'image filtrée et le remplacer par zéro et laisser les valeurs positives Comme représenté dans la Figure 2.16.

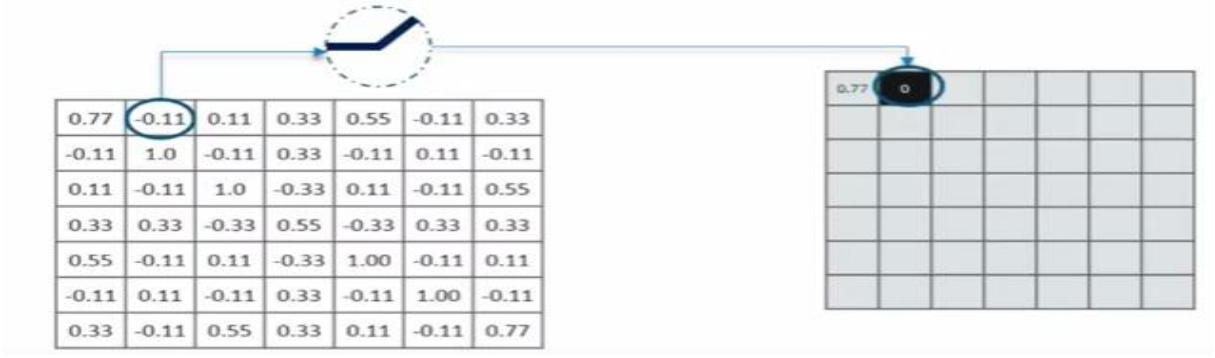


Figure 2.16 : Appliquer la fonction de ReLU.

Sortie pour un descripteur dans la Figure 2.17 :

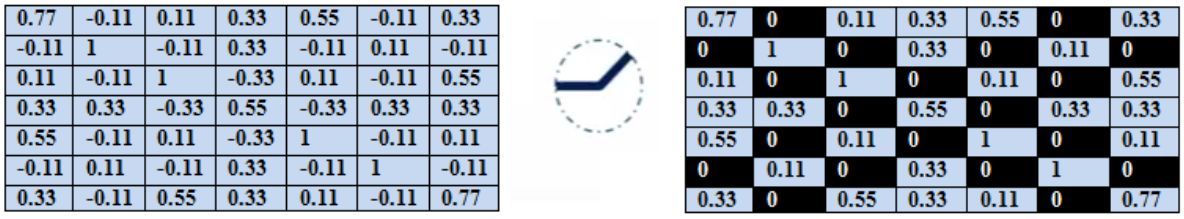


Figure 2.17 : Sortie de la couche ReLU pour un descripteur.

Sortie de la couche ReLU pour touts les descripteurs dans la Figure 2.18 :

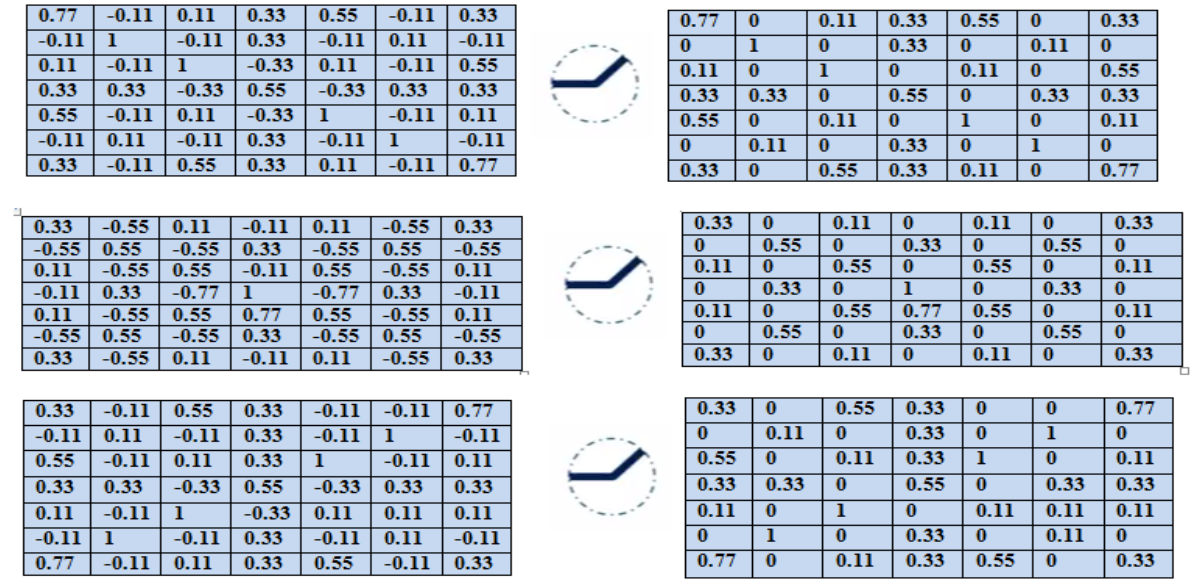


Figure 2.18 : Sortie de la couche ReLU.

Couche de pooling :

Nous calculons la valeur maximale dans chaque cellule. Nous commençons par la première image filtrée, dans première cellule, la valeur maximale est 1. Cette opération est représenté dans la Figure 2.19.

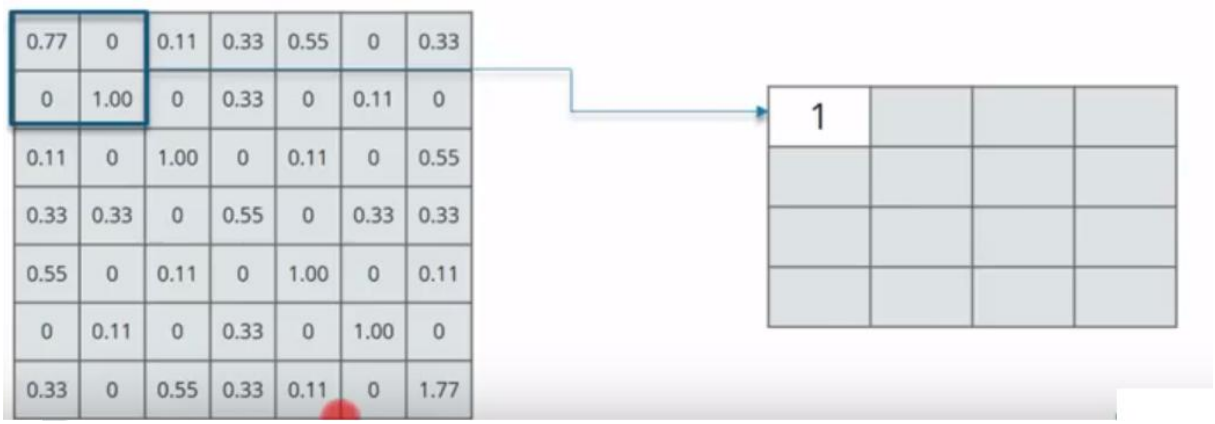


Figure 2.19 : Calculer la valeur maximale dans chaque cellule.

Déplacer la cellule sur toute l'image pour calculer le maximal de chaque cellule. La Figure 2.20 représente la sortie de la couche pooling pour un descripteur.

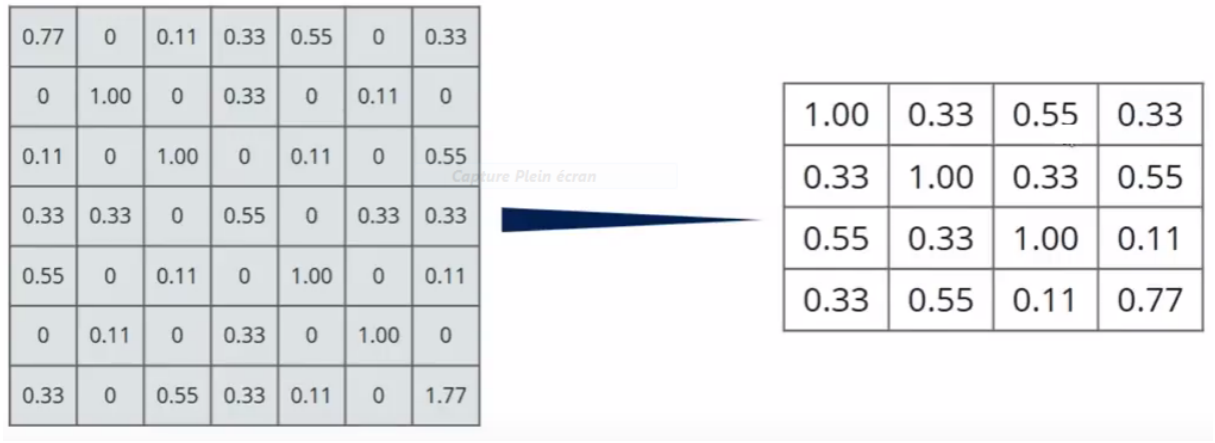


Figure 2.20 : La sortie de la couche pooling pour un descripteur.

La sortie de la couche de pooling après le passage (Figure 2.21)

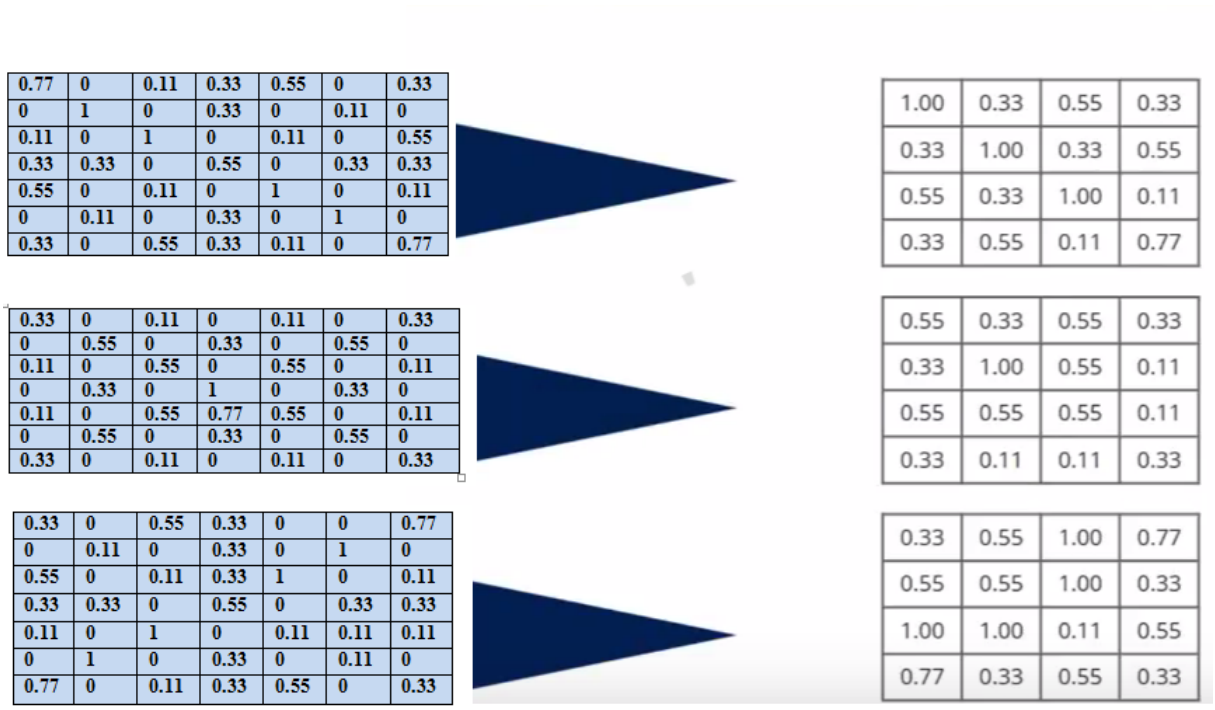


Figure 2.21 : La sortie de la couche de pooling.

La Figure 2.22 illustre l'ordre des couches jusqu'à maintenant. Figure 2.23 représente le résultat après plusieurs couches de convolution, ReLU et pooling.

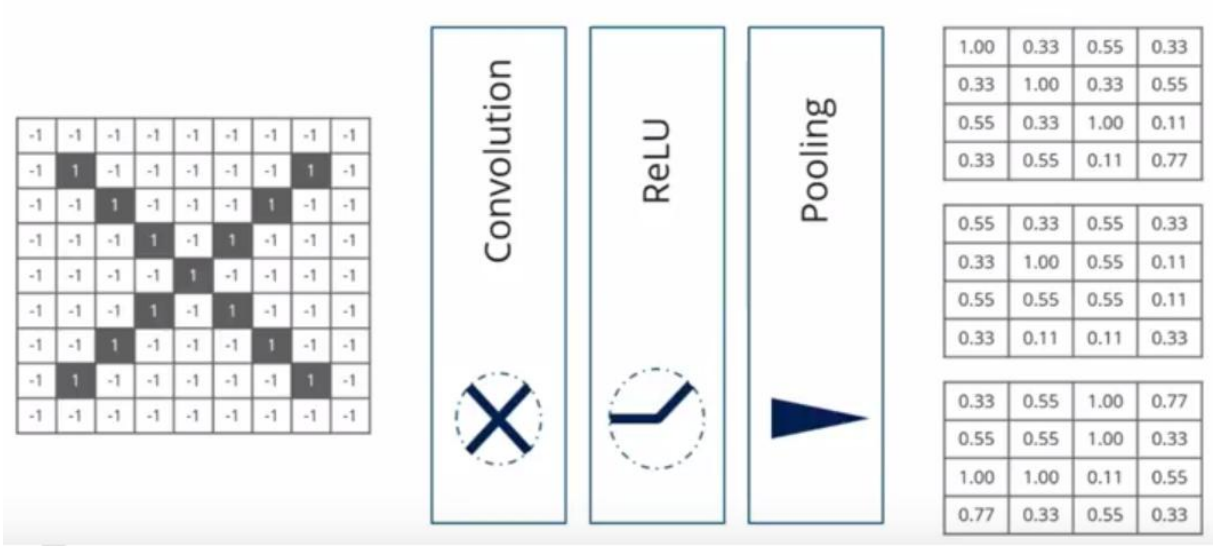


Figure 2.22 : L'ordre des couches.

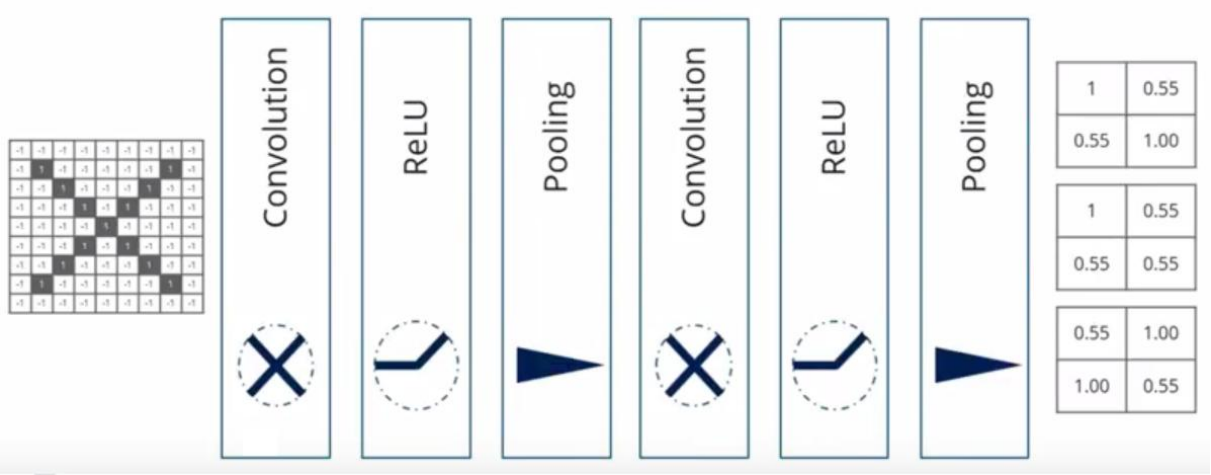


Figure 2.23 : Le résultat après plusieurs couches de convolution, ReLU et pooling.

Couche entièrement connectée

C'est la dernière couche où la classification réelle se produit. Ici nous prenons nos images filtrées et les mettons dans une seule liste comme la Figure 2.24.

Figure 2.24 : Couche entièrement connectée.

Sortie

Considérons la Figure 2.25 ci-dessous, comme vous pouvez le voir pour "X" il y a différents éléments qui sont élevés et de même, pour "O" nous avons des éléments différents qui sont élevés.

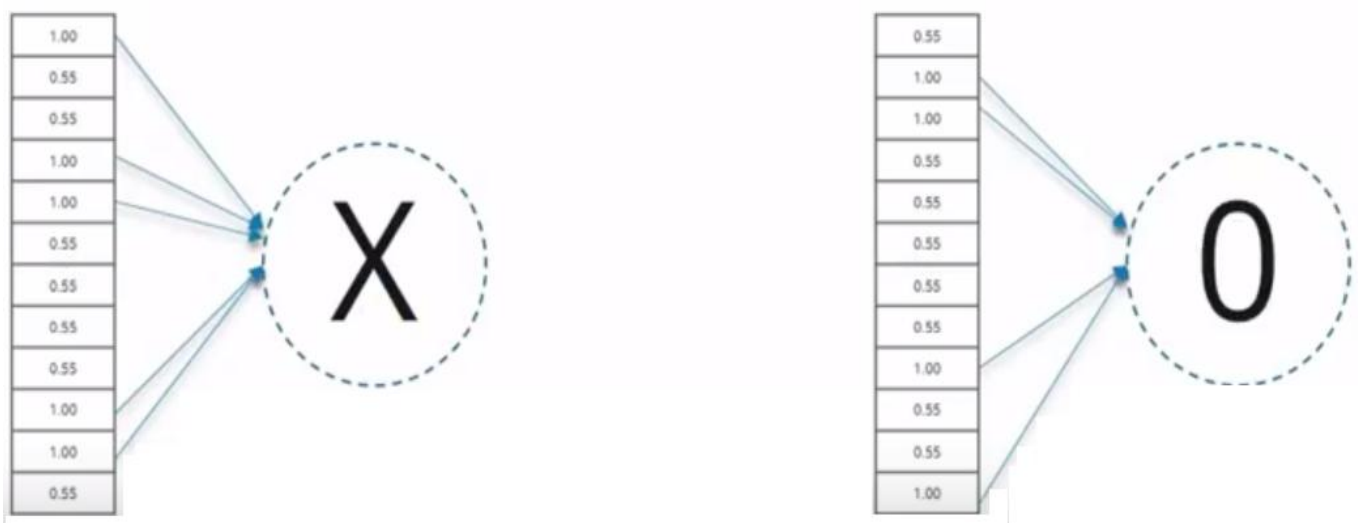


Figure 2.25 : Les vecteurs de ‘’X’’ et ‘’O’’.

Prédiction

considérer la Figure 2.26 ci-dessous d'un vecteur de nouvelle image d'entrée :

0.9
0.65
0.45
0.87
0.96
0.73
0.23
0.63
0.44
0.89
0.94
0.53

Figure 2.26 : Le vecteur d’une nouvelle image d’entrée.

Nous comparons ce vecteur avec le vecteur de "X" et "O", cette comparaison illustrée dans les Figure 2.27 et 2.28. Commencer par le vecteur "X" :

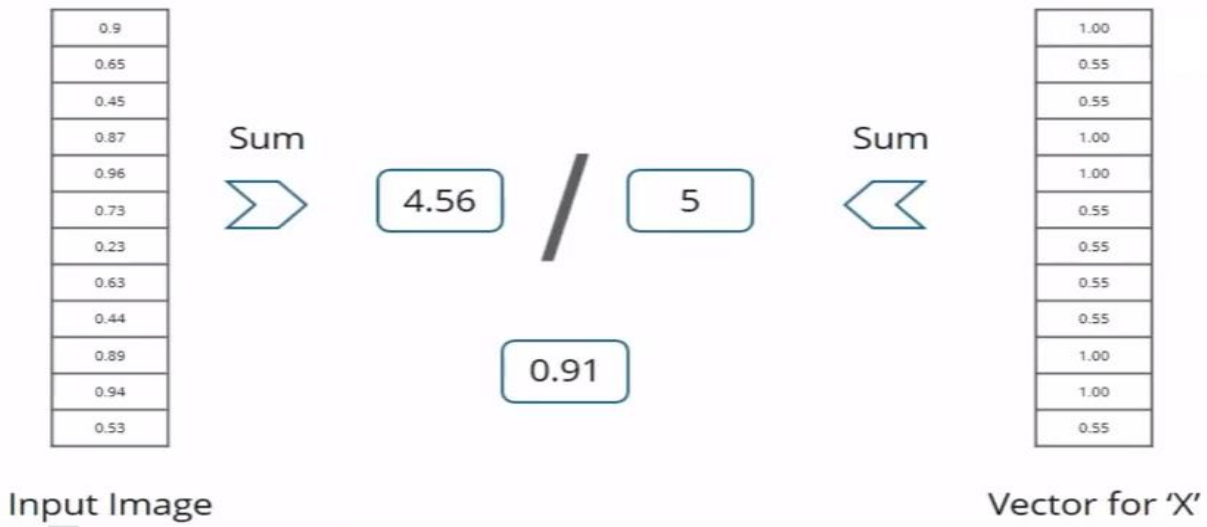


Figure 2.27 : Comparaison entre le vecteur d'entrée avec "X".

Comparer le vecteur d'entrée avec "O" :

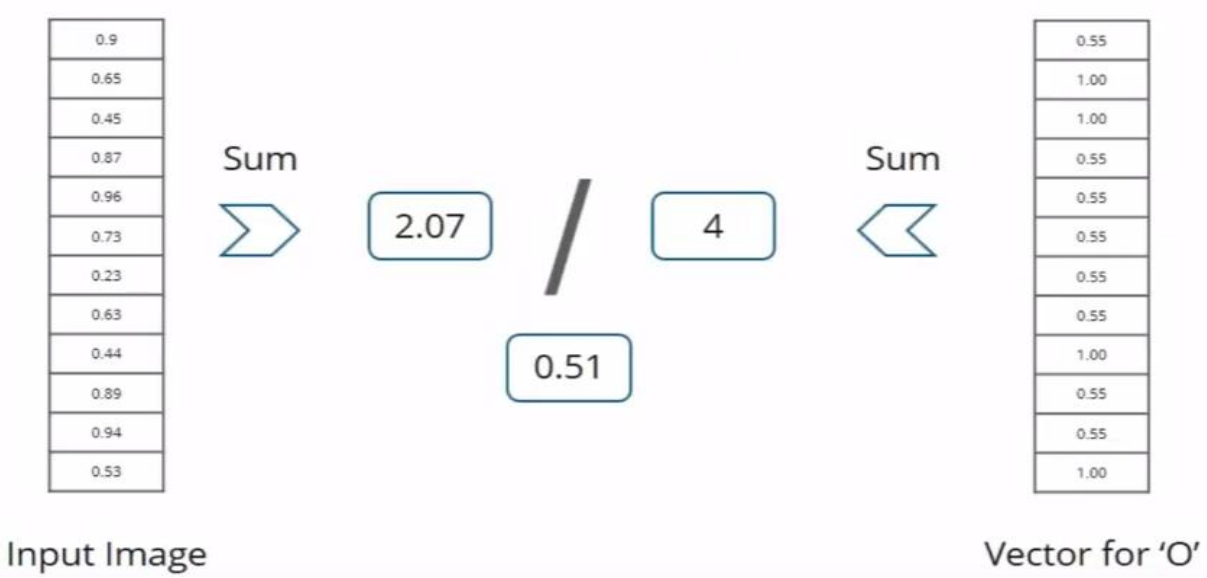


Figure 2.28 : Comparaison entre le vecteur d'entrée avec "O".

Résultats :

le vote dépend du degré de prédiction d'une valeur "X" ou "O" . Dans cet exemple, la valeur de prédiction de "X" est la plus élevée (Figure 2.29).

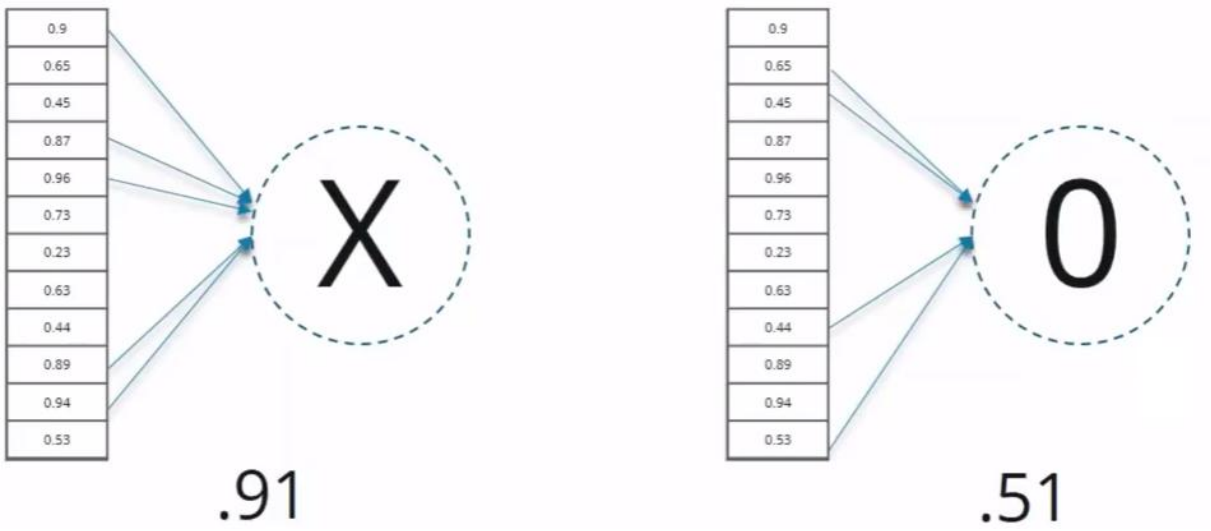


Figure 2.29 : Résultat finale est X.

2.4 Réseaux de neurones récurrents

Les humains ne commencent pas leur réflexion à partir de zéro à chaque instant. Par exemple, en lisant cette section, nous comprenons chaque mot en fonction de nos compréhensions précédentes, les réseaux de neurones traditionnels (feed-forward) ne peuvent pas le faire. Car ils ne pourraient pas utiliser leur raisonnement sur les informations précédant. Les réseaux de neurones récurrents (Recurrent Neural Networks RNN) résolvent ce problème. Ce sont des réseaux avec des boucles qui permettent à l'information de se répéter.

2.4.1 Présentation générale

Les RNN ont été proposés dans les années 80 (Rumelhart, et al., 1986; Elman, 1990; Werbos, 1988) pour modéliser les séries temporelles. Cette structure est similaire aux réseaux de neurones traditionnels, mais ces réseaux comportent des cycles dans leur graphe (Pascanu, et al., 2013). Ces cycles permettant au réseau d'entretenir une information en l'envoyant à lui-même telles que présente dans la Figure 2.30. Ces réseaux sont largement utilisés pour le traitement automatique des textes, notamment, dans la traduction automatique.

Ceci dit, un RNN prend en entrée une séquence d'événements $x = (x_1, x_2, \dots, x_T)$ et définit la séquence de vecteurs de sortie $y = (y_1, y_2, \dots, y_T)$ en fonction de $t \in [1, T]$, en passant par la définition de la séquence d'états cachés $h = (h_1, h_2, \dots, h_T)$:

$$h_t = H(w_{xh}.x_t + w_{hh}.h_{t-1} + b_h) \qquad (1)$$

$$y_t = f_y(w_{hy}.h_t + b_y) \qquad (2)$$

Où T est nombre total des vecteurs d'entrées, W_{xh}, W_{hh}, W_{hy} sont les matrices de poids entre les couches, b est le vecteur de biais et la fonction H utilisée dans le cas RNN est généralement la tangente hyperbolique ($\tanh(x)$).

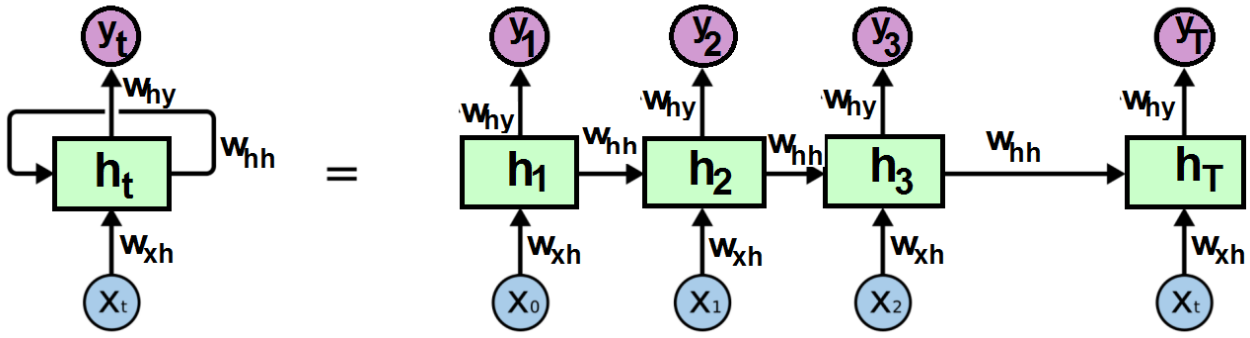


Figure 2.30 : Schéma d'un réseau neurones récurrent.

2.4.2 Apprentissage de réseau de neurones récurrent

La représentation hiérarchie du RNN représentée à la Figure 2.30 nous permet d'utiliser l'algorithme de rétropropagation du gradient généralisé aux réseaux de neurones traditionnels. Cette version est appelée rétropropagation du gradient à travers le temps (Backpropagation Through Time ou BPTT) (Rumelhart, et al., 1985).

L'algorithme BPTT dépend des étapes suivantes (Pascanu, et al., 2013) :

Pour chaque instant $t \in [1..T]$, le terme $E(t)$ est l'erreur à l'instant t est déterminé comme suit :

$$E_{(t)} = \frac{1}{2}.(C_{(t)} - y_{(t)})^2$$

Où $C_{(t)}$ est la cible, $y_{(t)}$ est la sortie produites de la couche à instant t.

Gradient vers l'arrière :

$$\frac{\partial E}{\partial w_{hh}} = \sum_t \frac{\partial E_{(t)}}{\partial w_{hh}}.$$

Exemple de gradient :

A titre d'illustration, nous prenons l'exemple de calcul du gradient à la deuxième sortie, en se basent sur les équations (1) et (2) :

$$\frac{\partial E_2}{\partial w_{hh}} = \frac{\partial E_2}{\partial y_2}.\frac{\partial y_2}{\partial h_2}.\frac{\partial h_2}{\partial H}.\frac{\partial H}{\partial a}.\frac{\partial a}{\partial w_{hh}}$$

Avec $a = (w_{xh}.x_2 + w_{hh}.h_1 + b_h)$.

Par la suite, nous passons à l'adaptation les paramètres :

$$\Delta w_{hh} = \eta.\frac{\partial E}{\partial w_{hh}}$$

Avec η est le taux d'apprentissage.

Enfin nous obtenons le nouveau paramètre à utiliser dans les itérations qui suivent.

$$w_{hh}{}^{new} = w_{hh} + \Delta w_{hh}$$

2.4.3 Problèmes de RNN

Les RNN présentent plusieurs avantages, peut être le plus important est qu'ils ne sont pas limités à des états internes discrets mais permettent des représentations de séquence distribuées continues. Cependant, ces réseaux prenent trop de temps pour être faisables ou ne pas bien fonctionner du tout.

Par ailleurs, ils ont des difficultés à traiter des séquences relativement longues, notamment ceux contenant plus de 10 événements(Hochreiter, et al., 2001). En effet, avec les calculs cumulatifs à long terme, l'erreur obtenue avec la rétropropagation du gradient diminue, ou augmente d'une manière exponentielle par rapport à l'échelle du temps. Ces deux problèmes sont nommés respectivement la « dissipation du gradient » (vanishing gradient) et « l'explosion du gradient » (exploding gradient)(Hochreiter, et al., 2001). La dissipation ou l'explosion du gradient s'aggrave dans ce cas en fonction du nombre de couches (Bouaziz, 2017).

Comme remède à ces problèmes, plusieurs alternatives ont été proposées. Dans la section suivante, nous présentons une alternative intéressante.

2.4.4 Long Short-Term Memory (LSTM)

Une des solutions les plus efficaces pour surmonter les problèmes ci-dessus est l'architecture Long Short-Term Memory (LSTM) ont été proposés par Hochreiter et Schmidhuber (1997). Les LSTM sont un type spécial de RNN, capable d'apprendre les dépendances à long terme.

2.4.4.1 Présentation générale

Les LSTM sont une catégorie de RNN dont l'architecture et la formulation mathématique générales sont identiques. Mais la différence fondamentale entre le RNN et LSTM réside dans le faite que dans les RNN simples, le traitement de la récurrence, symbolisé par la fonction f, est assuré par une simple fonction $tanh$ (Figure 2.31).Cependant, les LSTM ont une structure différente plus complexe. Où le traitement de la récurrence est assuré par une fonction de quatre étages (Figure 2.32). Le fonctionnement de cette fonction est décrit dans la section 2.4.4.2.

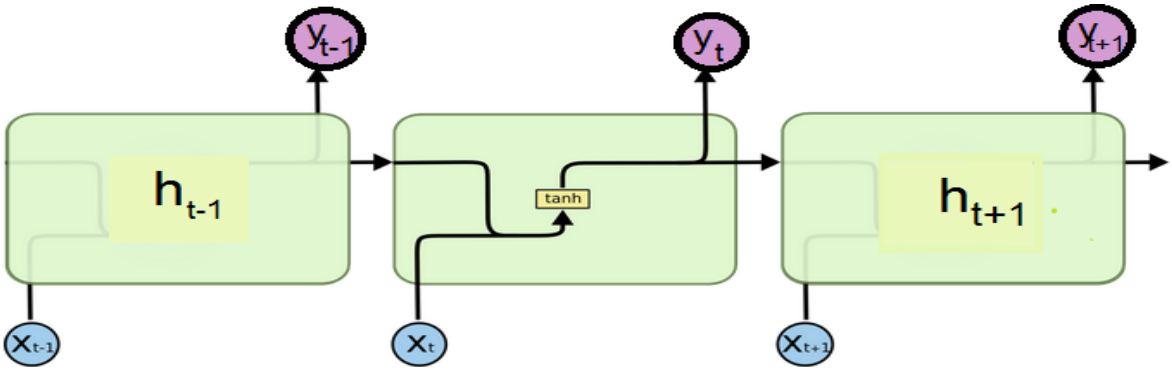


Figure 2.31 : Illustre architecture standard de RNN.

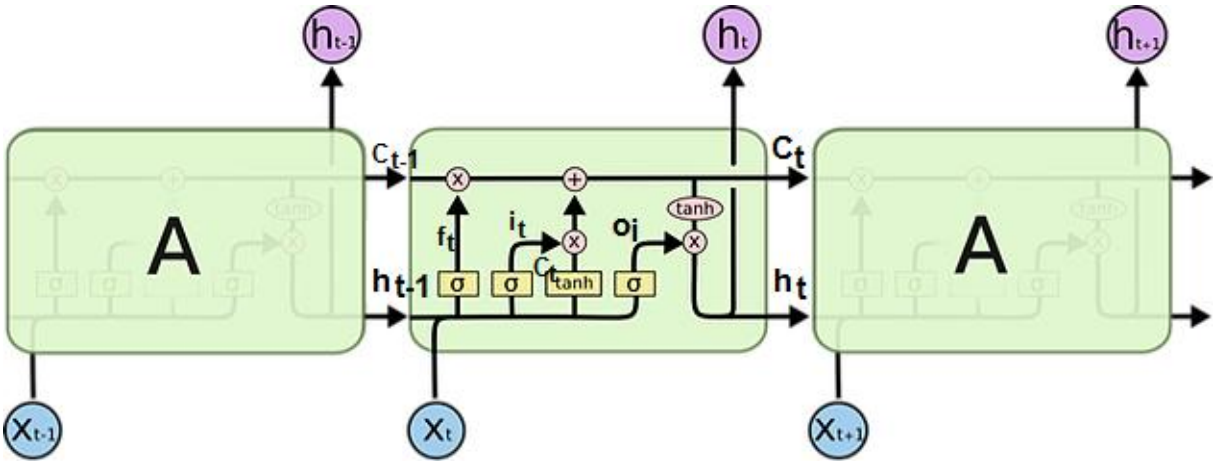


Figure 2.32 : Illustre architecture de LSTM.

2.4.4.2 Architecture du LSTM

L'architecture interne de LSTM est qu'il dispose d'une « cellule à mémoire » spéciale (Figure 2.32), qui remplace la couche caché $\boldsymbol{h(t)}$ de la Figure 2.30. La cellule de LSTM a un nœud centrale qui contient l'état (ou mémoire) de la cellule, et un certain nombre de « portes » divisées en 3 catégories. Ces portes permettent de gérer, d'une part, la tenue en mémoire de l'information séquentielle (portes d'entrée et d'oubli) et d'autre part, le rôle de l'état interne dans la production de chaque sortie (porte de sortie).

Les calculs au niveau de la cellule à mémoire sont effectués comme suit (Hocherieter & Schmidhuber, 1997):

$$f_t = \sigma\big(w_f[h_{t-1} + x_t] + b_f\big)$$

$$i_t = \sigma(w_i[h_{t-1} + x_t] + b_i)$$

$$\tilde{C}_t = \tanh(w_c[h_{t-1} + \ x_t] + b_i)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$o_t = \sigma(w_0[h_{t-1} + x_t] + b_0)$$

$$h_t = o_t * \tanh(C_t)$$

Où i, f et o, sont respectivement les portes d'entrée, d'oubli et de sortie, et C est le vecteur d'état interne ayant la même taille que celle du vecteur caché h. Les matrices w sont les poids entre la cellule C et les portes i, f et o. Enfin, σ est la fonction logistique sigmoïde.

2.4.5 Gated Recurrent Units (GRU)

Gated recurrent units GRU ont été proposés Cho, et al. (2014). GRU sont une variation récente des réseaux LSTM (Chung, et al., 2014).

2.4.5.1 Présentation générale

GRU pour faire en sorte que chaque unité récurrente capte de manière adaptative les dépendances de différentes échelles de temps. De même que pour l'unité LSTM, le GRU dispose d'unités de contrôle qui modulent le flux d'informations à l'intérieur de l'unité, cependant, sans avoir de cellules de mémoire séparées. Il a été démontré que les GRU sont plus performants que les LSTM classiques tout en étant plus rapides grâce à une architecture plus simple.

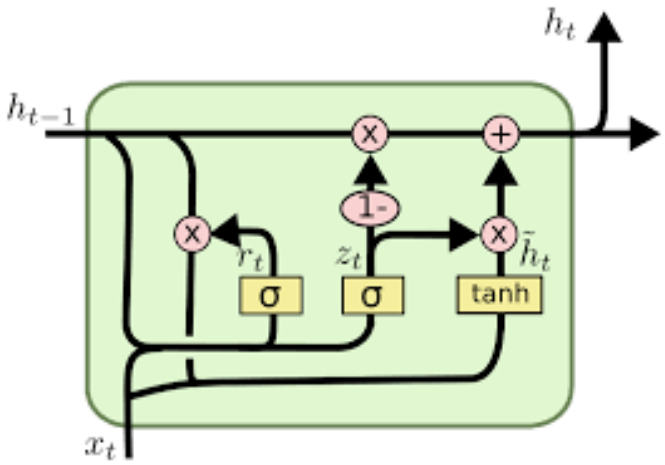


Figure 2.33 : Illustre architecture de GRU.

2.4.5.2 Architecture du GRU

Ils combinent les portes d'entrée et de sortie dans une porte d'update gate et fusionnent l'état caché et l'état de la cellule en un seul état, comme illustré à la Figure 2.33, ils sont définis comme suit:

$$z_t = \sigma(W_z.[h_{t-1} + x_t])$$

$$r_t = \sigma(W_r.[h_{t-1} + x_t])$$

$$\widetilde{h_t} = \tanh(W.[r_t * h_{t-1} + x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \widetilde{h_t}$$

Où r_t est une porte de réinitialisation (reset gate) et z_t est une porte d'update gate. Il a été démontré que les GRU sont plus performants que les LSTM classiques tout en étant plus rapides grâce à une architecture plus simple (Chung, et al., 2014).

Chapitre 3

Etat de l’art : classification de texte avec deep learning

3.1 Introduction

Dans ce chapitre, nous présentons une revue de la littérature pour la classification du texte utilisant le deep learning.

Nous présentons essentiellement les travaux suivant :

- Réseaux de neurones convolutionnels pour la classification des phrases(Kim, 2014).
- Réseaux de neurones convolutionnels récurrents pour la classification des textes(Siwei, et al., 2015).

Avant de présenter les travaux connexes à la classification de texte utilisant le deep learning, nous jugeons nécessaire de clarifier le concept d'analyse distributionnelle.

3.2 Analyse distributionnelle

Les relations entre les termes jouent un rôle important dans la terminologie pour définir le sens des termes. De nombreuses méthodes ont été proposées permettant l'acquisition des relations sémantiques entre termes, parmi celles-ci on trouve l'analyse distributionnelle.

3.2.1 Présentation générale

L'analyse distributionnelle remonte à Harris (1954), elle conduit au regroupement des termes sémantiquement proches dans des contextes partagés. Plus les contextes sont communs, plus les termes cibles sont sémantiquement proches. À partir de ces regroupements, il est possible d'obtenir un nombre important de relations classiques telles que la synonymie ou l'hyperonymie ou des relations propres au domaine.

3.2.2 Mise en œuvre du modèle

La mise en œuvre de l'analyse distributionnelle dans un processus automatique s'appuie généralement sur une représentation vectorielle des mots de corpus. Un mot cible est alors un point dans un espace à n-dimensions où chaque dimension correspond à des contextes possibles, et où la valeur associée aux dimensions est le nombre d'occurrences du contexte correspondant. Le vecteur de chaque mot cible représente donc les informations contextuelles mais aussi des données statistiques distributionnelles comme le nombre de contextes et le nombre d'occurrences des contextes partagés (Fabre & Lenci, 2015; Turney & Pantel, 2010).

À titre d'exemple la similarité sémantique entre deux mots cibles est calculée à l'aide du cosinus de leur angle dans leur espace. Les modèles vectoriels ont l'avantage de permettre une quantification facile de la proximité sémantique entre deux mots (Turney & Pantel, 2010).

3.2.3 Modèles distributionnels

Les modèles distributionnelles se différencier suivant le contexte de travail choisi, à savoir si voulu travail sur peu de document ou sur des données volumineux.

Nous citons dans ce qui suit deux exemples de méthodes distributionnels fréquemment utilisé dans les recherches liés au l'analyse distributionnelle.

3.2.3.1 Analyse sémantique latent

L'analyse sémantique latent (Latent Semantic Analysis, LSA) est un modèle statistique. LSA est à la base d'un nombre de plus en plus important de recherches en psycholinguistique(Landauer, et al., 1998), comme outil de recherche documentaire(Deerwester, et al., 1990). Mais très vite, grâce à ses performances, son utilisation s'est étendu à d'autres domaines (Landauer & Dumais, 1997) tels que l'analyse de la cohérence dans des textes (Bestgen, et al., 2003).

L'objectif est de représenter le sens des mots à partir de l'analyse de grands corpus de textes, en prenant en compte le contexte dans lequel chaque mot apparaît. La technique consiste à déterminer le nombre d'occurrences de chaque mot dans chaque contexte qui utilise une matrice. L'unité de contexte retenue est le paragraphe, qui possède l'avantage d'être de taille correcte tout en étant facilement repérable par une machine. Une forme d'analyse factorielle est ensuite appliquée à cette matrice afin de représenter chaque mot et chaque paragraphe par un vecteur de très grande dimension. Deux mots apparaissant dans des contextes similaires sont représentés par des vecteurs proches (la mesure de proximité est définie par le cosinus de leur angle) (Benoît, et al., 2001)

3.2.3.2 Word2Vec

Le modèle word2vec est propose par Mikolov, et al., (2013). Ce modèle permet de construire une représentation vectorielle des mots d'un texte ou d'une requête permettant de capturer la sémantique exprimée par chaque mot, en se basant sur le contexte auquel apparie le mot.

Deux architectures ont été proposées pour ce modèle, Skip-Gram et sac-de-mot continus (CBOW). Ces architectures utilisent les réseaux de neurones constitués de trois couches : entrée, caché et sortie tels qu'illustré dans la Figure 3.1.

Il est clair que le modèle Skip-Gram tente de prédire pour un mot donné le contexte dont il a issu. Alors que pour le modèle CBOW, l'objectif consiste à prédire un mot à partir d'un contexte donné.

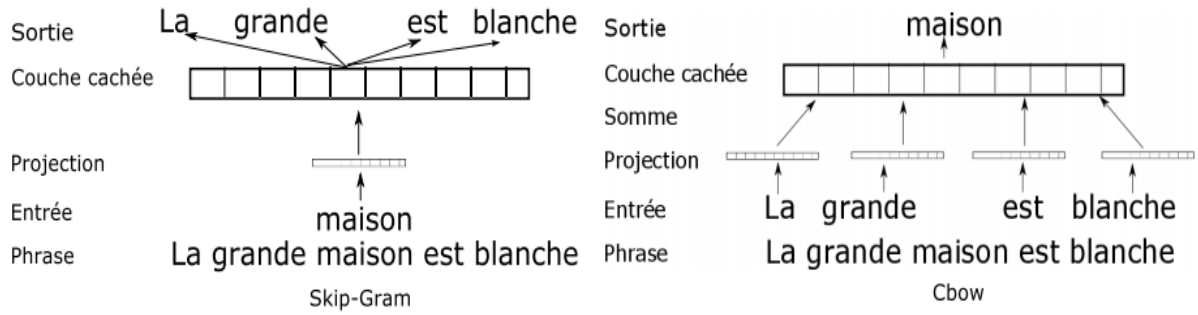


Figure 3.1 : Exemple des architecteur Skip-gram et CBOW de Word2vec.

3.3 Réseaux de neurones convolutionnels pour la classification des phrases

Les modèles de deep learning ont obtenu des résultats remarquables dans la reconnaissance des images (Krizhevsky, et al., 2012). Dans le traitement automatique de la langue, la plupart des méthodes utilisant le deep learning impliquant l'apprentissage des représentations vectorielles des mots par le biais les modèle de langage à base de neurones.

Dans cette section, nous présentons le travail de Kim, (2014) par la classification des phrases en utilisant les CNN. En effet, le présent travail est similaire à celui de Razavian, et al., (2014) appliqué par la classification d'images.

3.3.1 Modèle

L'architecture du modèle, représentante dans la Figure 3.2, est une forme simple de l'architecture CNN (Collobert, et al., 2011). Soit $x_i \in \mathbb{R}^k$ le vecteur de mot k −dimension correspondant au i^e mot de la phrase. Une phrase de longueur n est représentée :

$$x_{1:n} = x_1 \oplus x_2 \oplus \dots \oplus x_n$$

Où $\oplus$ est l'opération de concaténation. En général, soit $x_{i:\, i+j}$ se réfère à la concaténation des mots $x_i, x_{i+1}, \dots, x_{i+j}$.

3.3.1.1 Couche convolution

Une opération de convolution implique un filtre $w \in \mathbb{R}^{hk}$, qui est appliqué à une fenêtre de h mots pour produire une nouveau descripteur. Par exemple, un descripteur C_i est généré à partir d'une fenêtre de mots $x_{i:\, i+h-1}$ par

$$C_i = f(w \,.\, x_{i:\, i+h-1} + b)$$

Ici b est un terme de biais et f est une fonction non linéaire telle que la tangente hyperbolique. Ce filtre est appliqué à chaque fenêtre des mots dans la phrase pour produire une carte de descripteurs

$$C = [C_1,\ C_2, \ldots, C_{n-h+1}],$$

avec $C \in \mathbb{R}^{n-h+1}$.

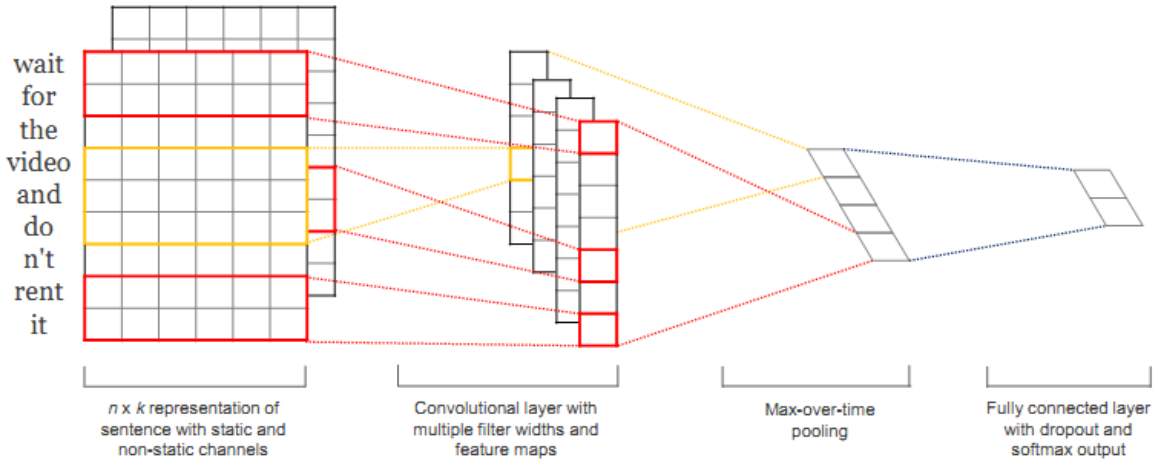


Figure 3.2 : Architecture de modèle avec deux canaux pour un exemple de phrase.

3.3.1.2 Couche pooling

À l'issue de la couche convolution, nous obtenons la carte descripteurs $C \in \mathbb{R}^{n-h+1}$. Dans cette couche, l'opération de pooling est appliquée sur la carte de descripteurs en prenant la valeur max :

$$\hat{C} = \max\{C\}$$

Cette opération de pooling traite naturellement des longueurs de phrases variables (Collobert, et al., 2011) . L'idée est de choisir le descripteur le plus important pour chaque carte descripteur.

3.3.1.3 Couche entièrement connectée

Le modèle utilise des filtres multiples (avec changent des tailles de fenêtre) pour obtenir des filtres forme l'avant dernière couche qui est passée à une couche entièrement connectée (couche de Softmax) dont la sortie est une distribution de probabilité sur les étiquettes.

3.3.1.4 Régularisation

Pour la régularisation, ils utilisent un dropout sur l'avant dernière couche avec une contrainte sur les normes l_2 des vecteurs de poids(Srivastava, et al., 2012). Le dropoutant aléatoirement sur des unités cachées (c'est-à-dire en mettre à 0 certains poids de descripteurs z telle que $z = (C_1, C_2, \dots, C_m)$). Créer le vecteur de masquage r des variables aléatoires de Bernoulli avec la probabilité p (un type de hyper-paramatre) d'être 1. Pour l'unité de sortie y en propagation par élément.

$$y = w(z \circ r) + b$$

Ou $\circ$ est l'opérateur de multiplication par élément et b est un terme de biais. Les gradients ne sont pas rétropropagés que par les éléments des vecteurs z pour chaque $r_i = 1$ (les unités non masquées).

Au moment du test, il n'y a pas de dropout, donc les vecteurs de descripteur z sont le plus grands donc ils mettent à l'échelle le vecteur final par la probabilité de Bernoulli $\hat{w} = p.w$. Contraindre les normes l_2 des vecteurs de poids de chaque classe à nombre fixe s.

If $\|w\|_2 > s$, puis le redimensionner de sorte que : $\|w\|_2 = s$.

3.3.2 Expérimentations et résultats

Les auteurs ont conduit leurs expérimentations sur différents benchmarks :

- **MR :** (Movie reviews) commentaires de films avec une phrase par commentaire avec 16448 mots (Pang & Lee, 2005).
- **SST-1** : (Stanford Sentiment Treebank) est une extension de MR mais avec des divisions train / dev / test fournies et des étiquettes à grain fin ré-étiquetées par Socher et al. (2013) avec 16262 mots.
- **SST-2 :** Identique à SST-1, mais avec les reviews neutres supprimés et les étiquettes binaires avec 14838 mots.

- **Subj :** L'ensemble de données de subjectivité où la tâche est de classer une phrase avec 17913 mots (Pang & Lee, 2004).
- **TREC :** est la tâche de l'ensemble de données de question implique de classer une question en 6 types de questions avec 9125 mots (Li & Roth, 2002).
- **CR :** (Customer reviews) commentaires des clients sur divers produits avec 5046 mots(Hu & Liu, 2004).
- **MPQA :** Sous-tâche de détection de polarité d'opinion de l'ensemble de données MPQA avec 6083 mots (Wiebe, et al., 2005).

3.3.2.1 Hyper-paramètres

Pour tous les ensembles de données, ils utilisent:

- ✓ non linéairement: Relu
- ✓ tailles de filtre de fenêtre h = 3, 4, 5
- ✓ Chaque taille de filtre a 100 cartes de descripteurs
- ✓ dropout: p = 0,5
- ✓ Contrainte l_2 pour les lignes de softmax s = 3
- ✓ taille de mini lot pour l'entraînement SGD: 50
- ✓ vecteur de mot: k = 300

3.3.2.2 Vecteurs de mots pré-apprentissage

L'initialisation de vecteurs de mots avec ceux obtenus à partir d'un modèle de langage de neurones non supervisé est une méthode courante pour améliorer les performances en l'absence d'un grand ensemble d'apprentissage supervisé. Les auteurs utilisent les vecteurs word2vec accessibles au public qui ont été formés sur 100 milliards de mots à partir de Google Actualités. Les vecteurs ont une dimension de 300 et ont été formés en utilisant l'architecture continue du sac de mots (Mikolov, et al., 2013). Les mots non présents dans l'ensemble des mots pré-apprentissages sont initialisés aléatoirement.

3.3.2.3 Résultats

Le modèle est validé en l'expérimentant avec plusieurs variantes :

- **CNN-rand :** où tous les mots sont initialisés aléatoirement.

- **CNN-static :** Un modèle avec des vecteurs pré-apprentissage à partir de word2vec. Ils sont conservés statiques.
- **CNN-non-static :** Même chose que ci-dessus mais les vecteurs pré-apprentissages sont finalisés pour chaque tâche.
- **CNN-multichannel :** Un modèle avec deux ensembles de vecteurs de mots. Chaque ensemble de vecteurs est traité comme un «canal» et chaque filtre est appliqué

Contre plusieurs méthodes :

- **RAE :** Autoencodeurs récursifs avec des vecteurs de mots pré-apprentissages de Wikipedia (Socher, et al., 2011).
- **MV-RNN :** réseau neurones récursif matriciel-vectoriel avec arbres d'analyse(Socher, et al., 2012).
- **RNTN :** réseau récursif de neurones récursifs avec fonction tensorielle et arbres d'analyse (Socher, et al., 2013).
- **DCNN :** Réseau de neurones convolutionnels dynamiques (Kalchbrenner, et al., 2014).
- **Paragraph-Vec :** régression logistique au-dessus des vecteurs de paragraphe (Le & Mikolov, 2014).
- **CCAE :** Autoencodeurs de catégorie combinatoire (Hermann & Blunsom, 2013).
- **Sent-Parser :** Analyseur de sentiment-spécifique parser (Dong, et al., 2014).
- **NBSVM, MNB** : Naive Bayes SVM et Multinomial Naive Bayes de Wang et Manning (2012).
- **G-Dropout, F-Dropout :** Dropout gaussien et dropout rapide de Wang et Manning (2013).
- **Tree-CRF:** arbre de dépendances avec champs aléatoires conditionnels (Nakagawa, et al., 2010).
- **CRF-PR :** Champs aléatoires conditionnels avec régularisation postérieure (Yang & Cardie, 2014).
- **SVMs :** SVM avec uni-bi-trigrams, de Silva et al. (2011).

Les résultats témoignent que :

Le modèle de base CNN-rand ne fonctionne pas bien. Alors que les modèles avec pré-apprentissage des vecteurs sont compétitives contres les modèles de deep learning les plus sophistiqués tels que DCNN (Kalchbrenner, et al., 2014), RNTN (Socher, et al., 2013).

3.4 Réseaux de neurones convolutionnels récurrents pour la classification des textes

La classification du texte est une composante essentielle dans de nombreuses applications, telles que la recherche sur le Web, le filtrage de l'information et l'analyse des sentiments (Aggarwal & Zhai, 2012). Un problème clé dans la classification du texte est la représentation des descripteurs. La représentation de descripteurs dans des études antérieures s'effectue sous forme de modèle de sac-de-mot BOW, bigrames, n-grames, ect (Post & Bergsma, 2013).

Huang, et al., Socher, et al., Socher, et al., (2011; 2011; 2013) ont proposé les réseaux de neurones récursifs qui se sont avérés efficaces en terme de construction de représentations de phrases. Mais, la performance des réseaux de neurones récursifs dépend fortement de la performance de l'arborescence textuelle. De plus, la construction d'une telle arborescence requirent une complexité temporelle de $O(n^2)$, où n est la longueur du texte.

Un autre modèle est le réseau de neurones récurrent (RNN). Ce modèle analyse un texte mot à mot et stocke la sémantique de tout le texte précédent (Elman, 1990). L'avantage de RNN est la capacité de mieux capturer l'information contextuelle. Cependant, le RNN est un modèle biaisé, où les mots suivants sont plus dominants que les mots précédents.

Le réseau de neurones convolutionnels (CNN) peut mieux capturer la sémantique des textes par rapport aux réseaux neuronaux récursifs ou récurrents. La complexité temporelle du CNN est $O\ (n)$. Cependant, les études précédentes dans les CNN tendent d'utiliser des noyaux de convolution simples en tant que fenêtre fixe (Collobert, et al., 2011; Kalchbrenner & Blunsom, 2013).

Comme une alternative au problème de la limitation des modèles ci- dessus, dans ce qui suit, nous nous focalisons sur une solution hybride (convolutionnel + récurrente) proposé par Siwei, et al., (2015). Les auteurs proposent un réseau de neurones convolutionnel récurrent (Recurrent Convolutional Neural Network, RCNN) et l'appliquent à la tâche de classification de texte. Le modèle peut réserver une plus large gamme de l'ordre des mots lors de l'apprentissage des représentations de textes.

3.4.1 Modèle RCNN

Le modèle RCNN est un réseau de neurones profond pour capturer la sémantique du texte. La Figure 3.3 montre la structure du réseau de ce modèle. L'entrée du réseau est un document D, qui est une suite de mots $w_1, w_2, \dots, w_n$. La sortie du réseau contient des éléments de la $class$. Ils utilisent $p(k \mid D, \theta)$ Pour indiquer la probabilité que le document soit de la classe k, où θ est le paramètre dans le réseau.

3.4.1.1 Apprentissage de la représentation des mots

Siwei, et al., (2015) combinent un mot et son contexte pour présenter un mot. Les contextes aident à obtenir une signification de mot plus précise. Dans ce modèle, ils utilisent une structure récurrente, qui est un réseau de neurones récurrent bidirectionnel, pour capturer les contextes.

Ils définissent la représentation du mot w_i est la concaténation du vecteur de contexte de gauche $c_l(w_i)$, le word embedding $e(w_i)$ et le vecteur de contexte de droite $c_r(w_i)$.

$$x_i = [c_l(w_i); e(w_i); c_r(w_i)] \qquad (1)$$

Le vecteur de contexte de gauche $c_l(w_i)$ et le vecteur de contexte de droite $c_r(w_i)$ sont calculés respectivement par les équations (2) et (3).

$$c_l(w_i) = f(W^{(l)}.c_l(w_{i-1}) + W^{(sl)}.e(w_{i-1})) \qquad (2)$$

$$c_r(w_i) = f(W^{(r)}.c_r(w_{i+1}) + W^{(sr)}.e(w_{i+1})) \qquad (3)$$

Où $e(w_i)$ est le word embedding du mot w_i. $c_l(w_i)$ est le contexte de gauche du mot w_i. $W^{(l)}$ est une matrice qui transforme la couche cachée (contexte) en la couche cachée suivante. $W^{(sl)}$ est une matrice utilisée pour combiner la sémantique du mot courant avec le contexte gauche du mot suivant. La fonction f est une fonction d'activation non linéaire. Le contexte gauche pour la premier mot dans chaque document utilise le même paramètre partagé $c_l(w_1)$.

Le contexte de droite $c_r(w_i)$ est calculé de manière similaire (Figure 3.3). Le contexte droit du dernier mot dans un document partage le paramètre $c_r(w_n)$.

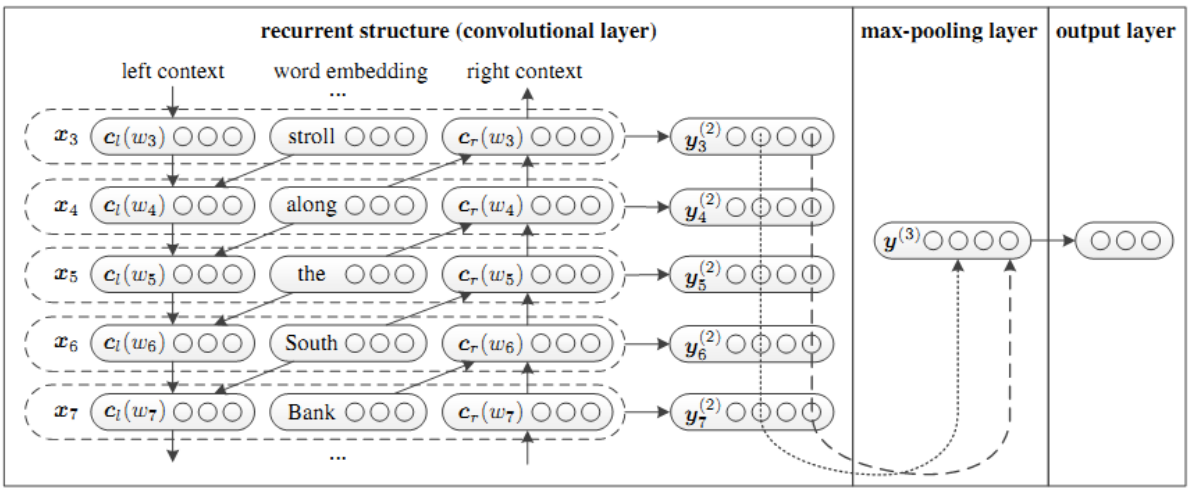


Figure 3.3 : La structure du réseau de neurones convolutionnel récurrent. Cette figure est un exemple partiel de la phrase «A sunset stroll along the South Bank affords an array of stunning vantage points», et l'indice indique la position du mot correspondant dans la phrase originale.

Après le calcul de tout les c_l et tous les c_r nous obtenons représentons x_i du mot w_i (équation (1).

Par la suite, nous appliquons une transformation linéaire avec la fonction d'activation $tanh$ à x_i et envoyer le résultat à la couche suivante.

$$y_i^{(2)} = \tanh(W^{(2)}.x_i + b^{(2)})$$

$y_i^{(2)}$ est un vecteur sémantique latent, dans lequel chaque facteur sémantique sera analysé pour déterminer le facteur le plus utile pour représenter le texte.

3.4.1.2 Apprentissage de la représentation du texte

Le réseau de neurones convolutionnel dans ce modèle est conçu pour représenter le texte. Une couche max-pooling est appliquée sur toutes les représentations des mots. Un texte de différentes longueurs est converti en un vecteur de longueur fixe. Le k-ième élément de $y^{(3)}$ est le maximum des k-ièmes éléments de $y_i^{(2)}$.

$$y^{(3)} = max_{i=1}^{n}(y_i^{(2)})$$

La couche max-pooling tente de trouver les facteurs sémantiques latents les plus importants dans le document.

3.4.1.3 Couche de sortie

La couche de sortie est similaire aux réseaux de neurones traditionnels, elle est définie comme :

$$y^{(4)} = W^{(4)}.y^{(3)} + b^{(4)}$$

Enfin, la fonction softmax est appliquée à $y^{(4)}$.

$$p_i = \frac{\exp(y_i^{(4)})}{\sum_{k=1}^{n} \exp(y_k^{(4)})}$$

3.4.1.4 Apprentissage de RCNN

Cette phase consiste à définir tous les paramètres pour être entrainés en tant que θ.

$$\theta = \{E, b^{(2)}, b^{(4)}, c_l(w_1), c_r(w_n), W^{(2)}, W^{(4)}, W^{(l)}, W^{(r)}, W^{(sl)}, W^{(sr)}\}$$

La cible d'apprentissage du réseau est utilisée pour maximiser la log-probabilité en ce qui concerne θ:

$$\theta \mapsto \sum_{D \in \mathbb{D}} \log \mathrm{p}(class_D \mid D, \theta)$$

où $\mathbb{D}$ est l'ensemble des documents d'apprentissage et $class_D$ est la classe correcte du document D.

La descente de gradient stochastique est utilisée pour optimiser la cible d'apprentissage.

$$\theta \leftarrow \theta + \alpha.\frac{\partial \log p(class_D \mid D, \theta)}{\partial \theta}$$

où α est le taux d'apprentissage.

Dans ce travail, les auteurs utilisent le modèle de Skip-gram pour pré-entrainer le word embedding $w_1, w_2, \ldots, w_T$ en maximisant la probabilité log moyenne.

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c \le j \le c, j \neq 0} \log p(w_{t+j} \mid w_t)$$

$$p(w_b \mid w_a) = \frac{\exp(\acute{e}(w_b)^T e(w_a)}{\sum_{k=1}^{|V|} \exp(\acute{e}(w_k)^T e(w_a))}$$

Où $|V|$ est le vocabulaire du texte non étiqueté. $\acute{e}(w_i)$ est un autre word embedding pour w_i.

3.4.2 Expérimentation et résultats

Les auteurs effectuent des expérimentations pour démontrer l'efficacité de la méthode proposée.

3.4.2.1 Ensembles de données

Pour conduire les expérimentations, les auteurs ont utilisé les quatre ensembles de données suivants :

- 20Newsgroups : cet ensemble de donnés contient les messages à partir de « twenty newsgroups ».
- Fudan Set : Un ensemble de document chinois de l'Université Fudan comprenant 20 classes.
- ACL Anthology Network : un ensemble de documents scientifiques publiés par l'ACL[2]. Le dataset est annoté par langues : d'anglais, japonais, allemand, chinois et français.
- Stanford Sentiment Treebank (SST) : Cet ensemble de données contient les avis sur les filmes analysés et étique par Socher et al. (2013). Les étiquettes forment 5 classes de sentiment, très négatif, négatif, neutre, positif et très positif. analysé et étiqueté par Socher et al. (2013).

3.4.2.2 Hyper-paramètres

Les auteurs ont utilisés les hyper-paramètres couramment utilisés dans les études précédentes :

- ✓ taux d'apprentissage $\alpha = 0.01$
- ✓ taille de la couche cachée $H = 100$
- ✓ dimension de word embedding $|e| = 50$
- ✓ dimension du vecteur de contexte $|c| = 50$
- ✓ Les words embeddings est respectivement pré-apprentissagee sur les dumps wikipedia anglais et chinois par le paramètre par défaut dans word2vec avec l'algorithme skip-grame.

[2] ACL : Association for Computational Linguistics.

3.4.2.3 Résultats

Les approches de réseau de neurones (RecursiveNN, RNN, CNN et RCNN) surpassent les méthodes largement utilisés dans la classification de texte :

- **Bag of Words/Bigrams + LR/SVM** Wang & Manning(2012).
- **Average Embedding + LR :** Cette méthode de base utilise la moyenne pondérée des embedding words et applique ensuite une couche softmax. Huang et al. (2012)
- **LDA (**Latent Dirichlet Allocation**) :** Deux variantes sont utilisés : ClassifyLDA-EM (Hingmire, et al., 2013) et Labeled-LDA (Li, et al., 2008).
- **Tree Kernels (Noyaux d'arbres)** : Deux méthodes majeures de comparaison ont été utilisées: la grammaire sans contexte (CFG) produite par l'analyseur de Berkeley (Petrov, et al., 2006) et l'ensemble de fonctions de relecture de Charniak & Johnson (2005).
- **RecursiveNN :** Deux méthodes ont été selectionnées pour la comparaison : le (RecursiveNN) (Socher, et al., 2011) et sa version améliorée, (Recursive Neural Tensor Networks) (RNTN) (Socher, et al., 2013).
- **CNN :** LA CNN de (Collobert, et al., 2011) est choisie pour la comparaison.

Les réseaux de neurones peuvent capturer plus d'informations contextuelles sur les entités.

La methode RecursiveNN surpasse CNN et RCNN sur l'ensemble de données SST. L'importance de la couche de max-pooling et de la couche de convolution dans la sélection des descripteurs est plus discriminante à travers la couche de max-pooling. La méthode RNTN (3-5 heures et complexité de temps $O(n^2)$) vs. RCNN (plusieurs minutes et complexité de temps $O\ (n)$).

Les résultats de comparaison RCNN contre les méthodes de noyaux d'arbre révèlent que, RCNN pourrait être utile dans les langages à faibles ressources.

En fin, les résultats de comparaisons de RCNN vs. CNN témoignent que la structure récurrente est meilleure que la structure basée sur la fenêtre. Ainsi, RCNN ne dépend pas de la taille de la fenêtre. En terme de performances, RCNN surpasse les méthodes CNN car la structure récurrente peut conserver des informations contextuelles plus longues et introduire moins de bruit.

Chapitre 4
Expérimentation

4.1 Introduction

Dans le présent chapitre, nous nous intéressons à l'implémentation des deux algorithmes abordés dans le chapitre III, à savoir le CNN et RCNN pour la classification de textes. Ces implémentations ont été développées par **Richard Socher** sous le langage **PYTHON** et les codes sources de ces dernières sont disponibles sur :

https://github.com/richliao/textClassifier

En effet nous procédons à la comparaisons des deux techniques CNN et RCNN.

4.2 Environnement matériel

L'environnement matériel utilisé pour conduire les expérimentations est un processeur Intel Core i5 à 2.2 GHZ avec 4GO de RAM sur Windows 10, 64 bit. Tous les algorithmes sont implémentés en Python.

4.3 Environnement de développement

Le langage utilisé est Python, c'est un langage de programmation inventé par **Guido van Rossum** en 1991, il est open source (Gratuit), multiplateformes opérant sur différents systèmes d'exploitation, oriente-objet. Il support l'héritage-multiple et les surcharges des opérations ainsi que d'autres options le rendant un langage simple et puissant à la foi.

Python est présent dans différents domaines, à titre nous exhaustif :

- ✓ Programmation système
- ✓ Web
- ✓ Domaine scientifique
- ✓ Bureautique
- ✓ multimédia

4.3.1 Pourquoi Python ?

Dans notre tâche qui marie entre le traitement automatique du langage naturel et le deep learning, le langage Paython se trouve mieux placé. En effet, il dispose de plusieurs bibliothèques telles que Pandas, Numpy, Tensorflow et Keras.

4.3.2 Bibliothèques

Nous citons dans ce qui suit les principales bibliothèques utilisées pour cette implémentation ainsi que le rôle de chacune.

4.3.2.1 Pandas

Cette bibliothèque permet la manipulation des donnés disposées sous-forme de tableaux et de réaliser sur ces dernier des opérations de l'algèbre relationnel telle que :

- ✓ Projection, la restriction,
- ✓ Opérations ensemblistes (union,….),
- ✓ Produit cartésien, jointures et agrégats.

4.3.2.2 Numpy

Numpy est une bibliothèque qui permet le chargement, le stockage et la manipulation des données venons de sources variées mais qui sont toujours traitées comme des tableaux à N-dimension.

Elle permet de réaliser des opérations mathématiques de haut niveau tel que :

- ✓ Algèbre linéaire
- ✓ Statistiques

4.3.2.3 Tensorflow

Tensorflow est une bibliothèque open-source développé par l'équipe de Google Brain qui implémente les méthodes d'apprentissage automatique basées sur le principe des réseaux de neurones.

4.3.2.4 Keras

Keras est une bibliothèque qui encapsule l'accès aux fonctions proposés par plusieurs bibliothèques dédiées au deep learning, en particulier la bibliothèque TensorFlow. Aussi, elle dispose de fonctions et procédures facilitant les traitements liées au deep learning.

4.4 Données pour l'expérimentation

Nous utilisons le dataset IMDb (Internet Movie Database) pour conduire nos expérimentations. IMDb comprend 25000 avis étiquetés critiquant des films individuels. Chaque avis dispose d'un 'ID' qui est un identifiant unique. La longueur moyenne de l'avis est de 225 mots, aucun film n'a plus de 30 avis.

Le dataset IMDb est spécialement conçu pour l'analyse des sentiments. Le sentiment des avis est binaire, ce qui signifie que si l'évaluation d'IMDB est inférieure à 5 alors le score de sentiment est 0 (sentiment négatif). Par contre si l'évaluation est supérieure ou égale à 7, alors le score de sentiment de 1 (sentiment positif).

Le dataset est téléchargeable sur : **https://www.kaggle.com/c/word2vec-nlp-tutorial/data**

4.5 Préparation de données

Nous commençons par lire le fichier **LabeledTrainData.csv** précédemment téléchargé en utilisant la fonction **read_csv** de la bibliothèque Pandas.

"5814_8" 1 "…biography[,] part feature film which i remember going to see at the cinema when it was originally released[.] Some of it has subtle messages about MJ[']s feeling towards the press and also the obvious message of drugs are bad m'kay[.]

Visually…"

Figure 4.1 : Exemple d'une partie d'un avis.

Dans la Figure 4.1 le code "5814_8" représente un identifiant de l'avis, le "1" dénote un sentiment positif suivi du texte de l'avis. Nous remarquons dans le texte la présence de certaines balises HTML, ponctuations et caractères spéciaux. Dans une perspective d'obtenir un texte brut, nous devons procéder au nettoyage du texte de l'avis en utilisant la bibliothèque **BeautifulSoup**.

La Figure 4.2 montre le résultat suite au nettoyage du texte d'un avis de la Figure 4.1.

"…biography part feature film which i remember going to see at the cinema when it was originally released Some of it has subtle messages about MJ s feeling towards the press and also the obvious message of drugs are bad m kay Visually…"

Figure 4.2 : Texte brut après nettoyage.

Une fois le texte nettoyé, afin de créer notre réseau de neurones, nous devons passer à une représentation matricielle du texte brut.

Dans un premier temps, nous passons à une représentation vectorielle de chaque texte brut. Dans Python, ceci est réalisée par le biais de l'opération Tokenization (la fonction tokenizer). Elle consiste à associer un entier à chaque mot du texte brut.

Vu le nombre important de mots dans chaque avis, on se limite à un nombre que nous définissons au préalable MAX_NB_WORDS qui n'est rien d'autre que le nombre de mots les plus fréquents dans chaque texte.

Après conversion des textes bruts en séquences d'entiers, nous passons à l'étape suivante qu'est la création d'une matrice associée à dataset dont chaque ligne représente un avis. Du moment que les textes des avis sont de tailles variables nous devons trouver le moyen pour les aligner en se limitant à une taille bien précise que nous définissons préalable (MAX_SEQUENCE_LENGTH). Cette possibilité est offerte par Python en utilisant la fonction **PAD_SEQUENCE** de la bibliothèque Keras où chaque vecteur d'un avis de taille inférieur à MAX_SEQUENCE_LENGTH est complété par des 0.

Afin de pouvoir traiter notre dataset, nous devons passer à une nouvelle représentation matricielle dite matrice mot-document des co-occurrences. Cependant, cette matrice ne représente que le nombre d'occurrences d'un mot dans un document (document dans notre cas est l'avis 'review') sans aucune liaison sémantique entre les mots.

Pour qu'on puisse créer les réseaux CNN et RCNN abordés dans le chapitre 3, nous devons avoir en entrée une matrice qui reflète la liaison sémantique entre les mots dans leur contexte (avis). Pour cela, nous pouvons faire recoure aux méthodes d'analyse distributionnelle.

À partir de notre matrice de mot-document des co-occurrences, nous genérons une nouvelle matrice qui s'appelle embedding_matrix en appliquant la méthode **Word2Vec** discuté dans la section 3.2.3.2 Dans Python, cette technique est assurée par la bibliothèque **Gensim**.

Un extrait de matrice embedding_matrix ressemble à ce qui suit :

```
[-0.38497   0.80092   0.064106 -0.28355  -0.026759 -0.34532  -0.64253...
 -0.11729  -0.33257   0.55243  -0.087813  0.9035    0.47102   0.56657...
  0.6985   -0.35229  -0.86542   0.90573   0.03576  -0.071705 -0.12327...
  0.54923   0.47005   0.35572   1.2611   -0.67581  -0.94983   0.68666...
  0.3871   -1.3492    0.63512   0.46416  -0.48814   0.83827  -0.9246...
```

```
-0.33722   0.53741  -1.0616   -0.081403 -0.67111   0.30923  -0.3923...
-0.55002  -0.68827   0.58049  -0.11626   0.013139 -0.57654   0.048833...
  ...         ...       ...       ...       ...      ...            ...]
```

En effet, embedding_matrix représente la distance sémantique entre les mots dans un contexte donné. Qui est la matrice d'entrée pour notre réseau de neurones.

4.6 Apprentissage du modèle

Notre démarche de classification de texte s'inscrit dans l'approche apprentissage supervisé.

Dans notre contexte, pour réaliser l'étude comparative entre les réseaux CNN et RCNN, nous procédons à l'apprentissage de quatre modèle, à savoir : CNN simple, CNN Deep, RCNN LSTM et RCNN GRU.

Pour le modèle CNN simple, nous construisons un classifieur simple à base de réseaux de neurones artificiels où nous considérons une architecture convolutionnelle simple qui utilise 128 filtres de taille 5 et un max pooling de 5 et de 35.

En ce qui concerne le CNN Deep, nous construisons un classifieur impliquant 128 filtres multiples de tailles 3, 4 et 5 et un max pooling de 5 et 35.

Maintenant, pour le RCNN LSTM, nous construisons un classifieur qui se base sur l'architecture Long Short-Term Memory (LSTM) dont nous avons décrit dans la section 2.4.4.

Finalement, pour le RCNN GRU, nous construisons un classifieur qui se base sur l'architecture Gated Recurrent Units (GRU) que nous avons décrit dans la section 2.4.5.

4.7 Validation

Une fois le modèle mis en œuvre, nous devons le tester et valider. Pour ce faire, nous avons calculé des métriques d'évaluation qui consiste à comparer les résultats de la classe prédite par rapport à la classe réelle.

Il est à noter que nous avons utilisée la méthode de validation 20/80. Autrement dit, le dataset est scindé en deux parties : la première partie contient 20000 avis pour entraînements et la deuxième partie contient 5000 avis pour le test.

Nous détaillons et interprétons dans ce qui suit les métriques utilisés par cette expérimentation.

4.7.1 Métriques utilisées

- **Matrice de confusion**

Pour une classification binaire, comme dans le cas de notre implémentation, nous distinguons 4 cas de classements possibles.

Vrai positif VP : ce sont des éléments positifs bien prédits.

Vrai négatif VN : ce sont des éléments négatifs bien prédits.

Faux positif FP : ce sont des éléments positifs mal prédits.

Faux négatif FN : ce sont des éléments négatifs mal prédits.

Le principe de la matrice de confusion peut être schématisé par la Table 4.1.

Table 4.1 : Principe de la matrice de confusion.

Prédictive \ Réel	Positive	Négative
Positive	VP	FP
Négative	FN	VN

.

- **Accuracy**

L'accuracy est un ratio entre les éléments bien classés et le nombre total d'exemples, dans le cas d'une classification binaire l'accuracy est calculée à partir de la matrice de confusion :

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

- **Erreur de classification**

L'erreur de classification est un ratio entre les éléments mal classés et le nombre total d'exemples, dans le cas d'une classification binaire l'erreur de classification est calculer à partir de la matrice de confusion comme suit :

$$Erreur\ de\ classification = \frac{FV + FN}{VP + VN + FP + FN}$$

- **Précision**

La précision est un ratio entre les éléments positives bien classés et le nombre positive total, dans le cas d'une classification binaire la précision est calculée à partir de la matrice de confusion comme suit :

$$précision = \frac{VP}{VP + FP}$$

- **Rappel**

Le rappel est un ratio entre les éléments positives bien classés et le nombre d'éléments de la classe à prédite, dans le cas d'une classification binaire le rappel est calculé à partir de la matrice de confusion comme suit :

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

- **F1-mesure**

F1-mesure est une mesure de compromis entre précision et rappel, calculée comme suit :

$$F1 - mesure = 2 * \frac{précision * rappel}{précision + rappel}$$

4.7.2 Résultats

Dans cette section, nous présentons les résultats obtenus à l'issue de la phase validation.

Table 4.2 : Résultats du classifieur CNN simple.

Classe	Précision	Rappel	F1-mesure	Accuracy	Support
Positive	0.50	0.96	0.66		2492
Négative	0.46	0.03	0.06		2508
Avg/ Total				0.89	5000

Les résultats sont présentés en considérons les quatre classifieur construisant dans la phase d'apprentissage. La Table 4.2 montre les résultats pour le classifieur CNN simple.

Alors que la Table 4.3 montre les résultats pour le classifieur CNN Deep.

Table 4.3 : Résultats du classifieur CNN Deep.

Classe	Précision	Rappel	F1-mesure	Accuracy	Support
Positive	0.87	0.47	0.52		2510
Négative	0.51	0.2	0.14		2490
Avg/ Total				0.903	5000

Les résultats du classifieur RCNN LSTM sont affichés dans la Table 4.4.

Table 4.4 : Résultats du classifieur RCNN LSTM.

Classe	Précision	Rappel	F1-mesure	Accuracy	Support
Positive	0.53	0.31	0.39		2538
Négative	0.50	0.72	0.59		2462
Avg/ Total				0.904	5000

En fin, la Table 4.5, illustre les résultats du classifieur RCNN GRU.

Table 4.5 : Résultats du classifieur RCNN GRU.

Classe	Précision	Rappel	F1-mesure	Accuracy	Support
Positive	0.49	0.38	0.47		2504
Négative	0.55	0.6	0.51		2496
Avg/ Total				0.904	5000

4.7.3 Discussion

À titre de rappel, lorsque nous comparons les approches de réseau de neurones (CNN et RCNN) aux méthodes traditionnelles largement utilisées (par exemple, BoW + LR), il s'avère

que l'approche basée sur un réseau de neurones peut efficacement composer la représentation sémantique des textes. Les réseaux de neurones peuvent capturer plus d'informations contextuelles des descripteurs par rapport aux méthodes traditionnelles basées sur le modèle BoW, et peuvent résister plus au problème de la rareté des données (chapitre 3).

Sur la base des résultats présentés dans la section 4.7.2, concernent le classifieur CNN, il semble que la complexité des réseaux de neurones convolutionnels n'améliore pas les performances du modèle de classification, du moins en utilisant un dataset plus grand, que nous ne serons pas en mesure de vérifier dans notre contexte.

En ce qui concerne le classifieur RCNN, le changement de la méthode de récurrence (LSTM et GRU) n'influe pas considérons sur les performances du classifieur.

Nous estimons qu'il est possible d'améliorer les performances des classifieurs CNN et RCNN comme suite :

- Affiner les hyper-paramètres
- prétraitement du texte
- utiliser la couche de dropout.

Enfin, nous pouvons observer que le classifieur RCNN outrepasse en terme de performances le classifieur CNN. Ceci, peut être du à la manière de capturer l'information contextuelle. Les réseaux CNN utilisent une fenêtre de mots fixe comme information contextuelle, tandis que les RCNN utilisent la structure récurrente pour capturer un large éventail d'informations contextuelles.

Conclusion générale

Conclusion générale

Dans l'ère de la révolution numériques, le traitement manuel des données devient difficile, voire impossible. Le recou à des méthodes intelligentes d'analyse de données se propose comme une alternative à ce problème.

Parmi les méthodes intelligentes d'analyse de données, la classification du texte est largement utilisée. Notre travail consiste à aborder le problème de classification de texte en utilisant la nouvelle technique deep learning. Nous avons d'abord introduit la technique du deep learning. Par la suit, nous avons étudié l'impact du deep learning sur la classification du texte en conduisant une étude empirique comparative de architecture convolutionnelles (Convolutional Neural Network, CNN) et récurrent (Recurrent Convolutional Neural Networks, RCNN). Le dataset IMDb utilisée pour l'expérimentation. Les résultats ont révélé l'impact visible du deep learning sur la classification du texte.

Pour en tirer d'avantages conclusions, nous envisageons étudie nos expérimentation sur des datasets plus consistants.

Malheureusement, actuellement, avec l'environnement disponible, il n'est pas possible de conduire ces expérimentations.

Bibliographie

Bibliographie

Aggarwal, C. & Zhai, C., 2012. A survey of text classification algorithms. Dans: *In Mining text data.* s.l.:s.n.

Arnold, L., 2013. *Learning Deep Representations : Toward a better new understanding of the deep learning paradigm,* paris: s.n.

Bengio, Y., 2009. *Learning Deep Architectures for AI ,* s.l.: Université de Montréal Canada.

Bengio, Y., Lamblin, P., Popovici, V. & Larochelle, H., 2007. *Greedy layer-wise training of deep networks. Advances in Neural Information Processing Systems,* s.l.: s.n.

Benoît, L., Maryse, B., Emmanuel, S. & Ira, N., 2001. *Un modèle de compréhension de textes fondé sur l'analyse de la sémantique latente,* France: Université Pierre Mendès.

Bestgen, Y., Degand, L. & Spooren, W., 2003. On the use of automatic techniques to determine the semantics of connectives. *Determination of information and tenor in texts: Multidisci-plinary approaches to discourse.*

Bishop, J. & Christopher, M., 1995. *Neural Networks for Pattern Recognition.,* New York: s.n.

Bouaziz, M., 2017. *Réseaux de neurones récurrents pour la classification de séquences dans des flux audiovisuels parallèles,* s.l.: s.n.

Ceri, S. et al., 1998. ACM Computing Classification. Dans: *« Web Information Retrival ».* s.l.:ISBN 978-3-642-39313-6.

Charniak, E. & Johnson, M., 2005. *Coarse-to-fine n-best parsing and maxent discriminative reranking.,* s.l.: In ACL.

Cho, k., van Merrienboer, B., Bahdanau, D. & Bengio, Y., 2014. *On the properties of neural machine translation: Encoder-decoder approaches.* s.l.:arXiv preprint arXiv:1409.1259.

Chung, J., Gulcehre, C., Cho, K. & Bengio, Y., 2014. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling.* s.l.:In: arXiv preprint arXiv:1412.3555.

Collobert, R. et al., 2011. Natural Language Processing (Almost) from Scratch.. *Journal of Machine Learning Research.*

Cybenko, G., 1989. *Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals, and Systems (MCSS),* s.l.: s.n.

Dechter, R. & Pearl, J., 1986. *The cycle-cutset method for improving search performance in AI applications,* s.l.: University of California, Computer Science Department.

Deerwester, S. et al., 1990. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science,* 41(6).

Deng, L. & Dong, Y., 2014. *Deep Learning: Methods and Applications,* s.l.: s.n.

Deng, L. & Yu, D., 2014. *Deep Learning: Methods and Applications,* s.l.: s.n.

Dong, L. et al., 2014. A Statistical Parsing Framework for Sentiment Classification..

Elman, J., 1990. Finding structure in time.. Dans: *Cognitive Science.* s.l.:s.n.

Fabre, C. & Lenci, A., 2015. Distributional Semantics Today Introduction to the spicial issue.. Dans: *In Traitement automatique des langues. Sémantique distributionnelle.* s.l.:s.n.

Harris, S., 1954. Distributional structure. Dans: s.l.:s.n.

Herculano-Houzel, S., 2009. *The human brain in numbers: a linearly scaled-up primate brain,* USA: s.n.

Hermann, K. & Blunsom, P., 2013. *The Role of Syntax in Vector Space Models of Compositional Semantics.* s.l.:In Proceedings of ACL.

Hingmire, S., Chougule, S., Palshikar, G. & Chakraborti, S., 2013. *Document classification by topic labeling,* s.l.: In SIGIR.

Hinton, G., Ronald, J. & Williams, R., 1986. Learning internal representations by error propagation.. Dans: *Paral lel distributed processing: explorations in the microstructure of cognition MA.* USA: s.n.

Hinton, G., Sutskever, I. & Geoffrey, E., 2008. Deep, narrow sigmoid belief networks are universal approximators.. Dans: *Neural Comput..* s.l.:s.n.

Hocherieter, S. & Schmidhuber, J., 1997. Long short-term memory.. Dans: *Neural computation.* s.l.:s.n.

Hochreiter, S., Bengio, Y., Frasconi, P. & Schmidhuber, J., 2001. Gradient flow in recurrent nets : the difficulty of learning long-term dependencies.. Dans: s.l.:s.n.

Hornik, K., 1989. Multilayer feedforward networks are universal approximators.. Dans: *Neural Networks.* s.l.:ISSN 0893-6080.

Huang, E., Socher, R., Manning, C. & Ng, A., 2012. *Improving word representations via global context and multiple word prototypes,* s.l.: In ACL, 873–882.

Huang, H. et al., 2011. *Dynamic pooling and unfolding recursive autoencoders for paraphrase detection,* s.l.: In NIPS.

Hu, M. & Liu, B., 2004. *Mining and Summarizing Customer Reviews.* s.l.:In Proceedings of ACM SIGKDD.

Huxley, A., Hodgkin, L. & Andrew, F., 1952. A quantitative description of membrane current and its application to condution excitation in never. *journal of physiology.*

Kalchbrenner, K., Grefenstette, E. & Blunsom, P., 2014. *A Convolutional Neural Network for Modelling Sentences.* s.l.:In Proceedings of ACL.

Kalchbrenner, N. & Blunsom, P., 2013. *Recurrent convolutional neural networks for discourse compositionality,* s.l.: In Workshop on CVSC.

Kim, Y., 2014. *Convolutional Neural Networks for Sentence Classification,* s.l.: s.n.

Kriesel, D., 2005. Dans: *Neural Networks.* Germany: s.n.

Krizhevsky, Sutskever, I. & Hinton, G., 2012. *ImageNet Classification with Deep Convolutional Neural neworks.* s.l.:In Proceedings of NIPS 2012.

Landauer, K., Foltz, W. & Laham, D., 1998. An introduction to Latent Semantic Analysis. *Discourse Processes,* Volume vol. (25) .

Landauer, T. & Dumais, S., 1997. A solution to Plato's problem : the Latent Semantic Analysis theory of acquisition, induction and representation of knowledge.

Larochelle, H., 2009. *Étude de techniques d'apprentissage non-supervisé pour l'amélioration de l'entraînement supervisé de modèles connexionnistes,* s.l.: s.n.

Le Cun, Y., 1989. *Generaliation and Network Design Strategies ,* s.l.: s.n.

Le Cun, Y. et al., 1990. Handwritten digit recognition with a back-propagation network. Dans: *Advances in neural information processing systems.* San Francisco, USA: Morgan Kaufmann Publishers Inc.

LeCun, Y. & Bengio, Y., 2007. *Scaling learning algorithms towards ai. In Large-Scale Kernel Machines.,* s.l.: s.n.

LeCun, Y. et al., 1989. *convolutionnel neural networks,* s.l.: s.n.

LeCun, Y., Bottou, L., Bengio, Y. & Haffiner, P., 1998. Gradient-based learning applied to document recognition.. Dans: s.l.:Proceedings of the IEEE.

LeCun, Y., Haffner, P. & Bottou, L., 1998. *Object recognition with gradient based learning. In: Shape, contour and grouping in computer vision..* s.l.:Springer Berlin Heidelberg.

Le, Q. & Mikolov, T., 2014. *Distributed Represenations of Sentences and Documents..* s.l.:In Proceedings of ICML.

Li, W., Sun, L. & Zhang, D., 2008. Text classification based on labeled-lda model. *Chinese Journal of Computers,* Volume 31.

Li, X. & Roth, D., 2002. *Learning Question Classifiers.* s.l.:In Proceedings of ACL.

Mikolov, T. et al., 2013. *Distributed Representations of Words and Phrases and their Compositionality.* s.l.:In Proceedings of NIPS 2013.

Minsky, L. & Papert, S., 1969. *Perceptrons: An introduction to computational geometry,* s.l.: s.n.

Nakagawa, T., Inui, K. & Kurohashi, S., 2010. *Dependency tree-based sentiment classification using CRFs with hidden variables.,* s.l.: In Proceedings of ACL.

Pang, B. & Lee, L., 2004. *A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts.* s.l.:In Proceedings of ACL.

Pang, B. & Lee, L., 2005. *Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales.* s.l.:In Proceedings of ACL.

Pascanu, R., Mikolov, T. & Bengio, Y., 2013. On the difficulty of training recurrent neural networks. Dans: s.l.:s.n.

Petrov, S., Barrett, L., Thibaux, R. & Klein, D., 2006. *Learning accurate, compact, and interpretable tree annotation,* s.l.: In Coling-ACL.

Post, M. & Bergsma, S., 2013. *Explicit and implicit syntactic features for text classification,* s.l.: In ACL.

Razavian, A., Azizpour, H., Sullivan, J. & Carlsson, S., 2014. *CNN Features off-the-shelf: an Astounding Baseline..* s.l.:s.n.

Rojas, R., 1996. The backpropagation algorithm. Dans: *Neural Networks..* NewYork: Berlin Heidelberg.

Rumelhart, D., Hinton, G. & Williams, R., 1985. Learning internal representations by error propagation.. Dans: s.l.:s.n.

Rumelhart, D., Hinton, G. & Williams, R., 1986. Learning representations by backpropagating errors. Dans: s.l.:s.n.

Schmidhuber, J., 2014. *Deep learning in neural networks : An overview,* s.l.: University of Lugano & SUPSI.

Siwei, L., Liheng, X., Kang, L. & Jun, Z., 2015. *Recurrent Convolutional Neural Networks for Text Classification,* s.l.: s.n.

Socher, R., Huval, B., Manning, C. & Ng, A., 2012. *Semantic Compositionality through Recursive Matrix-Vector Spaces.* s.l.:In Proceedings of EMNLP.

Socher, R. et al., 2011. *Semi-supervised recursive autoencoders for predicting sentiment distributions,* s.l.: In EMNLP.

Socher, R. et al., 2013. *Recursive deep models for semantic compositionality over a sentiment treebank,* s.l.: In EMNLP.

Srivastava, N., Krizhevsky, A., Sutske, I. & Hinton, G., 2012. Improving neural networks by preventing co-adaptation of feature detectors.

Turney, D. & Pantel, P., 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research,* 37(1).

Wang, S. & Manning, C., 2012. *Baselines and bigrams: Simple, good sentiment and topic classification,* s.l.: In ACL.

Werbos, P., 1988. Generalization of backpropagation with application to a recurrent gas market model. Dans: *Neural Networks.* s.l.:s.n.

Wiebe, J., Wilson, T. & Cardie, C., 2005. Annotating Expressions of Opinions and Emotions in Language.. Dans: *Language Resources and Evaluation.* s.l.:s.n.

Yang, B. & Cardie, C., 2014. *Context-aware Learning for Sentence-level Sentiment Analysis with Posterior Regularization.,* s.l.: In Proceedings of ACL.