UNIVERSITÉ MOHAMMED SEDDIK BENYAHIA
JIJEL
FACULTÉ DES SCIENCES EXACTES ET D'INFORMATIQUE

MEMOIRE DE MASTER

Présenté pour l'obtention du diplôme de :

MASTER

En **INFORMATIQUE**

Option : INTELLIGENCE ARTIFICIELLE

Par :

Deghdegh Tarek et Boulahdid Aymen

Thème

Investigation d'un classifieur basé sur un réseau de neurones à impulsions

Encadré par :

Dr. Zennir Mohamed Nadjib

Promotion 2021

Remerciements

Tout d'abord, nous remercions ***ALLAH*** *Tout-Puissant de nous avoir donné la patience et la force de nous offrir cet modeste travail.*
Ensuite, nous tenons à remercier ceux qui méritent ***Dr. ZENNIR Mohammed Nadjib*** *, qui, sinon pour ses conseils avisés, qui nous ont aidés et ont joué le plus grand rôle dans la réalisation de ce travail, et à partir de cette page, nous vous adressons, Professeur, nos plus chaleureux remerciements et gratitude d'être avec nous.*
Dès lors, nous tenons à remercier tous les membres de l'honorable ***famille*** *qui ne nous ont pas apporté leur soutien moral et matériel et ne nous ont pas abandonnés.*
Nous remercions également ***les amis*** *qui nous ont aidés et contribué à la réalisation de ce modeste travail.*
Nous remercions également ***les membres du jury*** *qui nous ont honorés en acceptant le jugement et en enrichissant cet modeste travail.*

Dédicace

Je dédie ce modeste travail

- **Ma mère** et rien n'est plus précieux que la mère pour son sacrifice, son amour, sa tendresse et son soutien pour moi tout au long de la période d'étude et ses prières pour moi.
- **Mon père** pour son soutien et sa présence à mes côtés et mon soutien tout au long de la période d'étude.
- **A ma tante Karima**, à qui je jure devant Dieu que s'ils me donnaient un million de pages pour la remercier et parler d'elle, je suis sûr que ce ne sera pas suffisant.
- **A mon cher grand-père et ma grand-mère bien-aimée** pour leurs prières et leur soutien.
- **A mes tantes** Wafaa, Sumaya et Linda pour leur soutien et leurs prières pour moi.
- **A mes frères** Oussama, Elias, Manal, Rehab.
- **A mes oncle** Nesredin,fateh pour leur soutien
- **A mes amis** Salem ,Moncef, Zaki, Mohamed, Fouad, Mouad, Draa Idriss Hichem, Yahya Sid Ali, Islam, Oussama, Hossam, Alaa, Walid.
- **A mon binôme Tarek** qui ne s'est pas consacré à fournir ce travail

Aymen

Dédicace

Je dédie ce modeste travail

- **À mes très chers parents**, source de vie, d'amour et d'affection.
- **À mes chers frères**, source de. joie et de bonheur.
- **À toute ma famille**, Source d'espoir et de motivauon.
- **À tous mes amis**, particulièrement Islem. ...
- **À mon encadreur**, Monsieur Mohamed Nadjib ZENNIR.
- **À tout mes enseignants**, partuclierement Oulmane Sara et Birouk Wafa.

Merci pour tout.

Tarek

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

LISTES DES ACRONYMES

IA	Intelligence Artificial
RN	Réseau de Neurones
RNA	Réseau de Neurones Artificial
MLP	MuLticouches Perceptron
SNN	Réseau de Neurones Impulsionnnels
BNN	Réseaux de Neurones Biologiques
3D	Trois Dimension
LIFM	Leaky Integrate and Fire
HHM	Hodgkin Huxley
Na	sodium
K	Potassium
FHN	FitzHugh-Nagumo
IM	Modèle d'Izhikevich
BP	Back Propagation
STDP	Spike Timing Dependent Plasticity
LTP	Long Term Potentiation
LTD	Long Term Depression
NAMDA	récepteurs sur la membrane du neurone
AMPA	récepteurs sur la membrane du neurone

Introduction générale

Un classifieur en apprentissage profond est un algorithme qui ordonne ou catégorise automatiquement les données dans une ou plusieurs "classes". L'un des exemples les plus courants est l'analyse des sentiments, qui consiste à détecter les sentiments positifs ou négatifs dans un texte. Elle est souvent utilisée par les entreprises pour détecter les sentiments dans les données sociales, évaluer la réputation de la marque et comprendre les clients. Étant donné que les clients expriment leurs pensées et leurs sentiments plus ouvertement que jamais sur les médias sociaux parce qu'ils leur donnent une liberté totale, l'analyse du sentiment devient un outil essentiel pour surveiller et comprendre ce sentiment.
Les algorithmes d'apprentissage profond sont utiles pour automatiser des tâches qui devaient auparavant être effectuées manuellement. Ils permettent d'économiser d'énormes quantités de temps et d'argent et rendent les entreprises plus efficaces.
Par conséquent, au cours de cette thèse, nous avons créé un classifieur pour l'analyse du sentiment basé sur la prochaine génération de réseaux neuronaux connus sous le nom de réseaux de neurones impulsionnels, pour lui permettre d'acquérir des connaissances profondes pendant la phase d'apprentissage qui lui permet de classer automatiquement .
Nous avons emprunté au cerveau humain le principe du travail de ces neurones et leur mode d'activité. Nous avons utilisé un neurone de type Izhikevich en raison de sa grande efficacité lors du calcul lorsque ces neurones sont utilisés en grand nombre, et un dataset tiré des sites de réseaux sociaux pour le dialecte tunisien. Dans les chapitres de ce mémoir, nous expliquerons toutes les étapes de la création de notre classifieur, à partir de la création d'un seul neurone impulsionnel jusqu'à l'obtention d'un système capable de classifier avec un pourcentage de précision qui atteint 75%. Nous aborderons dans les chapitres de cette recherche les points suivants :

- Lors du premier chapitre, nous évoquerons les réseaux neuronaux classiques et leurs méthodes d'entraînement.
- Dans le deuxième chapitre, nous présenterons de manière superficielle et simplifiée le cortex, puis nous énumérerons les types de neurones pulsés et les méthodes d'entraînement les plus utilisées pour ceux-ci.
- Dans le troisième chapitre, nous déroulerons les étapes de la construction de notre classifieur impulsionnel basé sur ce type de cellules, et comment nous avons traité les problèmes et obstacles rencontrés.
- Lors du chapitre final, nous décrirons l'environnement de travail et les outils utilisés pour créer ce classifieur, ainsi que la phase de test, et nous avons discuté des résultats obtenus.

CHAPITRE 1

LES RÉSEAUX DE NEURONES ARTIFICIELS.

1.1 Introduction

Le réseau de neurones (RN) est un paradigme de traitement de l'information inspiré du système nerveux humain.
Un réseau de neurones artificiels (RNA) est un ensemble d'algorithmes qui simulent le principe de travail et d'organisation des neurones biologiques d'un organisme vivant. Les algorithmes sont utilisés pour traiter des données et accomplir des tâches dans une variété de domaines. Les réseaux de neurones artificiels sont parmi les types les plus connus d'apprentissage automatique qui fournissent un logiciel capable d'apprentissage avec l'expérience. Le RNA est utilisé dans divers problèmes d'application, tels que la reconnaissance de formes, la classification de données, la reconnaissance vocale, le traitement d'images et l'identification du système à travers différentes procédures d'apprentissage. Les neurones sont connectés les uns aux autres par des liens de connexion appelés "poids". Ces poids, lorsqu'ils pondèrent l'entrée, donnent le potentiel net pour tout réseau de neurones typique. La sortie du réseau est obtenue en appliquant des fonctions d'activation à l'entrée du réseau. [1]
Un réseau artificiel se caractérise par :

- Son architecture (connexion entre neurones).
- Sa formation ou apprentissage (détermination des poids sur les connexions).
- Ses fonctions d'activation.

La figure 1.1 montre un simple réseau de neurones artificiels :

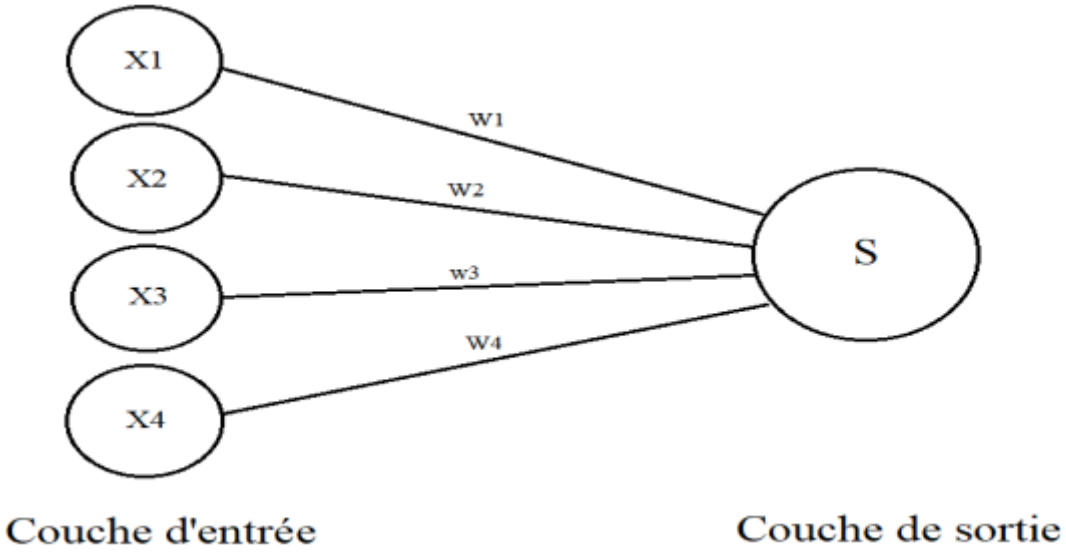


FIGURE 1.1 – Un simple réseau de neurones artificiels.

— Dans la figure précédente, il existe quatre neurones d'entrée (X_1, X_2, X_3, X_4) dans la couche d'entrée et un neurone de sortie 'S', Les poids sur les connexions de X_1, X_2 , X_3 et X_4,au neurone S sont donnés par W_1, W_2 , W_3 et W_4.

— Les activations (signaux de sortie) de ces neurones sont respectivement X_1, X_2, X_3 et X_4. L'entrée nette, Y, du neurone S est la somme des signaux pondérés des neurones X_1, X_2, X_3 et X_4, c'est-à-dire :

$$Y = W_1 * X_1 + W_2 * X_2 + W_3 * X_3 + W_4 * X_4 \tag{1.1}$$

L'activation yin du neurone S est donnée par une fonction de son potentiel.

$$S = F(Y) \tag{1.2}$$

1.2 Le neurone de McCulloch–Pitts

Le réseau de neurones McCulloch-Pitt est considéré comme le premier réseau de neurones. Les neurones sont connectés par des chemins pondérés dirigés [2] [1]. Le neurone de McCulloch-Pitt permet une activation binaire (1 ON ou 0 OFF), c'est-à-dire qu'il se déclenche avec une activation 1 ou ne se déclenche pas avec une activation 0. Si $w > 0$, alors le chemin connecté est dit être excitateur sinon il est connu comme inhibiteur. Les connexions excitatrices ont des poids positifs et les connexions inhibitrices ont des poids négatifs. Chaque neurone a un seuil fixe de déclenchement. C'est-à-dire que si l'entrée nette du neurone est supérieure au seuil, il se déclenche. [3] [1]
L'architecture du neurone de McCulloch-Pitts est illustrée à la figure 1.2 "Y" est le neurone de McCulloch-Pitts, il peut recevoir le signal d'un nombre quelconque d'autres neurones [4] [1]. Les poids de connexion de $X_1 \dots X_n$ sont excitateurs, notés " W " et les poids de connexion de $X_{n+1} \dots X_{n+m}$ sont inhibiteurs notés " $-P$ ". Le neurone de McCulloch-Pitts Y a une fonction d'activation comme :

$$f(yin) = \begin{cases} 1 & \text{if } yin \geq \theta \\ 0 & \text{if } yin \leq \theta \end{cases} \tag{1.3}$$

Où θ est le seuil et « yin » est le signal d'entrée net total reçu par le neurone Y. L'inhibition doit être absolue. Par conséquent, le seuil satisfait « $\theta > nw - p$ ». Le neurone de sortie se déclenche s'il reçoit k entrées excitatrices ou plus sans entrées inhibitrices, où « $KW \geq \theta > (K-1)W$ ». [1]

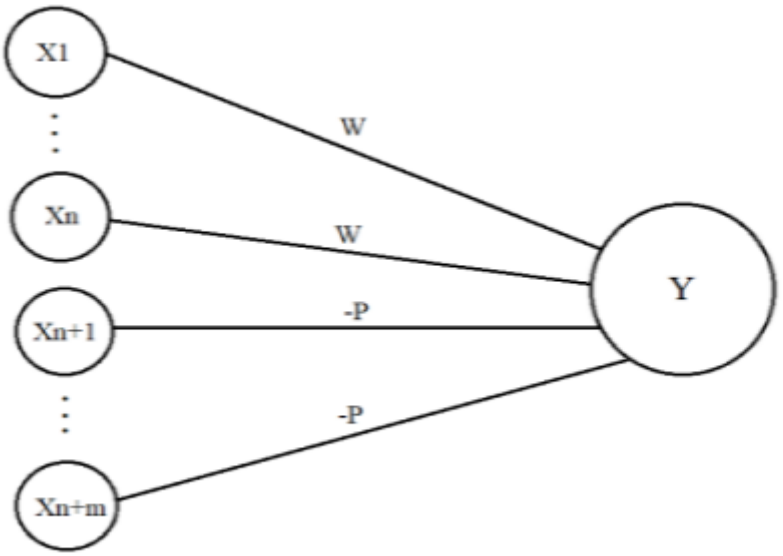


FIGURE 1.2 – Architecteure de McCulloch–Pitts.

1.3 Le Perceptron

Un perceptron est un type de neurone artificiel qui modélise le comportement d'un neurone biologique. Un perceptron est une fonction qui produit une sortie pour un ensemble fourni de valeurs d'entrée. La figure 1.3 donne une représentation visuelle d'un perceptron. Un perceptron accepte une, deux ou plusieurs valeurs numériques comme entrées. Il produit une valeur numérique en sortie.
Un perceptron opère sur des nombres, ce qui signifie que les entrées et la sortie sont des valeurs numériques (par exemple, des nombres entiers ou des valeurs à virgule flottante).

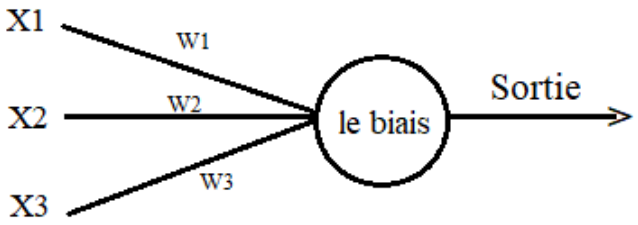


FIGURE 1.3 – Représentation d'un perceptron.

La figure 1.3 représente un perceptron. Toutes les entrées n'ont pas la même importance pour le perceptron. Par exemple, une entrée peut être plus importante que d'autres entrées. La pertinence d'une entrée est exprimée à l'aide d'un poids (également une valeur numérique) associé à cette entrée. Dans la figure 1.3, l'entrée x_1 est associée au poids w_1, x_2 au poids w_2 et x_3 à w_3. Différentes pertinences de certaines entrées permettent au réseau de modéliser un comportement spécialisé. Par exemple, pour une tâche de reconnaissance d'image, les pixels situés en bordure de l'image ont généralement moins de pertinence que les pixels situés au milieu. Poids associés avec les entrées correspondant aux pixels de bordure sera donc plutôt proche de zéro. En plus de la valeur d'entrée pondérée, un perceptron nécessite un biais, une valeur numérique servant de seuil. Nous désignons le biais par « b ».
Un perceptron reçoit un stimulus en entrée et répond à ce stimulus en produisant une valeur de sortie. La sortie obéit à une règle très simple : si la somme des entrées pondérées est supérieure à une valeur donnée, alors le perceptron tire 1, sinon, il déclenche 0. Par programmation, nous calculons d'abord la somme des entrées pondérées et du biais. Si cette somme est strictement supérieure à 0, alors le perceptron produit 1, sinon, il produit 0.
Formellement, sur la base du perceptron donné dans la figure 1.3, nous écrivons
$z = x_1 * w_1 + x_2 * w_2 + x_3 * w_3 + b$. Dans le cas général, on écrit $\sum_n^i (x_i * w_i + b)$. La variable i s'étend sur toutes les entrées du perceptron. Si $z \geq 0$, alors le perceptron produit 1 ou Si $z \leq 0$, il produit 0. [5]

1.4 Le Perceptron multicouches

En 1969, Minsky et Papert relèvent que le perceptron échoue pour des problèmes de classification simples, comme ceux où les classes ne sont pas linéairement séparables. Pour expliquer ce cas, le meilleur exemple est la fonction XOR. La figure 1.4 représente la séparation entre les fonctions ''XOR" ''ET" ''OR ''. Nous remarquons sur la figure 1.4 que le processus de séparation linéaire est possible en ''ET '' et ''OR", mais dans la fonction ''XOR" il est impossible, où les motifs (0 ; 0) Et (1 ; 1) appartiennent à une classe tandis que les motifs (1 ; 0) et (0 ; 1) appartiennent à une autre classe. [6]

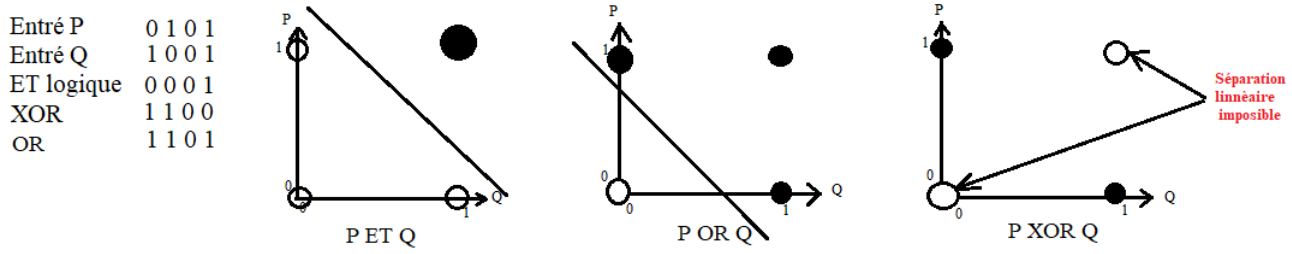


FIGURE 1.4 – Séparation linéaire impossible.

Dans le modèle du Perceptron Multicouches, les perceptrons sont organisés en couches. Les perceptrons multicouches sont capables de traiter des données qui ne sont pas linéairement séparables. Avec l'arrivée des algorithmes de rétro propagation, ils deviennent le type de réseaux de neurones le plus utilisé.
Les MLP sont généralement organisés en trois couches, la couche d'entrée, la couche intermédiaire (dite couche cachée) et la couche de sortie. L'utilité de plusieurs couches cachées est dû à un apprentissage plus approfondi.
La figure 1.5 nous montre la différence entre le perceptron simple et le perceptron multicouches et il illustre la structure d'un MLP présentant trois neurones en entrée, deux couches cachée, chaque couche cachées contient trois neurones et deux neurones en sortie. Lorsque tous les neurones d'une couche sont connectés aux neurones de la couche suivante, on parle alors de couches complètement connectées.

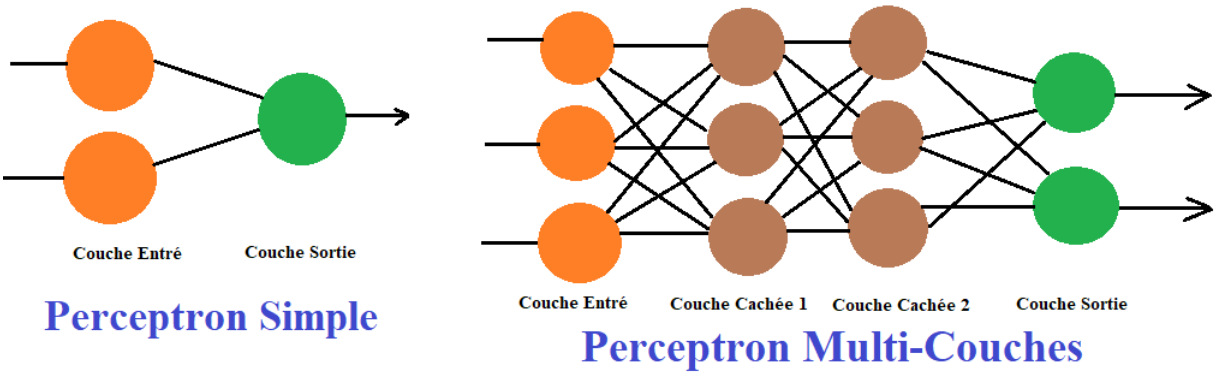


FIGURE 1.5 – Présentation des types de perceptron.

1.5 La Fonction d'activation

Dans le domaine des réseaux de neurones artificiels, la fonction d'activation est une fonction mathématique non linéaire appliquée à un signal en sortie d'un neurone artificiel. Le terme de fonction d'activation est synonyme de potentiel d'activation en biologie. C'est

un seuil de stimulation qui une fois atteint entraîne une réponse du neurone. Les fonctions d'activation sont utilisées selon leurs caractéristiques d'étendue de non linéarité, de différentiabilité de convergence vers l'identité en 0 de continuité ou de monotonie. Plusieurs fonctions d'activation ont été mises au point. [7] Nous présentons les plus connues :

— Identité :

$$\phi(x) = x \tag{1.4}$$

— Sigmoïde :

$$\phi(x) = \frac{1}{1 + \mathrm{e}^{-\mathrm{x}}} \tag{1.5}$$

— Tangente hyperbolique :

$$\phi(x) = \frac{2}{1 + \mathrm{e}^{-2\mathrm{x}}} - 1 \tag{1.6}$$

1.6 La propagation et la rétropropagation du gradient

Soit le couple $(\vec{x}(n), \vec{d}(n))$ désignant la n^e donnée d'entraînement du réseau ou :

$$\vec{x}(n) =< x_1(n), ..., x_p(n) > \text{ et } d(n) =< d_1(n), ..., d_q(n) > \tag{1.7}$$

correspondent respectivement aux p entrées et aux q sorties désirées du système. L'algorithme de rétro propagation consiste alors à mesurer l'erreur entre les sorties désirées $\vec{d}(n)$ et les sorties observées $\vec{y}(n)$:

$$\vec{y}(n) =< y_1(n), ..., y_p(n) > \tag{1.8}$$

résultant de la propagation vers l'avant des entrées $\vec{x}(n)$, et à rétro propager cette erreur à travers les couches du réseau en allant des sorties vers les entrées.

1.6.1 Cas de la couche sortie

L'algorithme de rétro propagation procède à l'adaptation des poids neurone par neurone en commençant par la couche de sortie. Soit l'erreur observée $e_j(n)$ pour le neurone de sortie j et la donnée d'entrainement n :

$$e_j(n) = d_j(n) - y_j(n) \tag{1.9}$$

Ou $d_j(n)$ correspond à la sortie désirée du neurone j et $y_j(n)$ à sa sortie observée.

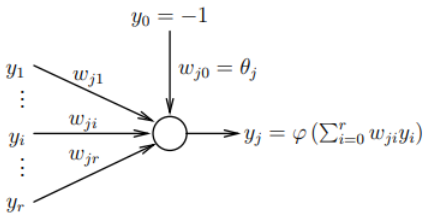


FIGURE 1.6 – Modèle de neurone de sortie.

— La variable n représentera toujours la donnée d'entrainement c'est-à-dire le couple contenant un vecteur d'entrées et un vecteur de sorties désirées.

— L'objectif de l'algorithme est d'adapter les poids des connexions du réseau de manière à minimiser la somme des erreurs sur tous les neurones de sortie.

— L'indice représente toujours le neurone pour lequel on veut adapter les poids.

— Soit $E(n)$ la somme des erreurs quadratiques observées sur l'ensemble C des neurones de sortie :

$$E(n) = \frac{1}{2} \sum_{j \in C} e_j^2(n) \tag{1.10}$$

La sortie $y_j(n)$ du neurone j est définie par :

$$y_j(n) = \varphi[v_j(n)] = \varphi[\sum_{i=0}^{r} w_{ji}(n) y_i(n)] \tag{1.11}$$

Ou $\varphi[.]$ est la fonction d'activation du neurone, $v_j(n)$ est la somme pondérée des entrées du neurone j, $w_{ji}(n)$ est le poids de la connexion entre le neurone i de la couche précédent et le neurone j de la couche courante, et est la sortie du neurone i. On suppose ici que la couche précédant contient r neurones numérotés de 1 à r, que le poids $w_{j0}(n)$ correspond au biais du neurone j et que l'entrée $y_0(n) = -1$. La figure 1.6 illustre l'équation (1.11).

— L'indice i représentera toujours un neurone sur la couche précédente par rapport au neurone j, on suppose par ailleurs que cette couche contient r neurones.

Pour corriger l'erreur observée, il s'agit de modifier le poids $w_{ji}(n)$ dans le sens apposé au gradient $\frac{\partial E(n)}{\partial w_{ji}(n)}$ de l'erreur (la figure 1.7).

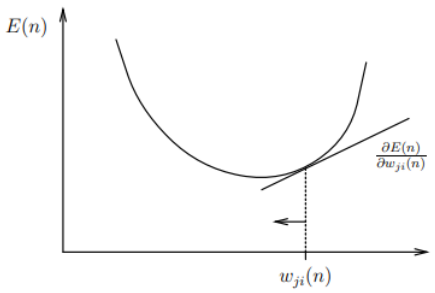


FIGURE 1.7 – Graphe gradient total.

— Cette dérivée partielle représente un facteur de sensibilité si je change un peu $w_{ji}(n)$, est-ce-que ca change beaucoup $E(n)$? si oui, alors je vais changer beaucoup $w_{ji}(n)$ dans le sens inverse de cette dérivée car cela devrait me rapprocher beaucoup du minimum local. Sinon, je dois changer seulement un peu $w_{ji}(n)$ pour corriger l'erreur car je suis tout près de ce minimum !

— Puisqu'il y a r neurones sur la couche précédant la couche de sortie, il y a aussi r poids à adapter et il importe donc de remarquer que la courbe da la figure 1.7 correspond en fait à une hyper-surface de $r+1$ dimensions !

— Par la règle de chaînage des dérivées partielle, qui nous dit que $\frac{\partial f(y)}{\partial x} = \frac{\partial f(y)}{\partial y}.\frac{\partial y}{\partial x}$, on obtient :

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial e_j(n)}.\frac{\partial e_j(n)}{\partial y_j(n)}.\frac{\partial y_j(n)}{\partial v_j(n)}.\frac{\partial v_j(n)}{\partial w_{ji}(n)} \tag{1.12}$$

Et on exprime la variation de poids $\Delta w_{ji}(n)$ sous la forme suivante :

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}(n)} \tag{1.13}$$

Avec $0 \leq \eta \leq 1$ représentant un taux d'apprentissage ou gain de l'algorithme.
Après avoir calculé chacun des termes du gradient de $\frac{\partial E(n)}{\partial w_{ji}(n)}$:
Obtenons :

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -e_j(n)y_j(n)[1-y_j(n)]y_i(n) \tag{1.14}$$

Et la règle dite du «delta » pour la couche de sortie s'exprime par :

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}(n)} = \eta \delta_j(n) y_i(n) \tag{1.15}$$

Avec :

$$\delta_j(n) = e_j(n)y_j(n)[1-y_j(n)] \tag{1.16}$$

Qui correspond à ce qu'on appelle le « gradient local ».
Jusqu'ici, nous avons traité seulement le cas de la couche de sortie ! Il reste maintenant à faire l'adaptation des poids sur les couches cachées. . .

Mais le problème est qu'on ne dispose plus de l'erreur observée ! ?

1.6.2 Cas d'une couche cachée

Considérons maintenant le cas des neurones sur la dernière couche cachée (le cas des autres couches cachées est semblable).

- La variable n désignera toujours la donnée d'entrainement c'est-à-dire un couple de vecteurs d'entrées et de sorties désirées.
- L'objectif sera toujours d'adapter les poids de la couche courante en minimisant la somme des erreurs sur les neurones de la couche de sortie.
- Les indices i et j désigneront respectivement (comme précédemment) un neurone sur la couche précédente et un neurone sur la couche courante.
- L'indice k servira maintenant à désigner un neurone sur la couche suivante.

Reprenons l'expression de la dérivée partielle de l'erreur totale $E(n)$ par rapport à w_{ji} mais en ne dérivant plus par rapport à l'erreur $e_j(n)$ car celle-ci est maintenant inconnue :

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial y_j(n)} . \frac{\partial y_j(n)}{\partial v_j(n)} . \frac{\partial v_j(n)}{\partial w_{ji}(n)} \tag{1.17}$$

Par rapport aux résultats obtenus pour la couche de sortie, les deux derniers termes de cette équation restent inchangés, seul le premier terme requiert d'être évalué :

$$\frac{\partial E(n)}{\partial y_j(n)} = \frac{\partial [\frac{1}{2} \sum_{k \in C} e_k^2(n)]}{\partial y_j(n)} = \sum_{k \in C} [e_k(n) . (-y_k(n)[1 - y_k(n)]) . w_{kj}] \tag{1.18}$$

Et en substituant l'équation (1.16) on obtient :

$$\frac{\partial E(n)}{\partial y_j(n)} = - \sum_{k \in C} \delta_k(n) w_{kj}(n) \tag{1.19}$$

En substituant l'équation (1.19) dans l'équation (1.17) :

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -y_j(n)[1 - y_j(n)][\sum_{k \in C} \delta_k(n) w_{kj}(n)] y_i(n) \tag{1.20}$$

Et :

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}(n)} = \eta \delta_j(n) y_i(n) \tag{1.21}$$

Avec :

$$\delta_j(n) = y_j(n)[1 - y_j(n)] \sum_{k \in C} \delta_k(n) w_{kj}(n) \tag{1.22}$$

On peut démontrer que les équations (1.21) et (1.22) sont valides pour toutes les couches cachées. [8]

Algorithmen de la rétropropagation :

1. Initialiser les poids W, avec des valeurs aléatoires entre [-1, 1].
2. Mélanger les exemples.
3. Pour tout exemple x faire :
   (a) Calculer les sorties de réseaux en propageant les entrées vers les sorties.
   (b) Corriger les poids en rétropropageant l'erreur :

$$w_{ji}(n) = w_{ji}(n-1) + \Delta w_{ji}(n) = w_{ji}(n-1) + \eta \delta_j(n) y_i(n) \tag{1.23}$$

   Ou le « gradient local » est défini par :

$$\delta_j(n) = \begin{cases} e_j(n) y_j(n)[1 - y_j(n)] & \text{Si } j \in \text{couche de sortie} \\ y_j(n)[1 - y_j(n)] \sum_k \delta_k(n) w_{kj}(n) & \text{Si } j \in \text{couche cachée} \end{cases} \tag{1.24}$$

4. Répéter les étapes 2 et 3 jusqu'à remplir le critère d'arrêt.

1.7 Conclusion

Dans ce chapitre, nous avons parlé du réseau de neurones artificiels et expliqué ses mécanismes, et nous avons parlé du premier modèle et de son développement, mais il reste un modèle approximatif pour les neurones biologiques.
Bien que le perceptron multicouche ne soit pas toujours efficace, il peut résoudre de nombreux problèmes avec un taux de réussite élevé. Nous avons également parlé des principaux types de fonctions d'activation, et expliqué les étapes de l'algorithme de propagation et rétropropagation du gradient et comment pour corriger les poids synaptiques, lors de la phase d'apprentissage du réseau de neurones.
Le prochain chapitre sera consacré à un mécanisme moderne qui peut être la meilleure approximation d'un neurone biologique en général, sur le cerveau humain et les types de neurones impulsionnels.

CHAPITRE 2

RÉSEAUX DE NEURONES IMPULSIONNELS.

2.1 Introduction

Les réseaux de neurones impulsionnels (SNN) sont une tentative de comprendre et d'imiter les fonctionnalités du cerveau humain - un défi clé de l'informatique de nouvelle génération.
Les réseaux de neurones impulsionnels (SNN) et leurs algorithmes d'apprentissage profond ont été inspirés par la structure, l'organisation et les nombreux aspects de l'apprentissage profond et de la représentation des connaissances profondes dans le cerveau humain.
Ce chapitre présente des informations de base sur le cerveau et révèle certains processus internes de l'apprentissage profond et de la représentation des connaissances profondes, qui serviront de source d'inspiration pour les réseaux de neurones impulsionnels (SNN) et l'algorithme de classification bio-inspiré (BI-AI) présentés dans les chapitres suivants.
Les informations présentées ici ne sont pas destinées à modéliser le cerveau dans sa complexité structurelle et fonctionnelle précise, mais plutôt à : emprunter les principes de traitement de l'information de ce dernier pour la création de SNN bio-inspirés. [9]
Nous allons également aborder certains des types de neurones impulsionnels et les équations mathématiques qu'ils représentent ainsi que les techniques d'apprentissage en relation telles qu'elles sont à l'œuvre (au regard des connaissances actuelles) dans le cortex.

2.2 Le cerveau humain

Le cerveau est l'organe le plus complexe du corps humain. Il est au centre d'un réseau de communication vaste et complexe qui recherche et collecte constamment des informations du reste du corps et du monde extérieur. Il est à l'origine de toutes nos pensées, actions, souvenirs, sensations et expériences du monde. Cette masse de tissu gélatineuse, qui pèse environ 1,4 kilogramme, contient une centaine de milliards de cellules nerveuses, ou neurones.
La complexité de la connectivité entre ces cellules nous étonne encore aujourd'hui. Chaque neurone peut entrer en contact avec des milliers, voire des dizaines de milliers d'autres, via

de minuscules structures appelées synapses. Notre cerveau forme un million de nouvelles connexions à chaque seconde de notre vie et pour chaque instant que nous vivons. La structure et la force des connexions changent constamment, le cerveau est extrêmement sensible à son environnement et il n'existe pas deux cerveaux identiques.
C'est dans ces connexions changeantes que les souvenirs sont stockés, les habitudes acquises et l'acquisition de connaissances, en renforçant certains modèles d'activité cérébrale, et en perdant d'autres, c'est ce qui a façonné nos personnalités et fait de chacun de nous son propre caractère unique.
Nous allons d'abord aborder les questions suivantes.
Premièrement, quel est le rôle du cerveau ?, de quoi est-il constitué ?, quels sont les principes de la mémoire et ceux de l'apprentissage profond ?.

2.2.1 Rôle du cerveau

La tâche principale du cerveau est d'aider à maintenir l'ensemble du corps dans un état optimal par rapport à l'environnement, afin de maximiser les chances de survie. Pour ce faire, le cerveau enregistre les stimuli et y répond en générant des actions.

2.2.2 Comment le cerveau agit-il ?

Le cerveau est modulaire : ses différentes parties font différentes choses. Les modules sont toutefois étroitement interconnectés et aucun d'entre eux ne fonctionne sans le soutien de nombreux autres (et du reste du corps). En général, les fonctions de bas niveau, comme l'enregistrement des sensations, sont fortement localisées, mais les fonctions de plus haut niveau, comme la mémoire et le langage, résultent des interconnexions entre les zones du cerveau. [10] Personne ne sait exactement comment l'activité électrique se transforme en expérience. Il s'agit là d'un problème difficile, qui n'a pas encore été résolu. Cependant, on en sait beaucoup sur les processus cérébraux qui transforment les informations entrantes en diverses composantes de l'expérience subjective, comme les pensées ou les émotions. [10]

Le cerveau est constitué d'environ 100 milliards de cellules. Environ 10% d'entre elles sont des cellules électriques spécialisées appelées neurones, qui envoient des signaux les unes aux autres, cette transmission de signaux différencie le fonctionnement du cerveau de tout autre processus corporel. Bien que les signaux soient électriques, le mode de transmission entre les cellules est chimique - les signaux sont transmis par des substances appelées neurotransmetteurs. [10]

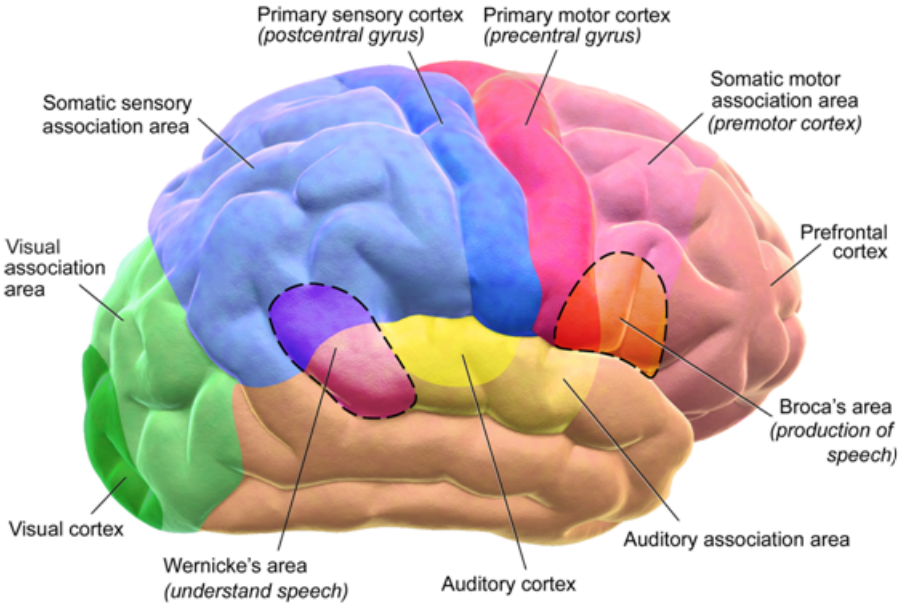


FIGURE 2.1 – Les sections du cerveau humain et leurs fonctions.

2.2.3 Les principes de la mémoire

La mémoire est un terme général utilisé pour désigner un certain nombre de fonctions cérébrales différentes. La caractéristique commune de ces fonctions est la recréation d'expériences passées par la mise à feu synchrone des neurones qui ont été impliqués dans l'expérience originale.

2.2.3.1 Qu'est-ce que la mémoire ?

Un souvenir peut être la capacité de se rappeler d'un poème ou de reconnaître un visage sur demande, une vague vision d'un événement passé depuis longtemps, l'habileté requise pour faire du vélo, ou la connaissance de l'environnement, ou le fait de savoir que vos clés de voiture sont sur la table. Tous ces phénomènes ont en commun d'impliquer l'apprentissage et la reconstruction totale ou partielle d'une expérience passée.
L'apprentissage est un processus au cours duquel les neurones qui s'activent ensemble pour produire une expérience particulière sont modifiés de sorte qu'ils ont tendance à s'activer à nouveau. une tendance à se déclencher ensemble à nouveau. L'action combinée des neurones qui s'ensuit reconstitue l'expérience initiale, ce qui en produit un "souvenir". Le fait de se souvenir rend les neurones impliqués encore plus susceptibles de se déclencher à nouveau à l'avenir, de sorte que la reconstruction répétée d'un événement le rend de plus en plus facile à se rappeler. [10]

2.2.4 Les principes de l'apprentissage profond et de la connaissance profonde

Le cerveau est une machine complexe et intégrée de traitement de l'information spatio-temporelle. Un cerveau humain comporte une série de zones structurelles et fonctionnelles

qui sont réparties dans un espace 3D contraint. Lorsque le cerveau traite des informations, qu'elles soient déclenchées par des stimuli externes ou par des processus internes, des voies spatio-temporelles complexes sont activées et des modèles sont formés dans tout ou partie du cerveau. [9]

2.2.4.1 Qu'est-ce que l'apprentissage profond et qu'est-ce que la connaissance profonde ?

L'apprentissage profond dans le cerveau consiste à établir de nouvelles connexions entre des groupes de neurones dans différentes parties du cerveau. Cela permet de construire le cerveau et de le rendre plus performant. [10]
Le tissu cérébral peut être "renforcé" et se construire comme un muscle, en fonction de la quantité d'exercice qu'il subit. Ainsi, si une personne apprend et pratique une compétence, comme jouer d'un instrument de musique ou faire des mathématiques, la partie du cerveau concernée par cette tâche devient physiquement plus grande. Elle devient également plus efficace et permet à la personne d'effectuer la tâche avec plus d'habileté et c'est ce que cela représente la connaissance profonde. [10] En d'autres termes, plus vous créez de connexions lors de l'apprentissage et plus elles sont fortes, mieux vous pouvez utiliser ce que vous apprenez et plus il vous faut de temps pour l'oublier. [10]

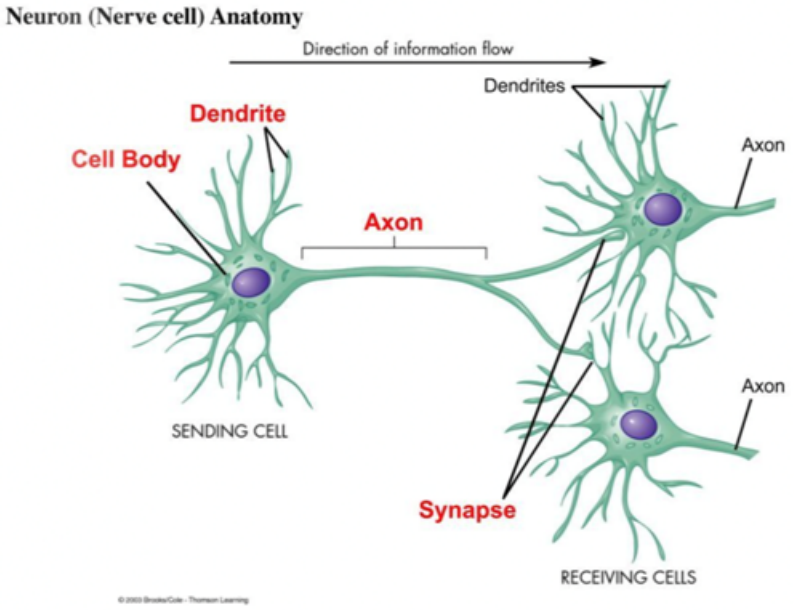


FIGURE 2.2 – Connexion des neurones impulsionnels par des synapses.

2.3 Réseaux de neurones impulsionnels (SNN)

Les neurones représentent les unités de traitement élémentaires du cerveau. Les neurones sont connectés les uns aux autres, par des synapses, selon un schéma complexe formant des structures spécifiques.
Les réseaux de neurones impulsionnels (SNN) sont des réseaux de neurones artificiels qui

imitent de plus près les réseaux de neurones naturels [53]. En plus de l'état neuronal et synaptique, les SNN intègrent le concept de temps et de lieu dans leur modèle de fonctionnement. [9]
Les considérations importantes dans la conception d'un algorithme d'apprentissage pour les SNN sont les suivantes : les modèles de neurones, la communication par les synapses, la topologie du réseau.

2.3.1 Modèles des neurones impulsionnels :

Ici nous présentons les modèles de neurones impulsionnels les plus répandus dans la littérature.

2.3.1.1 Modèle Hodgkin-Huxley (HHM)

Le travail de Hodgkin et Huxley (1952) sur la conduction nerveuse a longtemps été reconnu comme une réalisation scientifique exceptionnelle. Ils ont mené l'expérience sur l'axone géant d'un calmar. Ils ont utilisé des méthodes de voltage-clamp pour obtenir des résultats expérimentaux quantitatifs étendus [11], et à partir de l'expérience, ils ont conclu qu'il y a trois canaux ioniques dans le neurone, qui sont le sodium (Na), le potassium (K) et le canal de fuite (L) avec résistance.
Pour calculer le total du courant ionique Iion, qui est la somme de tous les canaux participants, on utilise les formules des équations (2.1) et (2.2). Dans l'équation (2.1), G_k représente tous les canaux impliqués, E_k représente le potentiel d'équilibre et Vm est le potentiel de la membrane. De plus, comme élaboré par [12] [13] trois portes de type "m" et une porte de type "h" sont utilisées pour contrôler le canal sodique, et quatre portes de type "n" pour contrôler le canal potassique. Ces variables de déclenchement sont calculées à l'aide des équations (2.3), (2.4) et (2.5) où le taux de transition pour chaque porte de l'état non permissif à l'état permissif est représenté par $\alpha_m(V)$, $\alpha_h(V)$ et $\alpha_n(V)$ et le taux de transition pour chaque porte de l'état permissif à l'état non permissif est représenté par $\beta_m(V)$, $\beta_h(V)$ et $\beta_n(V)$. [9]

$$I_{ion} = \sum_k I_k = \sum_k G_k(V_m - E_k) \tag{2.1}$$

$$I_{ion} = G_{Na} m^3 h(V_m - E_{Na}) + G_K n^4 (V_m - E_K) + G_L(V_m - E_L) \tag{2.2}$$

$$\frac{m}{dt} = \alpha_m(V)(1 - m) - \beta_m(V)m \tag{2.3}$$

$$\frac{h}{dt} = \alpha_h(V)(1 - h) - \beta_h(V)h \tag{2.4}$$

$$\frac{n}{dt} = \alpha_n(V)(1 - n) - \beta_n(V)n \tag{2.5}$$

Ce modèle de neurone ne décrit que les canaux et le flux d'ions dans le neurone lors de la génération d'impulsion, ce qui est loin du neurone biologique complexe et comporte donc plusieurs faiblesses [14] [15] [9], mais il est aussi remarquablement réussi, il est devenu un point fondamental et de départ pour le développement de nombreux autres modèles

de neurones simplifiés.

2.3.1.2 Modèle Leaky Integrate-and-Fire (LIFM)

Par rapport au modèle HHM qui traite des canaux ioniques et des flux ioniques, le modèle LIF (Leaky Integrate and Fire) considère le neurone comme un intégrateur fuyant, qui émet une impulsion si la tension d'entrée atteint un seuil, puis revient à un état de repos. [9]
représente le neurone comme une combinaison parallèle d'une résistance "fuyante" (conductance, g L) et d'un condensateur (C). Une source de courant I(t) est utilisée comme entrée de courant synaptique pour charger le condensateur et produire un potentiel V(t).
Lorsque le potentiel dépasse le seuil ($V(t) \geq V_{th}$), le condensateur se décharge jusqu'à un potentiel de repos E L en utilisant le commutateur commandé par la tension, comme un neurone biologique. Ainsi, le modèle LIF est régi par l'équation différentielle suivante :

$$C\frac{dv}{dt} = -g_L(V(t) - E_L + I(t)) \tag{2.6}$$

2.3.1.3 Pinsky et Rinzel

Le modèle neuronal utilisé est directement inspiré des travaux de Traub (Traub et al., 1991) sur la modélisation précise des neurones pyramidaux . Puisque le neurone est de nature tubulaire, Traub propose de le décomposer en 19 compartiments distincts : un pour le soma et 18 pour la dendrite. La dynamique électrique de chaque compartiment est guidée par le modèle résistif de Hodgkin et Huxley (1952) qui généralement la plus proche d'un neuron bioloique. Le modèle de Traub reproduit fidèlement l'électrophysiologie mais possède une grande complexité calculatoire qui le rend inutilisable lorsque les neurones sont connectés en grand nombre dans un réseau. Pour cela, Pinsky et Rinzel (1994) proposent un modèle qui réduit ces 18 compartiments à seulement 2 principaux : le premier pour le soma et le second pour la dendrite. Ce modèle appelé "modèle bi-compartimental" propose des propriétés électrophysiologiques très proches de celles du modèle de Traub tout en divisant par 9 sa complexité générale. C'est pour ces avantages aussi bien sur le tableau de la plausibilité que de la complexité que ce modèle a été choisi pour implémenter la dynamique interne d'une cellule de lieu. Les variations temporelles des potentiels du soma et de la dendrite sont exprimées par le système d'équations différentielles couplé suivant :

$$C_m dV_{s,i}/dt = -I_{Leak}(V_{s,i}) - I_{Na} - I_{k-DR} + (g_{c,i}/p)(V_{d,i} - V_{s,i}) + I_{ext}/p \tag{2.7}$$

$$C_m dV_{d,i}/dt = -I_{Leak}(V_{d,i}) - I_{Ca} - I_{K-AHP} - I_{K-C} - I_{syn}/(1-p) + I_d/(1-p) + (g_{c,i}/(1-p))(V_{s,i} - V_{d,i}) \tag{2.8}$$

2.3.1.4 Modèle FitzHugh-Nagumo (FHN)

Ce système, suggéré par FitzHugh (1961), est une simplification bidimensionnelle du modèle HHM de la génération d'impulsion dans les axones géants de calmar.

$$V = V - V^3/3 - W + I \tag{2.9}$$

$$W = 0.08(V + 0.7 - 0.8W) \tag{2.10}$$

— V est le potentiel de membrane.

— W est une variable de récupération.

— I est la magnitude du courant de stimulation.

La motivation du modèle FHN était d'isoler conceptuellement les propriétés essentiellement mathématiques de l'excitation et de la propagation des propriétés électrochimiques du flux d'ions sodium et potassium. Le modèle consiste en une variable de type tension ayant une non-linéarité cubique qui permet une auto-excitation régénérative via une rétroaction positive, et une variable de récupération ayant une dynamique linéaire qui fournit une rétroaction négative plus lente.

2.3.1.5 Modèle d'Izhikevich (IM)

Dans le modèle IM [24], un simple neurone impulsionnel est formulé en combinant la plausibilité biologique du modèle HHM et l'efficacité de calcul des neurones LIF. Le modèle est défini comme dans l'équation (2.11) où v est la tension de la membrane, u est une variable de récupération utilisée pour ajuster v, I(t) est les courants d'entrée, et a et b sont des paramètres ajustables.

$$\frac{dv}{dt}(t) = 0.04v^2 + 5v + 140 - u + I(t) \tag{2.11}$$

$$\frac{du}{dt}(t) = a(bv - u) \tag{2.12}$$

Une valeur de seuil est fixée à 30 mV et si la tension v est supérieure à ce seuil, v et u sont réinitialisés.

$$\text{if } v \geq 30mV \text{ , then} \begin{cases} v = c \\ u = u + d \end{cases} \tag{2.13}$$

En fonction des valeurs des paramètres de l'IM, un neurone peut manifester différents comportements d'impulsionnels, comme le montre dans la figure 2.3 .

— Le paramètre a décrit l'échelle de temps de la variable de récupération u.

— Le paramètre b décrit la sensibilité de la variable de récupération u aux fluctuations sous-seuil du potentiel de membrane v.

— Le paramètre c décrit la valeur de réinitialisation du potentiel de membrane v après les impulsions.

— Le paramètre d décrit la remise à zéro de la variable de récupération u après une impulsion. [9]

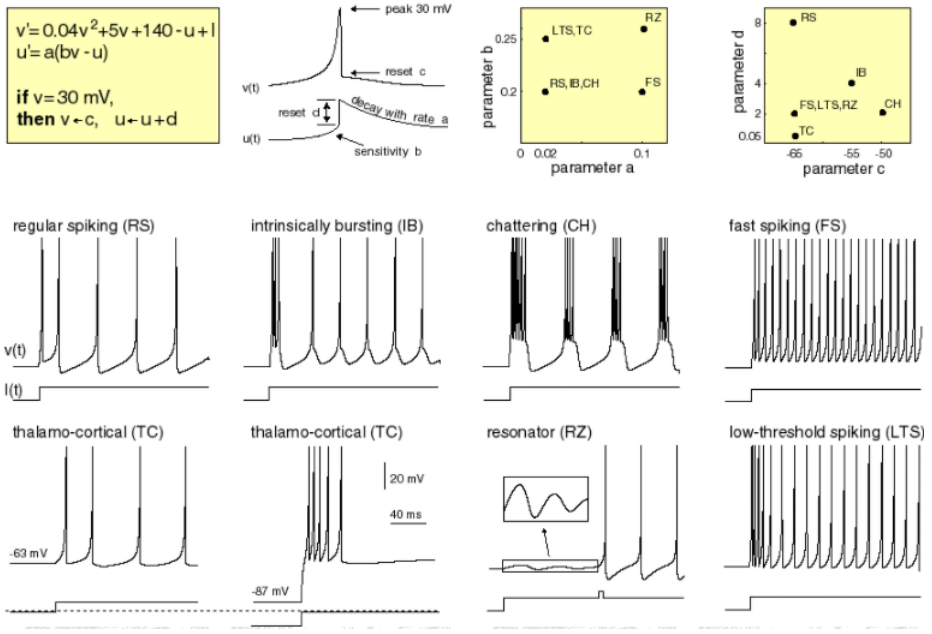


FIGURE 2.3 – Les différents types de neurones izhikevich qui correspondent à différentes valeurs des paramètres.

2.4 Règles d'apprentissage

Comme c'est le cas dans les réseaux de neurones biologiques, le timing précis des spikes est l'un des facteurs les plus importants dans le codage des données et le calcul des SNN [25] [26] afin de générer un traitement efficace de l'information dans le système. L'information est représentée et codée dans les spikes qui sont très dépendants du timing exact des tirs, ainsi l'apprentissage dans les SNN est un processus très complexe. En général, l'apprentissage est défini comme le processus d'adaptation des paramètres et la règle d'apprentissage est définie comme la procédure d'ajustement des poids de connexion [9]. L'apprentissage dans les SNN est lié aux changements des poids de connexion entre deux neurones impulsionnels. Plusieurs algorithmes et équations ont été proposés jusqu'à présent, dont certains sont présentés dans ce chapitre :

2.5 Hebbian

La théorie hebbienne est une théorie neuroscientifique affirmant qu'une augmentation de l'efficacité synaptique résulte de la stimulation répétée et persistante d'une cellule présynaptique sur une cellule postsynaptique.
" Lorsqu'un axone d'une cellule A est suffisamment proche pour exciter une cellule B et qu'il participe de manière répétée ou persistante à son déclenchement, un processus de croissance ou un changement métabolique se produit dans l'une des cellules ou dans les deux, de sorte que l'efficacité de A en tant que cellule déclenchant B est accrue ", (Hebb 1949).
Cependant, Hebb soulignait que la cellule A devait "participer à la mise à feu" de la cellule B, et qu'une telle causalité ne pouvait se produire que si la cellule A se mettait à feu juste avant, et non en même temps que la cellule B. Cet aspect de la causalité dans

les travaux de Hebb préfigurait ce que nous savons aujourd'hui de la plasticité dépendant de la synchronisation des pointes, qui nécessite une préséance temporelle.
En général, La règle d'apprentissage de Hebb, est une règle d'apprentissage qui spécifie de combien le poids de la connexion entre deux unités doit être augmenté ou diminué proportionnellement au produit de leur activation, qui stipule que les connexions entre les neurones peuvent être renforcées si les neurones tirent simultanément.

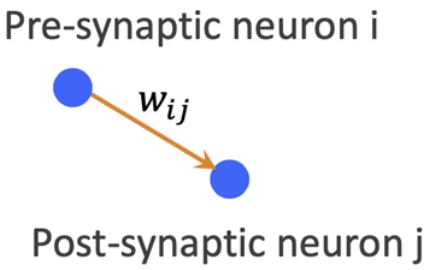


FIGURE 2.4 – Poids w_{ij} entre un neurone présynaptique i et un neurone postsynaptique j.

Mathématiquement :

$$\Delta W_{ij} = \eta * x_i * x_j \tag{2.14}$$

Où :

- W_{ij} le poids de la connexion du neurone i au neurone j.
- x_i , x_j les niveaux d'excitation binaires du neurone i et j.
- η le taux d'apprentissage

Les règles d'apprentissage basées sur la corrélation sont maintenant généralement appelées apprentissage Hebbien. est souvent considéré comme la base neuronale de l'apprentissage non supervisé.

2.6 Spike-Propagation

Depuis les travaux de Rashevsky et d'autres au début des années 60, la valeur réelle de la sortie d'un neurone est supposée correspondre à sa fréquence de tir moyenne. Cependant, on s'intéresse de plus en plus à la possibilité d'un codage de l'information dans le rythme des potentiels d'action individuels (pics). Pour les réseaux de neurones impulsionnels avec de multiples synapses retardées, Natschl ager et Ruf ont décrit un puissant algorithme d'apprentissage non supervisé basé sur une version temporelle de l'apprentissage de Hebbian. Pour étudier la puissance de calcul de tels réseaux sans les contraintes associées à l'apprentissage de Hebbian, un algorithme de rétropropagation d'erreur (BP) a été dérivé pour les réseaux de neurones impulsionnels, analogue au travail de Rumelhart et al. Pour tenir compte de la nature discontinue des neurones à pointes, la fonction de seuillage a été approximée, ce qui est validé pour de petits taux d'apprentissage. L'algorithme est capable d'apprendre des tâches non linéaires complexes dans les réseaux de

neurones à impulsions de la même manière que les réseaux de neurones traditionnels.
SpikeProp [32] est conçu pour déterminer un ensemble de temps de tir souhaités (t_j^d) de tous les neurones de sortie, au niveau des neurones post-synaptiques pour un ensemble donné de modèles d'entrée. Ceci est réalisé en appliquant une fonction d'erreur E, en particulier l'erreur des moindres carrés moyens pour minimiser l'erreur de la différence au carré entre les temps de sortie d'entraînement t_j et les temps de sortie souhaités t_j^d. Néanmoins, deux hypothèses sont mentionnées : chaque neurone ne peut tirer qu'une fois à chaque étape de traitement et l'évolution temporelle du potentiel de membrane du neurone après le tir est ignorée. Le poids w_{ij}^k reliant le neurone pré-synaptique et le neurone post-synaptique est déterminé pour minimiser l'erreur (Equation (2.15)) dans laquelle g est le taux d'apprentissage. [9] [33]
Pour la fonction d'erreur :

$$E = \frac{1}{2}\sum_j (t_j - t_j^d)^2 \tag{2.15}$$

Pour error-backpropagation :

$$\Delta w_{ij}^k = -\eta \frac{\partial E}{\partial w_{ij}^k} \tag{2.16}$$

2.7 Spike-Timing-Dependent Plasticity

2.7.1 LTP

Potentialisation à long terme ou " long term potentiation" en anglais (LTP), est un processus par lequel les connexions entre les neurones deviennent plus fortes avec une activation fréquente. On pense que la LTP est un moyen par lequel le cerveau change en réponse à l'expérience, et peut donc être un mécanisme sous-jacent à l'apprentissage et à la mémoire. Un seul type de LTP sera discuté ici : la LTP dépendante des récepteurs NMDA. Le mécanisme sous-jacent à la LTP du récepteur NMDA est décrit ainsi que le renforcement des connexions synaptiques là où elle se produit.
Il existe un certain nombre de façons dont LTP peut se produire (figure 2.5). Le mécanisme le plus connu fait intervenir un récepteur du glutamate connu sous le nom de récepteur NMDA. Dans la LTP dépendante du récepteur NMDA, la libération de glutamate active d'abord un sous-type de récepteur de glutamate connu sous le nom de récepteur AMPA. Les récepteurs NMDA se trouvent à proximité de ces récepteurs AMPA, mais ne sont pas activés par de faibles niveaux de libération de glutamate car le canal ionique d'un récepteur NMDA est bloqué par un ion magnésium. Si des potentiels d'action fréquents provoquent une plus grande stimulation des récepteurs AMPA, cela entraînera la dépolarisation du neurone postsynaptique.
Maintenant, la cellule postsynaptique est plus sensible au glutamate car elle a plus de récepteurs pour y répondre. De plus, on pense qu'il y a des signaux qui reviennent à travers la synapse pour stimuler des niveaux plus élevés de libération de glutamate. Tout cela rend la synapse plus forte et plus susceptible d'être activée à l'avenir.
Ce processus est également associé à des modifications de la transcription des gènes dans le neurone, qui peuvent conduire à la production de nouveaux récepteurs ou à des mo-

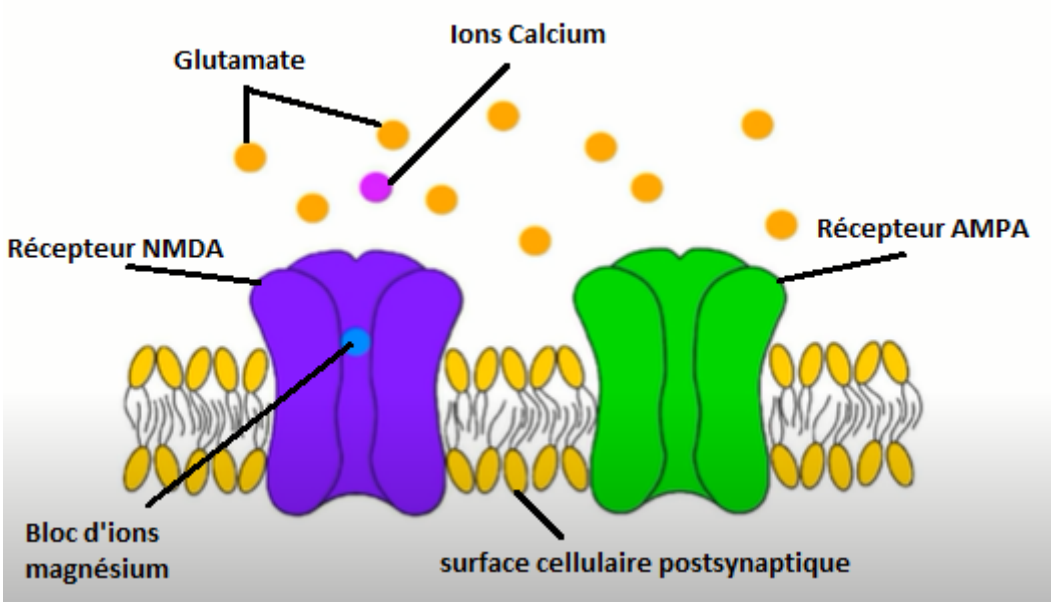


FIGURE 2.5 – La membrane du neurone.

difications de la structure de la cellule. Ces changements semblent être importants pour pérenniser la réactivité accrue de la LTP. [34]

2.7.2 LTD

La dépression à long terme, ou " long term depression" en anglais (LTD), est un processus par lequel les connexions synaptiques entre les neurones s'affaiblissent. C'est le processus opposé à la potentialisation à long terme. Bien que les fonctions du LTD ne soient pas complètement comprises, on pense qu'il est important pour la formation de la mémoire, peut-être en réinitialisant les changements synaptiques précédents pour permettre la formation de nouveaux souvenirs via une potentialisation à long terme.
Il existe plusieurs mécanismes différents par lesquels la LTD peut se produire, mais le mieux compris d'entre eux implique les mêmes récepteurs du glutamate impliqués dans la potentialisation à long terme : les récepteurs NMDA et AMPA. Les récepteurs NMDA sont généralement bloqués par un ion magnésium, qui n'est éliminé que si le neurone post-synaptique devient suffisamment dépolarisé comme cela peut se produire par l'activation du récepteur AMPA, lorsque le bloc est retiré, le calcium peut s'écouler dans le neurone, provoquant une dépolarisation supplémentaire. Alors que la potentialisation à long terme se produit généralement après une stimulation brève mais de haute intensité d'un neurone post-synaptique, la LTD peut être causée par une stimulation prolongée de faible intensité ou une stimulation qui se produit après le déclenchement d'un potentiel d'action. Avec le type de stimulation modeste qui entraîne une LTD, il n'y a pas assez de dépolarisation pour provoquer une élimination généralisée du blocage du magnésium du récepteur NMDA. Cependant, il y en a assez pour que certains récepteurs NMDA permettent au calcium de pénétrer dans la cellule. Ce faible niveau de calcium est insuffisant pour activer les enzymes qui facilitent la potentialisation à long terme, mais on pense qu'il active une cascade cellulaire qui provoque l'élimination des récepteurs AMPA. Cela réduit le nombre de récepteurs du glutamate sur le neurone postsynaptique et affaiblit la synapse.

LTD peut également entraîner d'autres changements qui diminuent la force des synapses, comme une diminution de la quantité de glutamate libérée par le neurone présynaptique, et elle peut également impliquer d'autres récepteurs comme les récepteurs métabotropiques du glutamate ou d'autres récepteurs de neurotransmetteurs. [34]

2.7.3 STDP

- Spike-time-dependent plasticity (STDP) est une famille de mécanismes d'apprentissage postulés à l'origine dans le contexte d'algorithmes d'apprentissage automatique artificiel (ou de neurosciences computationnelles), exploitant des calculs basés sur des impulsions (comme dans le cerveau) en mettant l'accent sur les timings relatifs de ces impulsions. Gerstner a commencé à rapporter les premiers algorithmes d'apprentissage dépendant du temps de pointe [[35], [36]] en 1993. STDP s'est avéré meilleur que la plasticité basée sur la corrélation de Hebbian pour expliquer les phénomènes corticaux.[[37], [38]] Étonnamment, des preuves expérimentales de STDP ont été rapportées par des groupes de neurosciences au cours de la dernière décennie [[41]- [49]], nous pouvons donc aujourd'hui affirmer que l'existence physiologique de STDP a été raisonnablement bien établie. Cependant, les implications complètes des principes moléculaires et électrochimiques derrière le STDP sont encore en débat.
Avant de décrire mathématiquement STDP, expliquons d'abord comment les neurones échangent des informations et quelles sont les connexions synaptiques.
La connexion entre un neurone pré-synaptique et un neurone post-synaptique est appelée synapse. Grâce à la propriété de plasticité (LTP et LTD), les synapses peuvent changer de forme ou de fonction sur des périodes de quelques secondes, minutes ou peut-être même toute la vie. Les connexions synaptiques peuvent être modifiées dans la structure ou la fonction lorsque les systèmes nerveux biologiques essaient d'apprendre et de mémoriser. Le changement de force synaptique dépend du moment précis des pointes pré-synaptiques et post-synaptiques. En d'autres termes, la force synaptique est fonction de la synchronisation des pics entre les neurones pré-synaptiques et post-synaptiques. [52]
Dans la figure 2.6, nous présentons une illustration montrant comment fonctionne l'échange entre les cellules pre et post-synaptiques

La figure 2.6 illustre deux neurones connectés par une synapse. Le neurone pré-synaptique envoie un pic pré-synaptique " $V_{mem-pre(t)}$ " à travers l'un de ses axones jusqu'à la jonction synaptique. Les impulsions neuronales sont des tensions membranaires provenant de l'extérieur de la membrane cellulaire "V_{pre+} " par rapport à l'intérieur "V_{pre-}". Ainsi($V_{mem-pre} = V_{pre+} - V_{pre-}$) et ($V_{mem-pos} = V_{pos+} - V_{pos-}$).
Les "grandes" tensions membranaires lors d'un pic provoquent l'ouverture et la fermeture d'une variété de canaux membranaires moléculaires sélectifs permettant à de nombreuses substances ioniques et moléculaires de s'écouler, ou les empêchant de traverser la membrane. Dans le même temps, les vésicules synaptiques à l'intérieur de la cellule pré-synaptique contenant des « paquets » de neurotransmetteurs fusionnent avec la membrane de telle sorte que ces « paquets » sont libérés dans la fente synaptique (l'espace intercellulaire entre les deux neurones à la jonction synaptique Comme décrit dans la sec-

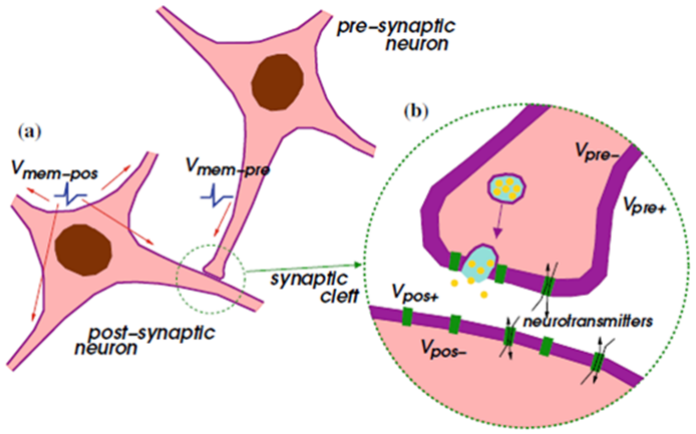


FIGURE 2.6 – L'échange entre les cellules pre et post-synaptiques.

tion B (figure 2.6)).
Les neurotransmetteurs sont collectés en partie par la membrane post-synaptique, contribuant à un changement de sa conductivité membranaire. L'effet cumulatif des pointes pré-synaptiques (provenant de ce neurone ou d'autres neurones pré-synaptiques) finira par déclencher la génération d'une nouvelle pointe au niveau du neurone post-synaptique. Chaque synapse est caractérisée par une "force synaptique" (ou poids) "w" qui détermine l'efficacité d'un pic pré-synaptique à contribuer à cette action cumulative au niveau du neurone post-synaptique. Ce poids "w" pourrait bien être interprété comme la taille et/ou le nombre de paquets de neurotransmetteurs libérés lors d'un pic pré-synaptique. Cependant, pour nos analyses, nous interpréterons plus généralement w comme une sorte de paramètre structurel de la synapse (comme la quantité d'une ou plusieurs substances métaboliques) qui contrôle directement l'efficacité de cette synapse par pointe.
Le poids synaptique w est considéré comme non volatil et de nature analogique, mais il change dans le temps en fonction de l'activité des implusions des neurones pré- et post-synaptiques. Ce phénomène a été initialement observé et rapporté en 1949 par Hebb, qui a introduit son postulat d'apprentissage hebbien vu précedemment.
STDP est un raffinement de cette règle de 1949 qui prend en compte le moment relatif précis des pics pré et post-synaptiques individuels, et non leurs taux moyens au fil du temps. Dans STDP, le changement de poids synaptique w est exprimé en fonction de la différence de temps entre le pic post-synaptique à tpos et le pic pré-synaptique à tpre (voir Fig. 2.7). [52]

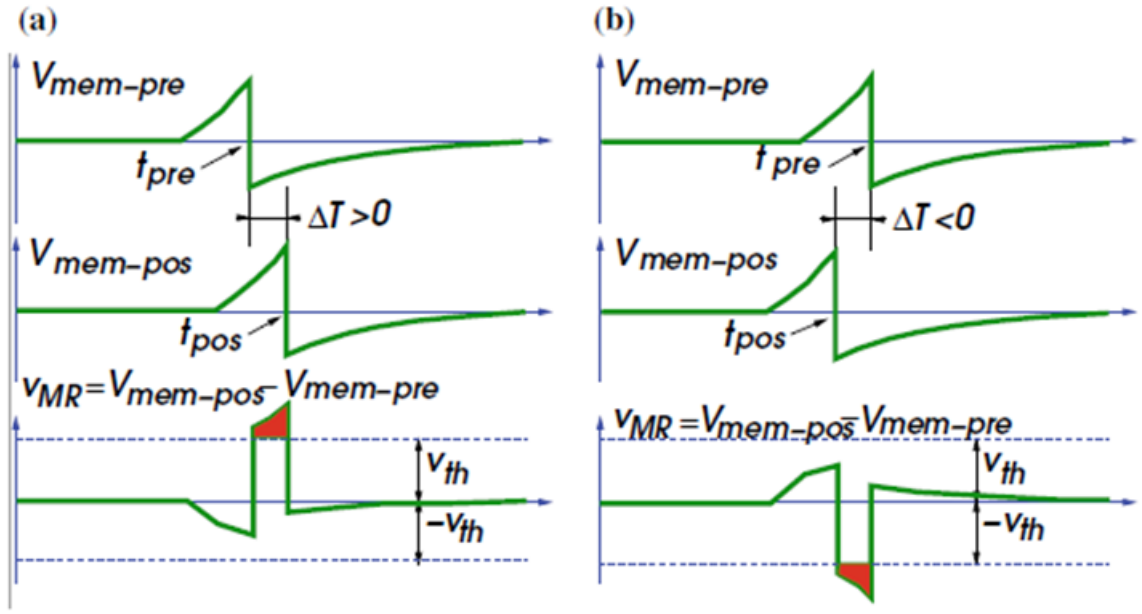


FIGURE 2.7 – La différence de temps entre les pics.

Pour donner un exemple, soient deux neurones « x_i » et « x_j » connectés l'un à l'autre avec un poids spécifique « W_{ij} », le neurone « x_i » a été déclenché signifie qu'il a subi un effet cumulatif de pointes pré-synaptiques, cet effet conduit finalement à la génération d'un nouveau pic dans le neurone post-synaptique, où nous entendons ici le neurone « x_j ». Le poids entre les deux neurons « w_{ij} » change de telle manière que :
Si le neurone « x_i » se déclenche avant le neurone « x_j » donc ($t_i - t_j < 0$ or $\Delta t_{ij} > 0$) la synapse est renforcée (wij augmente) sinon si le neurone « x_i» se déclenche après « x_j » ($t_i > t_j$ ou $\Delta t_{ij} > 0$), la synapse s'affaiblit (w_{ij} diminue).
Soit $F(.)$ la fonction STDP, le poids de la synapse est modifié selon la règle Delta $w_{ij} = w_{ij} + \beta F(\Delta t_{ij})$, β étant une pondération de la fonction STDP, Δt_{ij} la différence entre t_i et t_j et $F(.)$ tels que :

$$F(\Delta t) = \begin{cases} -A - exp^{(-\Delta t/\tau -)} \\ +A + exp^{(\Delta t/\tau +)} \end{cases} \tag{2.17}$$

— $\tau-$ et $\tau+$: déterminent les plages d'intervalles inter-spike pré-postsynaptiques sur lesquels se produisent le renforcement et l'affaiblissement synaptique.

— $-A$,$+A$: qui sont tous les deux positifs, déterminent les quantités maximales de modification synaptique.

2.8 Pourquoi SNN ?

Les réseaux de neurones impulsionnels (SNN) promettent de combler le fossé entre les réseaux de neurones artificiels (ANN) et les réseaux de neurones biologiques (BNN) en exploitant des neurones biologiquement plausibles qui offrent une inférence plus rapide,

une dépense énergétique moindre et des capacités de traitement de l'information en fonction des événements.
Les réseaux neuronaux biologiques présentent certaines caractéristiques qui les rendent supérieurs à quelques égards par rapport aux techniques traditionnelles d'apprentissage automatique, y compris les réseaux neuronaux classiques du chapitre 1 :

— La modélisation prédictive et la prédiction d'événements.

— Traitement rapide et massivement parallèle de l'information .

— structures évolutives (de dizaines à des milliards de neurones impulsionnels).

— faible consommation d'énergie si elle est mise en œuvre sur des plates-formes neuromorphiques.

— Apprentissage profond et représentation des connaissances profondes dans les SNN inspirés par le cerveau.

— Permettre le développement d'une interface homme-machine en utilisant un réseau neuronal inspiré du cerveau.

Méthode/ caractéristiques	**Méthodes statistiques (MLR, kNN, SVM)**	**ANN (par exemple, MLP, CNN)**	**SNN**
Informations	Scalaires	Scalaires	Spike sequences
Représentation des données	Scalaires, vecteurs	Scalaires, vecteurs	Modèles TSTD entiers
Apprentissage	Statistique, limité	Règle hebbienne	Spike-time dependent
Traiter avec TSTD	limité	Modérer	Excellent
Calcul parallèle	limité	Modérer	Massive
Prise en charge matérielle	Standard	VLSI	VLSI neuromorphe

TABLE 2.1 – Une analyse comparative entre SNN et d'autres méthodes d'apprentissage automatique.

2.9 Conclusion

Comme nous l'avons vu dans ce chapitre, il existe un grand nombre de modèles pour simuler les neurones impulsionnels. L'apprentissage dans un réseau de neurones impulsionnels dépend principalement de la modification des poids de communication entre ces neurones impulsionnels afin d'obtenir un temps de déclenchement précis et idéal pour ces cellules. L'un des modèles d'apprentissage les plus en vogues est le modèle STDP qui est, à

l'heure actuelle, le seul qui explique de manière satisfaisante l'apprentissage dans le cortex. Dans le chapitre suivant, nous introduisons un nouvel algorithme de classification basé sur un modèle de neurone impulsionnel de type Izhikevich et un apprentissage basé sur STDP.

CHAPITRE 3

UN CLASSIFIEUR NEURONAL POUR L'ANALYSE DES SENTIMENTS

3.1 Introduction

Bien que nous ayons essayé d'utiliser la structure du cerveau humain et sa façon unique de travailler pour concevoir l'architecture de notre application, ainsi que sa capacité à apprendre en profondeur, il est important de noter que tout cela reste hautement spéculatif et dans les limites de notre connaissance primitive du cerveau humain.
Nous nous sommes appuyés sur un modèle mathématique qui représentera les neurones impulsionnels du type de ceux que nous avons présenté au chapitre 2, ainsi que sur les méthodes d'apprentissage en profondeur que nous avons mentionnées dans ce même chapitre. Nous discuterons de tout cela dans ce qui suit, et nous détaillerons les défis auxquels nous avons été confrontés au cours de travail.

3.2 L'ingénierie du classifieur :

Nous nous sommes grandement inspirés des principes de l'ingénierie du cerveau humain, comme présenté au chapitre 2, pour construire notre classifieur. Le cerveau humain est composé de différentes parties qui remplissent des fonctions différentes, le classifieur a été construit sur la base de ce principe de modularité.
Afin de bâtir ce classifieur, nous nous sommes concentrés sur une tâche de classification bien connue : l'analyse des sentiments dans les textes. Cette tâche nécessite de décider de la classe d'un texte présenté en entrée. Ce texte peut véhiculer des sentiments positifs et sera inclus à la classe "positive", ou au contraire inspirer des sentiments négatifs, auquel cas il sera classé dans la classe "négative".
De par la modularité du classifieur, les deux classes seront apprises dans deux structures distinctes. Par structure, nous entendons un réseau connecté de neurones impulsionnels aussi appelé "connectome". Le premier connectome est dédié à la détection des commentaires positifs qui est aussi appelé la section positive le second connectome est dédié à la détection des commentaires négatifs et il se nommera la section négative ou le réseau de

neurones impulsionnel négatif.
Les deux connectomes sont construits par apprentissage STDP en présentant séquentiellement tous les commentaires positifs (resp. négatifs) à l'entrée des neurones adéquats dans le connectome positif (resp. négatif) A l'interrogation du système sur la nature du texte en entrée, ces deux connectomes seront sollicités pour propager des ondes neuronales induites par l'activation de neurones rendant compte des mots du texte d'entrée.

Le classifieur discrimine les deux classes en estimant lequel des deux connectomes propage mieux l'onde neuronale, ou une autre manière de le dire, réagit avec une réponse plus forte lorsqu'il est exposée à la stimulation.

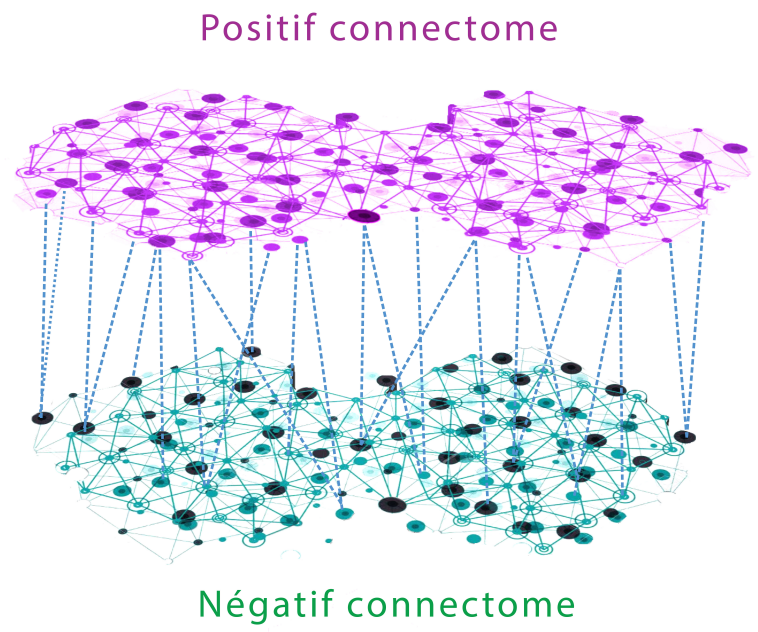


FIGURE 3.1 – Les deux parties principales qui composent le système .

3.3 Construction du réseau (Connectome)

Le connectome est un réseau de neurones impulsionnels, un groupe de neurones impulsionnels connectés les uns aux autres par des synapses. Le nombre de ces neurones pulsants et la puissance de leur connexion diffèrent d'un connectome à l'autre, ce qui

signifie que chaque classe a sa structure spécialisée.
Cette unique structure qui caractérise chaque classe est formée pendant la phase d'apprentissage. Chaque neurone impulsionnel dans le réseau de neurones représente un mot dans la langue d'écriture des commentaires. Cet mots ne diffèrent pas entre le connectome positif et négatif. Les mots positifs dans la langue seront naturellement plus sollicités dans le connectome positif et les mots négatifs dans le second connectome.
En outre, certains neurones impulsionnels représentant des mots positifs auront une connexion plus dense dans le connectome positif, ce même mot pourrait être complètement déconnecté dans le connectome négatif. C'est cette différence de connectivité, qui aura un impact dominant dans la propagation des ondes neuronales, ce qui permet la descrimination des deux classes.

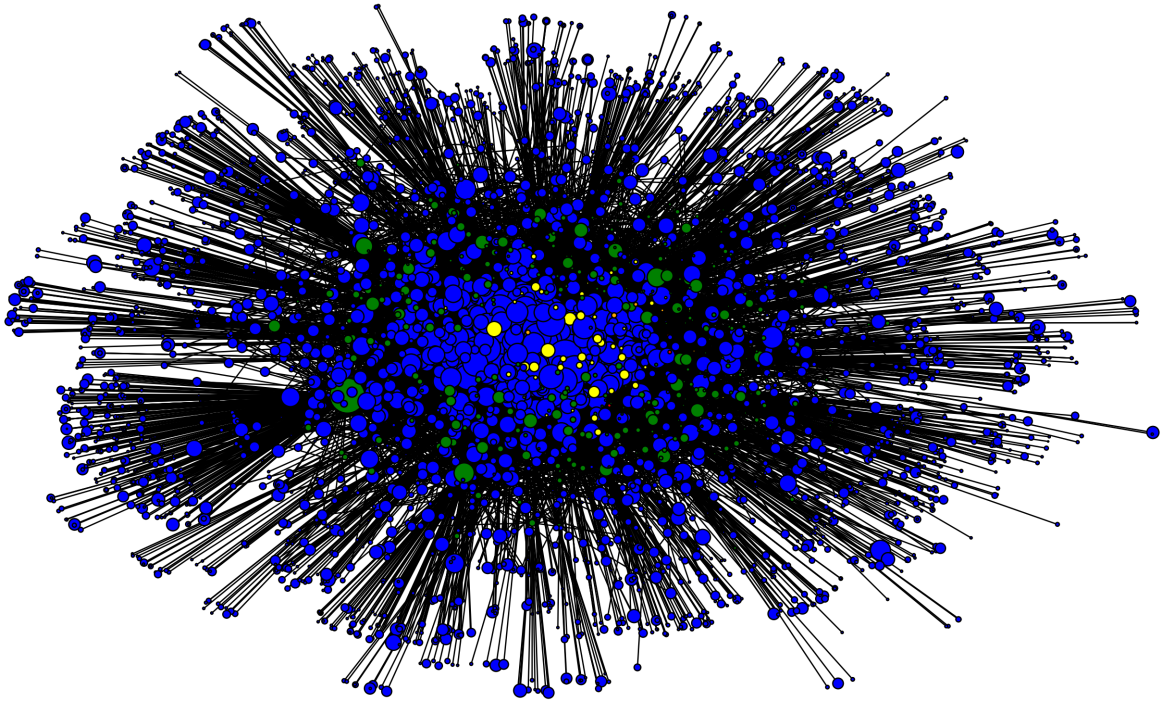


FIGURE 3.2 – La connectivité d'un seul mot dans le connectome avec un profondeur de 4 niveaux.

3.4 Le principe de l'apprentissage

L'apprentissage sera supervisé. À ce stade, le réseau de neurones de chaque connectome est construit à partir de ce qu'il lui est proposé comme base pré-étiquetée pour l'apprentissage. Par exemple, lorsque nous lui proposons un commentaire positif à l'entrée pour qu'il apprenne, le connectome positif est celui qui traitera ce commentaire, car chaque mot du commentaire représentera un neurone impulsionnel dans le réseau en formation, et ces mots dans le commentaire forment de nouveaux liens entre les neurones ou renforce

des liens antérieurs au sein du réseau de neurones impulsionnels de ce connectome. Le même processus est également appliqué dans le connectome négatif, il traite de la même manière avec commentaires négatifs.

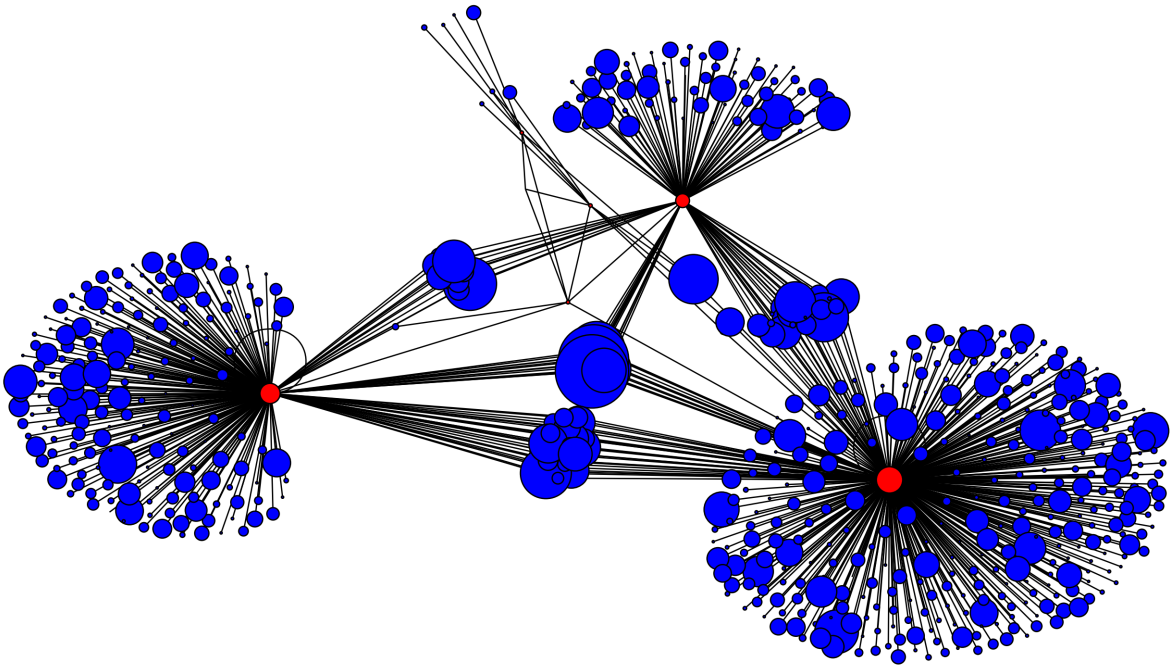


FIGURE 3.3 – Le début de l'apprentissage et la création de synapses entre un groupe de neurones.

3.5 Le principe de stimulation

La stimulation du cortux dépend des organes sensoriels en particulier et du reste du corps en général lors de l'interaction humaine avec son environnement, où des impulsions nerveuses sont envoyées au cerveau après quoi il sera décidé si ce signal sera traité ou ignoré.
Sur la base de ce principe, nous avons adopté la méthode de stimulation pour notre classifieur. Par exemple, lorsque nous proposons un commentaire en entrée, nous donnons une certaine force de courant aux neurones impulsionnels qui représentent les mots de ce commentaire, puis nous observons comment ces neurones affectent le reste des neurones et la façon dont ils interagissent les uns avec les autres.
Les neurones impulsionnels qui représentent les mots du commentaire entré interagissent de manière plus puissante avec le connectome de la classe auquel ils. Ce qui signifie que la réponse la plus active, électriquement parlant, se produira dans le connectome auquel appartient le commentaire. (Figure 3.4)

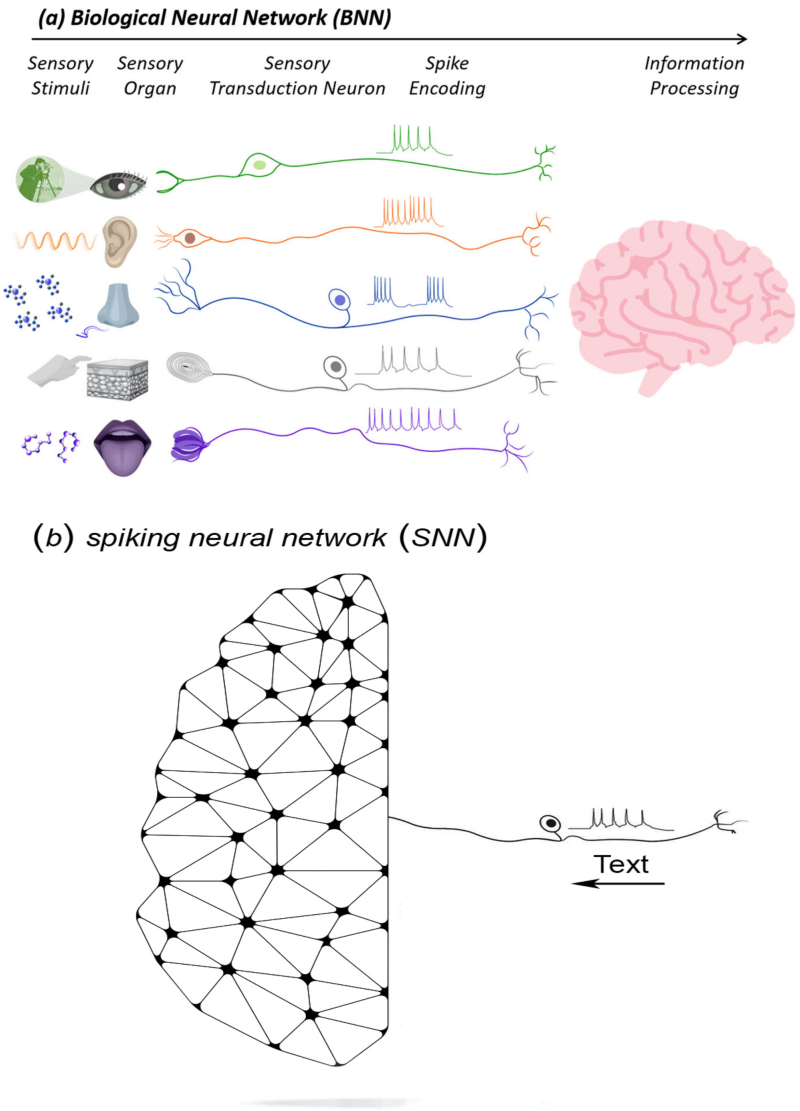


FIGURE 3.4 – La stimulation, et aussi comment les neurones impulsionnels sont stimulés.

3.6 Étapes de création du classifieur

3.6.1 Expression d'un seul neurone impulsionnel

Sur la base de ce que nous avons vu au chapitre 2 et de la façon dont les cellules biologiques étaient représentées par des équations mathématiques primitives, mais qui offrent des performances décentes, nous avons choisi le modèle d'Izehekivich (IM) car il combine l'efficacité de calcul du modèle de Hodgkin-Huxley et la simplicité de LIFM, et cela pour une facilité de manipulation dans le calcul avec ce type de neurones pulsés, même s'ils sont en grand nombre au sein du réseau de neurones impulsionnels.
Bien que l'équation mathématique du IM nous était déjà connue, l'intensité du courant à introduire dans cette équation doit être donnée des valeurs raisonnables difficilement maitrisable, pour que le neurone se comporte correctement et soit proche du neurone biologique dans son activité.

Grâce à l'expérience, nous avons remarqué que l'intensité du courant est directement proportionnelle à la valeur de Δt dans l'équation de IM, et que la valeur initiale du courant de membrane du neurone ne doit pas dépasser 200mV.

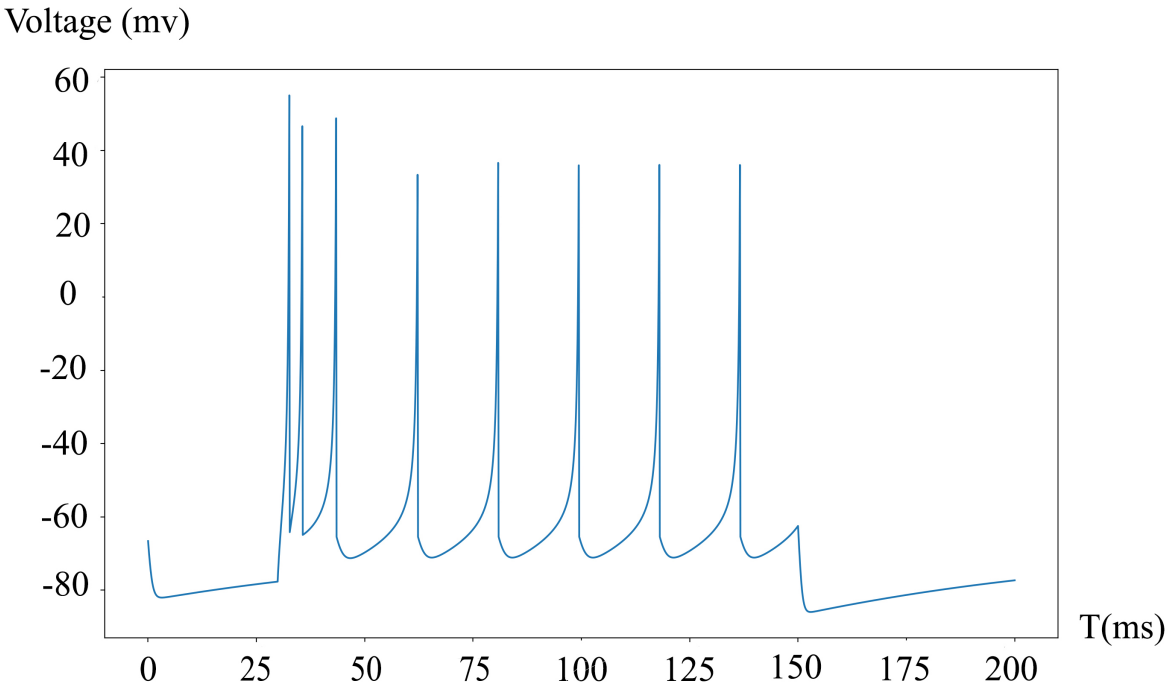


FIGURE 3.5 – La courbe d'activité d'un neurone impulsionnel (Izhikevich (RS)).

La figure précédente montre l'activité d'un neurone impulsionnel de type Izhikevich (RS) après avoir été soumise à une stimulation.

3.6.2 Apprentissage et plasticité synaptique

Par soucis de cohérence, l'utilisation de modèles à canaux pour l'activité du neurone biologique, implique d'utiliser des modèles réalistes sur la façon dont ces neurones interagissent entre eux, c'est-à-dire en reliant ces neurones les uns aux autres, comme dans le réseau de neurones biologiques.
Nous avons adopté un modèle pour connecter ces neurones, qui est basé sur ce que nous avons vu au chapitre 2. Ansi, le temps est un facteur essentiel pour l'apprentissage dans le cerveau humain qui régule l'instant d'activation exact des neurones pulsants. Nous avons ici remplacé le concept de temps par la distance entre les mots, en plus d'utiliser une partie de la règle STDP décrite au chapitre 2.
Soit l'exemple de commentaire positif suivant :

■ bon courage sahib tasahel.

Afin de permettre l'apprentissage de ce commentaire, nous prenons chaque mot un à un et calculons le temps d'occurrence dans la phrase entre lui et les mots qui l'entourent, la distance physique entre les mots dans un commentaire devient une distance temporelle,

par exemple :
Le temps (la distance temporelle) entre "bon" et "courage" est de 1 ms, entre "bon" et "sahib" elle est de 2 ms, et entre "bon" et "tasahel"elle est de 3 ms. Lorsque la fonction STDP est appliquée au Δt (distance temporelle) de chaque couple de mots, poids des synapses de connexion entre les couples de neurones qui les représentent sont modifiés par l'ajout d'une quantité qui obéit qui suit l'exponentielle inverse de la distance temporelle : plus la distance est courte, plus la modification est puissante. Le tableau suivant illustre les distances temporelles entre les mots du commentaire d'exemple :

Δt	**Bon**	**courage**	**sahib**	**tasahel**
Bon	0	1 ms	2 ms	3 ms
courage	X	0	1 ms	2 ms
sahib	X	X	0	1 ms
tasahel	X	X	X	0

TABLE 3.1 – La distance temporelle de chaque couple de mots.

Le tableau qui suit présente la quantité à ajouter aux poids, cette quantité est inversement proportionnelle à la distance temporelle.

$F(\Delta t)$	**Bon**	**courage**	**sahib**	**tasahel**
Bon	0	1.5	1 ms	0.5
courage	X	0	1.5	1
sahib	X	X	0	1.5
tasahel	X	X	X	0

TABLE 3.2 – La quantité a ajouter aux poids (entre chaque couple de mots).

Comme nous l'avons mentionné précédemment, chaque mot dans la langue d'origine est représenté par un neurone impulsionnel, et puisqu'il y a quatre mots dans ce commentaire, il y aura une activité entre ces quatre neurones dans le réseau, et cette activité consiste à créer de connexions entre ces cellules ou renforcer les connexions qui existaient déjà. (Figure 3.6)

Note :

Un neurone peut être affecté par un seul neurone ou un groupe de neurones précédemment connectés.
Ce qu'on entend par stimuler un neurone c'est lui donner un courant d'une certaine valeur, si le stimulus est suffisant, le neurone se déclenchera.

3.6.3 Discrimination des classes par propagation d'ondes

Dans le classifieur que nous proposons, la descrimination entre les classes se fait par l'activité électrique résultante dans tous les connectomes à la proposition d'un commentaire en entrée. Les neurones représentatifs de chaque mot sont stimulés. Par leur activation, ces neurones activent leurs voisins, qui vont à leur tour activer leur voisins produisant

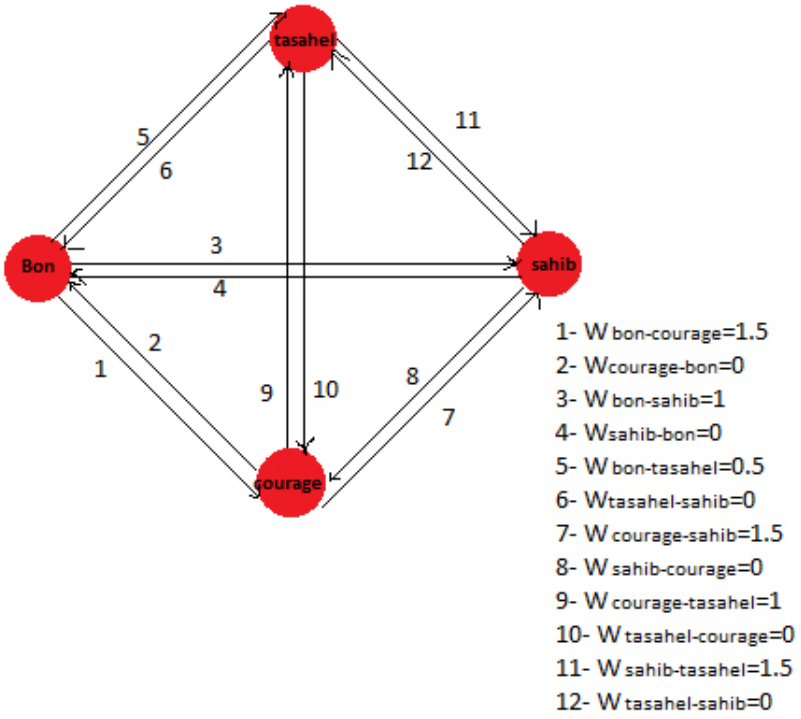


FIGURE 3.6 – Les pois sont créés entre les neurones impulsionnels qui représentent les mots.

des activations en cascades donnant naissance à une onde électrique qui va se propager dans le connectome. Pour schématiser ce concept, nous considérons le commentaire suivant “rabi yra7mou w ywasa3 3lih”.
Les deux figures 3.7 représentent l’activation des neurones à la présentation simultanée du commentaire dans le connectome positif et négatif.
La figure du connectome positif, qui comptabilise le nombre de neurones en activité à

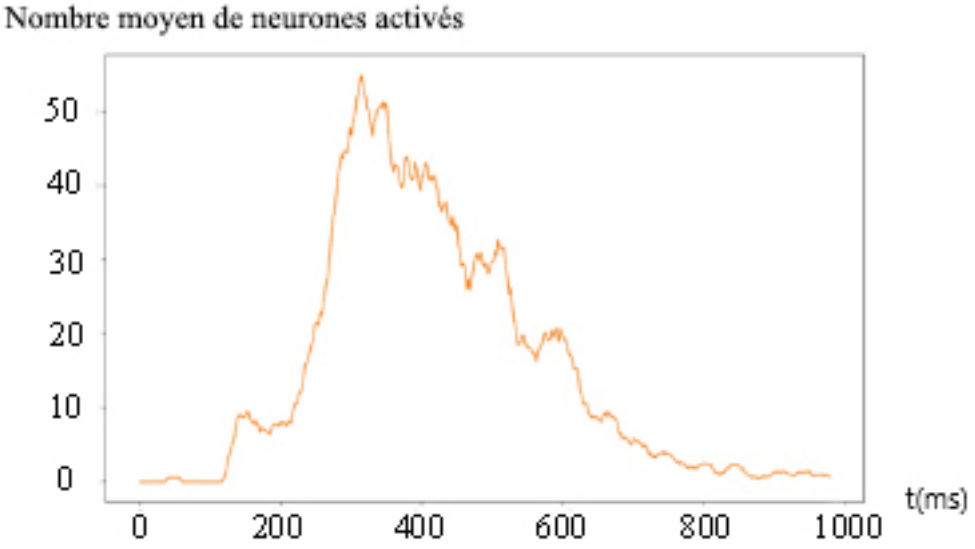


FIGURE 3.7 – La courbe du connectome positif.

un temps t, montre l'instant de la naissance de l'onde au moment de la présentation du commentaire entre 20 et 35ms. Vers 60ms, des activations en cascades commencent à se propager dans le connectome atteignant l'apogée vers 150 ms avec plus de 50 neurones qui s'activent simultanément. De par la topologie du connectome, l'onde finit par s'estomper en atteignant les 'bordures externes' du réseau de neurones.

Le connectome négatif, ne voit se propager aucune onde. Il y a à celà deux causes qui

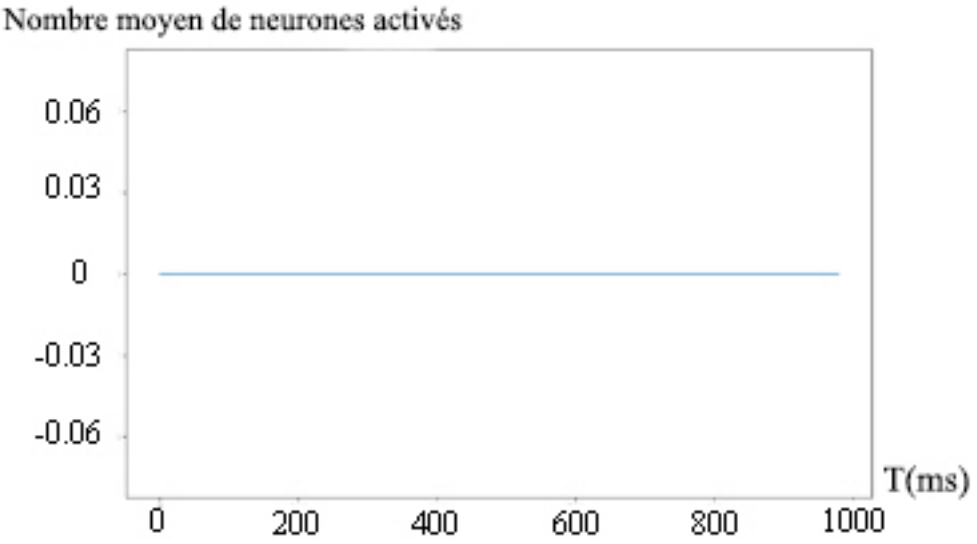


FIGURE 3.8 – La courbe du connectome négatif

se superposent. Premièrement, les mots du commentaire dans ce connectome sont faiblement connectés au reste du réseau car apparaissant trop peu dans la collection des commentaires négatifs. Deuxièmement, nous avons instauré un système d'inhibition entre les connectomes qui crée une concurrence entre eux. Puisque les deux connectomes sont activés simultanément, le premier connectome qui voit naître une onde va imposer une inhibition au second connectome neurone par neurone produisant un silence plus ou moins modéré dans ce dernier.

Pour discriminer les deux classes, nous calculons l'intégrale des neurones activés dans les deux graphes précédents, la classe à l'intégrale la plus grande est sélectionnée. Dans le cas général, nous procédons au calcul des intégrales de tous les connectomes en sélectionnant le plus grand. Le connectome i est sélectionné selon l'équation :

$$\arg_i max(\int_0^{t_{max}} f_i(t)dt) \tag{3.1}$$

où $f_i(t)$ est la fonction du nombre de neurones activés par rapport au temps.

3.6.4 Inhibitions croisées inter-connectomes

Étant donné que chaque connectome a sa propre structure qui a été acquise au cours de l'apprentissage, le nombre de neurones dans chaque connectome et la façon dont ils sont liés les uns aux autres reste confuse et peu clairs pour nous. L'activation de chaque neurone dépend de la valeur de l'intensité du courant en entrée , et dans le cas de son déclenchement, il affecte les neurones qui lui sont connectés, et il n'est donc pas possible

de prédire exactement comment l'onde se propagera.
Afin de surmonter ce problème, nous proposons une inhibition inter-connectome. L'idée d'inhibition est basée sur le fait qu'avant d'activer chaque neurone, nous vérifions si la valeur d'une fonction fitness prédéfinie du mot représenté par ce neurone dans ce connectome est supérieure à celle du même neurone dans l'autre connectome. Le neurone à la fonction de fitness la plus grande s'active. De cette façon, nous sommes assurés que dans chaque connectome les neurones qui lui appartiennent se déclenchent dans leur classification correcte, et il est ainsi possible d'ignorer les mots superflus. La fonction de fitness discriminatoire utilise comme paramètres le nombre de connexions que possède le neurone avec le connectome ainsi que la somme des poids qui l'y relie :

$$f_i(v, p) = 1.05 * v + 1.2 * p \tag{3.2}$$

où v est le nombre de connexions et p la somme des poids qui relient le neurone i au connectome.

3.6.5 Promouvoir l'apprentissage par la dépression (oubli)

Nous avons essayé d'appliquer le principe de dépression afin d'améliorer autant que possible la qualité de l'apprentissage de chaque connectome, augmentant ainsi le taux de classification correct de l'algorithme par l'application du principe de la dépression LTD basé sur ce que nous avons vu dans le chapitre 3. Le LTD est un processus par lequel les connexions synaptiques entre les neurones sont affaiblies. La dépression est le processus inverse de la potentialisation à long terme (LTP).
Le principe de l'idée est simple, au lieu d'apprendre chaque connectome à partir de la partie qui lui est allouée dans la base d'apprentissage uniquement, elle prendra en compte toute la base, c'est-à-dire que chaque connectome apprend la partie qui lui est allouée par potentialisation et apprend la partie qui ne lui est pas allouée à dépression.
En termes plus simples, pour le connectome responsable de la détection des commentaires positifs, au lieu d'apprendre cette section uniquement à partir de commentaires positifs, elle apprend à partir de commentaires positifs en appliquant le principe de LTP comme nous l'avons expliqué précédemment, et également à partir de commentaires négatifs appliquant le principe d'LTD. Le processus LTD suit le processus LTP. Une fois que la section positive est apprise grâce aux commentaires positifs et que les liens synaptiques entre les neurones sont créés avec les poids appropriés, vient la phase de LTD qui dépend des commentaires négatifs. A chaque fois qu'une suite négative négative de deux mots coïncide avec le même ordre dans un commentaire positif, la synapse qui relie les neurones qui représentent ces mots est affaiblie d'un petit pourcentage.
Le but de ce processus lors de l'étape d'apprentissage est que si des mots existent de la même manière dans les commentaires positifs et négatifs présentés au système, LTD lui permet d'affaiblir les synapses entre les neurones qui représentent ces mots, c'est ce qu'on appelle communément l'oubli. De cette manière, nous assurons que les deux connectomes n'apprendront pas la même chose, et assurons ainsi un système de discrimination et de classification correcte.

3.7 Le bruit dans le connectome

3.7.1 Qu'est-ce que le bruit dans le connectome ?

Ce que l'on entend par le bruit d'ondes dans un connectomeest le déclenchement aléatoire de certaines neurones indésirables, ou la poursuite de l'activité d'un neurone sans s'arrêter, ce qui conduit à son impact continu sur les neurones qui lui sont connectées et qui forment ce qu'on appelle des boucles. L'occurrence de ce phénomène conduit à un taux d'erreur élevé. En général le système ne réussira pas à classer correctement dans la plupart des cas.
Parfois pour catégoriser correctement un commentaire particulier comme un commentaire positif ou un commentaire négatif, les deux connectomes répondent avec la même activité. parfois la mauvaise classe répond mieux que la classe correcte, ce qui conduit à de nombreux faux positifs.

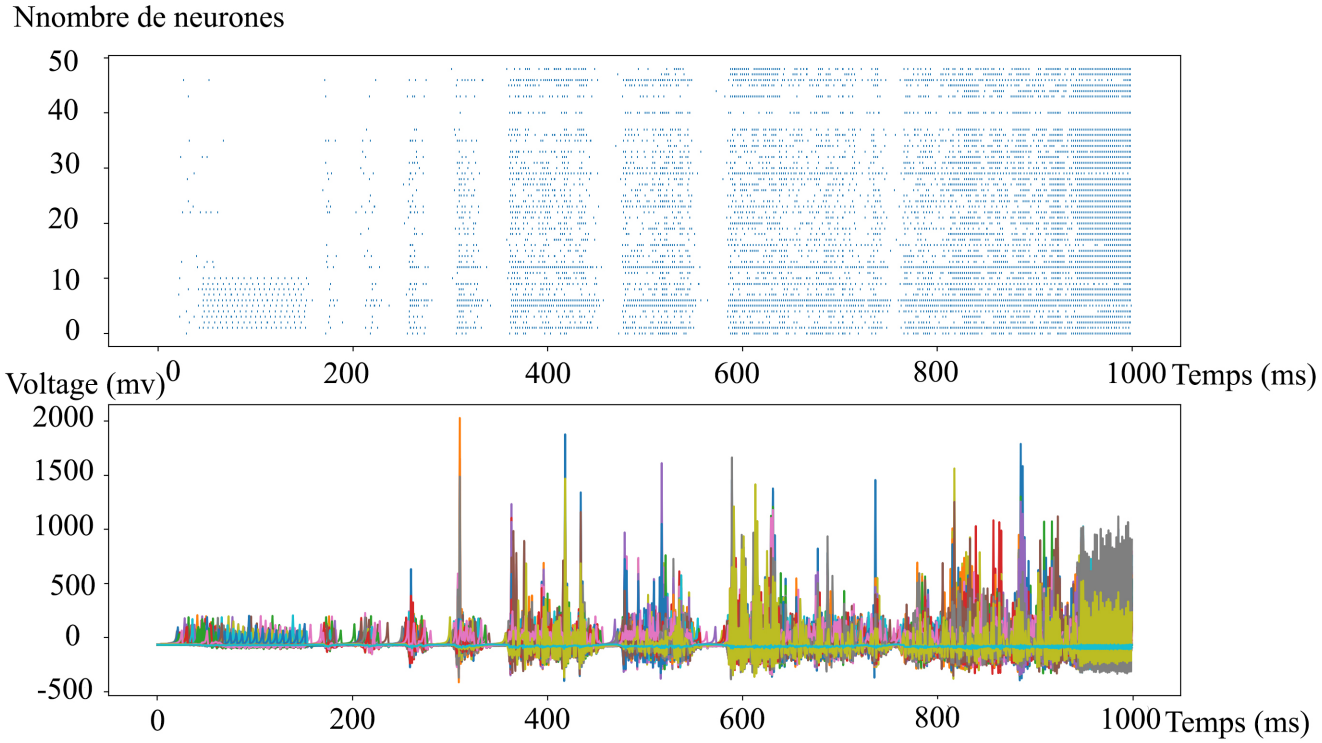


FIGURE 3.9 – L'activité de 50 neurones impulsionnels avec présence de bruit, et instabilité de voltage dans le réseau.

Dans la figure précédente, nous remarquons dans la première partie que les neurones après leur premier déclenchement ne se sont pas arrêtés. Ainsi tous les neurones ont commencé à émettre des impulsions de manière intense.
Dans la deuxième partie de la figure, on note que l'intensité de courant est totalement instable et atteint 2000 mV, alors qu'elle ne doit pas dépasser 200 mV au maximum.
Le processus d'inhibition que nous avons présenté un peu plus haut s'avère être une contre-

mesure efficace au problème de bruit. En effet, l'inhibition permet justement de contrôler les intensités de courant instables.

3.7.2 Créer un dictionnaire unifié pour le classifieur (Tokenisation)

Nous utilisons pour ce cas pratique la langue « darija » tunisienne. Ce choix a été effectué car la darija tunisienne est la seule qui propose un dataset cohérent. Avant de passer à l'apprentissage du classifieur, un prétraitement est nécessaire. Il faut d'abord créer un dictionnaire complet unifié pour cette langue dans un processus communément appelé "tokenisation". Ainsi, un grand nombre de mots dans leur forme pluriels, conjugués ou autres, n'y seront plus référés que sous la forme d'un unique token. Ce dictionnaire a été construire selon ces étapes :

— Les mots tels qu'ils étaient écrits dans l'original ont été sélectionnés, puis radicalisés. Par exemple, le mot « bercha » a été amputé de ses voyelles ne laissant que les consonnes, car la plupart des mots du dialecte tunisien possèdent de nombreuses alternatives, par exemple « bercha » (qui signifie "beaucoup" en tunisien) s'écrit sous plusieurs formes comme « Barcha/brcha/berch », qui ont toutes en commun la racine « brch ».

— Nous supprimons les lettres répétées telles que "hhhhh" et "mconnecti" comme nous l'avons vu auparavant, cela devient "hhhhh" et "mcnnct" et après avoir supprimé les lettres répétées cela devient "h" et "mcnct".

— Lorsque nous rencontrons des mots de liaison comme "W, et, ou...", nous les supprimons.

— Nous gardons les chiffres dans les mots car ils ont un sens comme 9, 5, 7, 8, 3 et 2.

Number	2	3	5	7	8	9
Exemple	2ala2	3ala	5oya	7amed	8adwa	9ader
La racine	2l2	3l	5y	7md	8dw	9dr

TABLE 3.3 – Convertissez les noms qui contiennent des nombres.

Notre dictionnaire sera exclusivement constitué des racines primaires de tous ces mots. Le tableau suivant montre certaines des racines prises dans le dictionnaire :

Les mots	**Ecritures alternatives**	**La racine des mots**
3ambali	3embali/3mbeli/3embalo	3mbl
bercha	Barcha/ brcha /berch	brch
sahib	Sahib/shib	Shb
5oya	5eya /5ayi	5y
9a3ed	9e3ad/9a3ad/9e3ed	93d
mconnecti	mCennecti/mconnecte	cnct

TABLE 3.4 – Changements qui se produisent dans les mots et comment en extraire les racines.

3.7.3 Suppression des caractères de liaison dans les commentaires

Les caractères de liaison comme [w, et, par, bah, zayma, win, ...] n'ont aucune signification positive ou négative dans la langue d'origine, en plus de cela, au cours de la phase d'apprentissage, nous avons remarqué que les neurones qui représentent ces lettres dans les connectomes positifs et négatifs sont connectées à un très grand nombre des neurones sans bénéfice, et conduisent à leur stimulation d'une manière désordonnée et irrégulière. Par exemple, prenons la lettre "w (et)" nous la trouverons associée à un grand nombre de neurones positifs et également à un grand nombre de neurones négatifs. Ainsi déclencher le neurone qui représente la lettre "w" conduira à activer un grand nombre de neurones dans les deux connectomes et conduisent ainsi à l'incapacité du système à classer correctement.

3.7.4 Suppression de certains mots-clés dans les deux connectomes

Par l'expérience, nous avons remarqué que dans les deux connectomes ont des mots qui dans leur classification correcte ne lui appartiennent pas. Les neurones représentant ces mots sont connectés avec un grand nombre d'autres neurones. Au cas où ces neurones représentant ces mots sont déclenchés, ils affectent les neurones qui lui sont connectés, et conduisent ainsi à créer un bruit incapacitant le classifieur à classifier correctement.
La raison de la présence de neurones qui représentent ces mots-clés dans les deux sections de cette manière est due au fait qu'ils sont présents sous cette forme dans la base d'apprentissage L'application n'est pas en mesure de les supprimer automatiquement. En conséquence, nous avons supprimé manuellement certains d'entre eux.
Exemple de mors supprimes [wllh, bravo, bon, salam , ...] dans le connectome négatif.

3.7.5 La liste taboue

Afin d'éviter un bruit trop important conséquent d'une activation éternelle des neurones du connectome qui conduit à la génération d'ondes puissantes non-désirées, il existe un processus naturel qui permet l'extinction automatique d'un neurone après son activation appelée “Inhibition post-impulsionnel” créé par un courant d'hyper-polarisation. Notre modèle de neurone ne permettant pas de rendre compte de ce courant, nous avons décidé d'utiliser le principe de la liste taboue, et ceci afin de s'assurer que l'activité de chaque cellule est arrêtée au bout d'un certain temps à compter de son déclenchement, par le placement de cet neurone dans la liste taboue.

3.8 Conclusion

Dans ce chapitre, nous avons abordé l'algorithme que nous avons conçu pour la classification bio-inspirée du cortex en construisant un système qui s'appuie sur des réseaux de neurones impulsionnels agencés en connectomes.
Nous avons décrit le processus d'apprentissage dans ce contexte ainsi que la manière de discriminer les classes entre elles. Face au problème du bruit, nous avons proposé diverses contre-mesures dont la plus importante et les "inhibitions croisées".
Dans le chapitre suivant, nous parlerons plus spécifiquement des outils et de l'environnement de travail, ainsi que des classes de l'application.

CHAPITRE 4

TESTS ET RÉSULTATS : LE CAS DE L'ANALYSE DES SENTIMENTS

4.1 Introduction

La capacité à créer cette application à partir de zéro et de préparer un environnement pour un modèle d'apprentissage en profondeur et la capacité de tirer parti des données proposées pour l'apprentissage était un défi au début en plus du fait, que l'idée de l'application est nouveau en soi et il n'y a pas d'expériences préalables pour ce travail, dans ce chapitre nous parlerons en général et simplifié sur les outils utilisés et l'environnement de travail, ainsi que le code pour l'application.

4.2 Environnement de travail et outils

Notre travail a été développé en Python, où nous avons utilisé un ensemble de bibliothèques et d'outils au cours des différentes étapes de création de l'application, depuis l'entraînement et le traitement des mots jusqu'à la catégorisation des commentaires et des textes, ce afin d'obtenir les meilleures performances possibles avec le moindre effort, et c'est ce dont nous discuterons dans cette partie.

4.2.1 Bibliothèques Python

Une bibliothèque est une collection de codes pré-combinés qui peuvent être utilisés de manière itérative pour réduire le temps nécessaire au codage. Elles sont particulièrement utiles pour accéder aux codes pré-écrits fréquemment utilisés, au lieu de les écrire à partir de zéro à chaque fois. Comme les bibliothèques physiques, elles constituent une collection de ressources réutilisables, ce qui signifie que chaque bibliothèque a une source racine. C'est le fondement des nombreuses bibliothèques open-source disponibles en Python.

4.2.1.1 Bibliothèques de calculs

Numpy :

NumPy est un paquetage Python, qui signifie Python Numérique, est une bibliothèque composée d'objets de tableaux multidimensionnels et d'une collection de routines pour traiter ces tableaux. NumPy permet d'effectuer des opérations mathématiques et logiques sur des tableaux. `https://www.tutorialspoint.com/numpy/index.htm`
Nous avons utilisé cette bibliothèque pour les calculs des équations mathématiques qui représentent les neurones impulsionnels, ce qui nous a permis de réduire considérablement le temps passé sur l'exécution de ce calcul

Math :

Le module mathématique Python est une fonctionnalité importante conçue pour traiter les opérations mathématiques. Il est fourni avec la version standard de Python et est présent depuis le début. La plupart des fonctions du module mathématique sont de fines enveloppes autour des fonctions mathématiques de la plate-forme C. Comme ses fonctions sous-jacentes sont écrites en CPython, le module mathématique est efficace et conforme à la norme C. `https://realpython.com/python-math-module/`
Nous avons utilisé cette bibliothèque car elle contient de nombreuses fonctions prédéfinies, ce qui nous évite de les réécrire, par ex : nous l'avons utilisée pour la fonction log() pour normaliser les poids des neurones.

Random :

Le module aléatoire est un module intégré permettant de générer des variables pseudo-aléatoires. Il peut être utilisé pour effectuer une action de manière aléatoire, par exemple pour obtenir un nombre aléatoire, sélectionner un élément aléatoire dans une liste, mélanger des éléments de manière aléatoire, etc. `https://www.tutorialsteacher.com/python/random-module`
Nous avons utilisé cette bibliothèque afin de définir une distribution de probabilité sur l'occurrence (ou non) de l'inhibition inter-connectomes. Si le nombre aléatoire tiré satisfait la distribution pour un couple de neurones, ces neurones subiront le processus d'inhibition.

4.2.1.2 Prétraitement et sauvegarde des données

Re :

Les expressions régulières, souvent abrégées en regex, sont une séquence de caractères utilisée pour vérifier si un motif existe ou non dans un texte (chaîne) donné. Si vous avez déjà utilisé des moteurs de recherche, des outils de recherche et de remplacement dans des traitements de texte et des éditeurs de texte, vous avez déjà vu des expressions régulières. Elles sont utilisées côté serveur pour valider le format des adresses électroniques ou des mots de passe lors de l'enregistrement, pour analyser des fichiers de données textuelles afin de trouver, remplacer ou supprimer certaines chaînes de caractères, etc. Elles aident à manipuler les données textuelles, ce qui est souvent un prérequis pour les projets de science des données impliquant l'exploration de texte.`https://www.datacamp.com/community/tutorials/python-regular-expression-tutorial`

Nous avons utilisé cette bibliothèque pour extraire les mots de texte, car elle permet de diviser le texte pour prendre les mots et ignorer le reste des symboles et dles emojis.

CSV :

Un fichier CSV (Comma Separated Values file) est un type de fichier texte brut qui utilise une structuration spécifique pour organiser les données tabulaires. Comme il s'agit d'un fichier texte brut, il ne peut contenir que des données textuelles réelles, c'est-à-dire des caractères ASCII ou Unicode imprimables.
La structure d'un fichier CSV est révélée par son nom. Normalement, les fichiers CSV utilisent une virgule pour séparer chaque valeur de données spécifique.`https://realpython.com/python-csv/`
Nous avons utilisé cette bibliothèque pour traiter les fichiers de type CSV, car la base d'apprentissage était un fichier de type CSV.

Pickle :

est utilisé pour sérialiser et désérialiser les structures d'objets Python, également appelé marshalling ou flattening. La sérialisation fait référence au processus de conversion d'un objet en mémoire en un flux d'octets qui peut être stocké sur disque ou envoyé sur un réseau. Plus tard, ce flux de caractères peut ensuite être récupéré et désérialisé pour revenir à un objet Python. Le décapage ne doit pas être confondu avec la compression ! La première est la conversion d'un objet d'une représentation (données en mémoire vive (RAM)) à une autre (texte sur disque), tandis que la seconde est le processus d'encodage des données avec moins de bits, afin d'économiser de l'espace disque. `https://www.datacamp.com/community/tutorials/pickle-python-tutorial#what`
Nous avons utilisé cette bibliothèque afin de stocker les connectomes sur le disque dur, ainsi que le dictionnaire du classifieur, car elle permet de sauvegarder des fichiers sous forme de code binaire et se caractérise également par la rapidité de la lecture et l'écriture

4.2.1.3 Le ploting des données :

Matplotlib :

matplotlib.pyplot est une collection de fonctions de style commande qui font fonctionner matplotlib comme MATLAB. Chaque fonction pyplot apporte des modifications à une figure : par exemple, elle crée une figure, crée une zone de traçage dans une figure, trace des lignes dans une zone de traçage, décore le tracé avec des étiquettes, etc. Dans matplotlib.pyplot, plusieurs états sont préservés à travers les appels de fonction, de sorte qu'il garde la trace de choses telles que la figure et la zone de traçage actuelles, et les fonctions de traçage sont dirigées vers les axes actuels. `https://matplotlib.org/2.0.2/users/pyplot_tutorial.html`
Nous avons utilisé cette bibliothèque afin de tracer les courbes qui montrent la propagation des ondes dans les connectomes.

NetworkX :

NetworkX est un progiciel en langage Python pour la création, la manipulation et l'étude de la structure, de la dynamique et des fonctions des réseaux complexes. Pygraphviz est

une interface Python pour le progiciel de mise en page et de visualisation de graphes Graphviz.`https://www.python-course.eu/networkx.php`
Nous avons utilisé cette bibliothèque afin de faciliter l'interaction visuelle avec les neurones, lors du dessin de ces neurones et des neurones qui leur sont connectés.

4.2.1.4 Les Classes de l'application

Nous allons évoquer la structure du code de l'application avec une définition simple et brève, mais tout reste expliqué en détail dans l'application, que nous avons mis en ligne sur github et accessible à l'adresse `https://github.com/Tarek-DG/wcsnn`.

dictionary :

Une classe dédiée au traitement de la langue que le classifieur apprend, car elle permet de traiter des mots et des textes afin d'obtenir un vocabulaire unifié spécifique au classifieur. Cette classe tokenize les mots (pour chaque mot correspond un identifiant numérique unique) pour faciliter le traitement ultérieur.

— get_serial_number() : C'est une fonction qui prend un mot ou un texte et renvoie un nombre ou une série de nombres qui représentent le mot ou le texte saisi selon le dictionnaire du classifieur.

Train :

Cette classe est dédiée à la conversion de la base l'apprentissage, qui est un ensemble de commentaires pré-classés en un ensemble de chaînes numériques qui expriment ces commentaires en fonction du dictionnaire du classifieur, et ce afin de faciliter et d'accélérer les traitements ultérieurs dans la partie restante du classifieur.

— converting_comments() : C'est une fonction qui convertit les commentaires de la règle d'apprentissage en chaînes numériques

Network :

Cette classe est responsable de l'apprentissage du système, car elle permet la mise en place du réseau ou de la structure qui représente chaque connectome en fonction de ce que nous fournissons au système à apprendre.
Cette classe contient un ensemble de fonctions qui permettent d'enseigner le système, ainsi qu'un ensemble de fonctions qui permettent d'identifier tous les détails et informations du réseau.

— Quelques fonctions de base :
  - learn_srial_number_by_ltp() : Cette fonction permet d'apprendre à commenter selon le principe de LTP.
  - learn_srial_number_by_ltd() : Cette fonction permet d'apprendre à commenter selon le principe de LTD.

- write_network() : Permet d'enregistrer la structure de chaque connectome dans un fichier Pickle.
- most_used_roots() : Il vous permet de connaître les mots les plus utilisés dans ce connectome, ainsi que le nombre de leurs liens et les poids qui les représentent.
- nomber_of_living_neurons() :retourne le nombre de neurones pulsants actifs à ce moment 't' de la simulation.

SentimentAnalysis :

Cette classe est chargée de classer les textes et les commentaires, en fonction de l'activité des neurones impulsionnels représentés par l'équation de Izhikevich et de la façon dont ils communiquent entre eux dans chaque connectome, car le mode de communication est la structure qui a été conçue par la classe network.py lors de la phase d'apprentissage.

— Quelques fonctions de base :

- activate() : C'est la fonction chargée de stimuler les neurones impulsionnels qui représentent les mots du commentaire, puis de suivre l'activité du connectome.
- neuron_fitness() : C'est la fonction qui permet aux neurones d'inhiber le neurone correspondant dans l'autre connectome.

viso :

Cette classe est destinée à l'interaction visuelle avec les neurones impulsionnels, car elle permet de visualiser chaque neurone et les neurones qui lui sont connectées.

4.3 Test et Résultats

Lors de cette partie, nous allons présenter les résultats des tests effectués grâce à notre classifieur impulsionnel. Nous présenterons le matériel utilisé, la taille du dataset considéré et les différents temps d'exécutions et précisions obtenus.

4.3.1 Les appareils utilisés

Les simulations du classifieurs ont toutes étaient effectué sur un ordinateur de type :

— Apple MackBook pro
— Puce Apple M1 avec CPU 8 cœurs, GPU 8 cœurs et Neural Engine 16 cœurs
— 16 Go de mémoire unifiée

4.3.2 La base d'apprentissage

Afin d'obtenir un système capable d'acquérir une connaissance approfondie et fiable, nous avons utilisé une base d'apprentissage pré-étiquetée.

4.3.2.1 Le dataset du dialecte tunisien

Ce dataset est issu de la compétition ZINDI intitulée “AI4D iCompass Social Media Sentiment Analysis for Tunisian Arabizi” et accessible à cette adresse AI4D iCompass Social Media Sentiment Analysis for Tunisian Arabizi. Cette base contient des commentaires tirés des sites de réseaux sociaux et est classée en trois classes.

— Commentaires positifs.

— Commentaires négatifs.

— Commentaires neutres.

Le nombre total de commentaires est 70 000, nous en avons utilisé 90% et en avons laissé 10% pour le test.
Le nombre de commentaires positifs est 38 293, et le nombre de commentaires négatifs est 29 295, tandis que le nombre de commentaires neutres que nous avons eu est 2412.
Nous avons complètement ignoré les commentaires neutres lors de la phase d'apprentissage du système, car leur nombre est très faible par rapport aux deux autres types, et c'est ce qui a rendu le connectome pour ce type de commentaires lors des tests dans les premières étapes de création de l'application incapable rivaliser avec les deux autres connectomes, ce qui nous a incités à l'ignorer.

4.3.2.2 Les résultats

Afin de tester notre classifieur et d'acquérir une réelle expérience, nous avons entraîné ce classifieur à partir de la base d'apprentissage du dialecte tunisien que nous avons évoqué précédemment, et nous illustrons les résultats que nous avons obtenus dans les tableaux suivants :

Apprentissage :

Nous avons donné au classifieur un ensemble de commentaires de types positifs et négatifs, afin qu'il puisse apprendre et construire les connectomes respectifs, à travers desquels il devient capable de classer correctement. Voici les temps d'exécution obtenus :

Type de commentaire	Le nombre des commentaires	Temps d'exécution
Positive	34640	5.436 sec
Négatif	21170	5.564 sec

Table 4.1 – Les résultats de l'apprentissage.

— **Type de commentaire :** La classe à laquelle appartient le commentaire.

— **Le nombre des commentaires :** Le nombre de commentaires appris.

— **Temps d'exécution :** Le temps qu'il faut pour apprendre.

Comme on peut le voir dans le tableau précédent, le nombre de commentaires varie entre le type de commentaires positifs et négatifs. Initialement, lorsque nous avons pris 90% de commentaires positifs et négatifs et appris le classifieur, nous avons constaté que le nombre

de liens dans la connectome négative est presque le double du nombre de liens dans la connectome positive, et la raison de cette différence est que la taille des commentaires positifs dans la base d'apprentissage est courte par rapport à la taille des commentaires négatifs. Cela a permis au connectome de détection des commentaires négatifs de créer un plus grand nombre de liens entre les neurones par rapport au connectome positive.
Par souci d'équité entre les deux connectomes, nous avons pris 90% des commentaires positifs et nous avons compté le nombre de mots dans ces commentaires qui est de 203000. Par la suite, nous avons énuméré les mots des commentaires négatifs et nous nous sommes arrêtés lorsque le nombre de mots dans les commentaires est devenu égal au nombre de mots dans les commentaires positifs.
De cette façon, la différence entre le nombre de liens entre les deux connectomes est devenue acceptable, qui est 269121 dans la connectome positive et correspondant à 310844 dans la connectome négative.
Ceci explique que malgré la grande différence dans le nombre de commentaires qui ont été marqués pour le connectome positif et le connectome négatif, le temps d'exécution est presque le même.
Nous tenons à faire remarquer que cette tendance de la sur-représentation des commentaires négatifs est un problème plus social que technique. Les internautes ont plus tendance à écrire de longs commentaires négatifs que positifs et celà impacte tous les analyseurs de sentiments du monde. Le meilleur exemple serait le bot de Microsoft appelé Tay qui en apprenant depuis Twitter est devenu raciste.
Nous faisons remarquer que le temps d'entraînement de notre classifieur est exceptionnellement court en comparaison aux temps moyens nécessaires à l'entraînement de réseaux profonds "traditionnels" qui se compte et heures et en jours.

Résultat :

Dans ce qui suit, nous présentons la précisions de notre classifieur lors de la validation et du test.

A - Validation

Afin de vérifier la crédibilité du classifieur, nous avons testé sa capacité pour le classement correcte sur un ensemble de commentaires qu'il avait préalablement appris et nous avons obtenu les résultats présentés dans le tableau suivant :

Type de commentaire	**Le nombre des commentaires**	**Temps d'exécution**	**Pourcentage de précision**
Positive	3598	1 H 5 min 51 sec	78.889 %
Négatif	3560	1 H 5 min 57 sec	72.174 %

Table 4.2 – Les résultats de la validation.

— **Le nombre des commentaires :** Nombre de commentaires testés.
— **Temps d'exécution :** Le temps qu'il a fallu pour tester ces nombre de commentaires.

— **Pourcentage de précision :** Pourcentage de correcte réponse pour le classifieur

Le temps de réponse du classifieur est d'approximativement 1s par commentaire.

B - Test

Et afin d'obtenir le pourcentage de précision du classifieur, nous avons testé sur un ensemble de commentaires qu'il n'avait jamais appris, et nous avons pu obtenir les résultats indiqués dans le tableau suivant :

Type de commentaire	**Le nombre des commentaires**	**Temps d'exécution**	**Pourcentage de précision**
Positive	3598	1 H 6 min 4 sec	75.034 %
Négatif	3560	1 H 6 min 13 sec	70.431 %

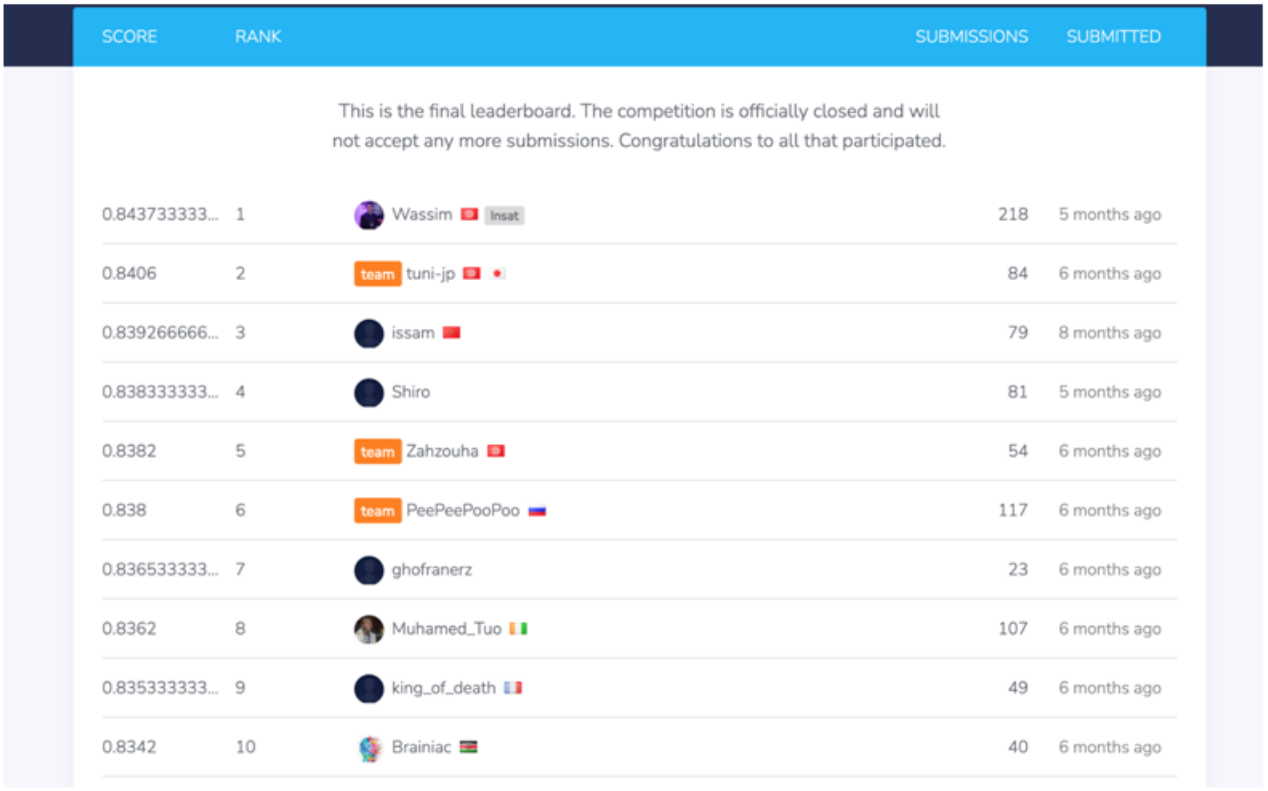
TABLE 4.3 – Les résultats de test.

Comme nous le notons dans les deux tableaux ci-dessus, la différence de pourcentage de précision entre les connectomes négatives et positives n'est pas grande, ce qui indique que la structure des deux connectomes est proche en taille et en nombre de synapses.

4.3.2.3 Pourquoi on ne peut pas comparer avec d'autres modèles ?

Nous n'avons pas comparé les résultats que nous avons obtenus avec d'autres classifieurs de la littérature, car ce dataset n'est pas utilisé de manière académique. Notre but initial était double, proposer un nouveau classifieur, et le tester sur un dataset qui soit d'une utilité pour nous. L'idéal aurait été d'utiliser un dataset de la darija algérien, malheureusement il n'existe pas encore bien qu'un projet soit en route pour le créer.
Néanmoins, nous avons obtenu des résultats très satisfaisants, avec un pourcentage de précision de 75%, lorsque l'on compare ce type de réseau de neurones avec les réseaux de neurones classiques, qui atteignent généralement 80% - 84%. Les meilleurs résultats obtenus pour ce dataset tournent autour de 84% (voir figure ci-dessous) et utilisent des technologies récentes de type "Transformer" et mécanisme d'attention. Nous considérons que notre résultat est satisfaisant bien qu'il soit encore possible d'améliorer davantage notre classifieur qui en est à ses balbutiements. Nous considérons que le temps d'apprentissage extrêmement court est un avantage décisif et que le rapport "temps d'entraînement"/"précision" est favorable.

Au cours de l'expérimentation et de notre revue de la base d'apprentissage utilisée, nous avons remarqué qu'il y a des commentaires que deux personnes peuvent différer dans leur classification, et il y a des commentaires dont la classification dépend du contexte de la phrase, ou des commentaires dans lesquels des termes sont manipulés, qui peut sembler normal à l'esprit humain mais reste déroutant pour le système. D'autant qu'il s'appuie sur lui pour apprendre, et cela affecte la qualité de classification.
Nous faisons remarquer que le classifieur que nous proposons est hautement modulaire. C'est-à-dire que par son architecture une classe = un connectome, il est l'un des rares



SCORE	RANK		SUBMISSIONS	SUBMITTED
This is the final leaderboard. The competition is officially closed and will not accept any more submissions. Congratulations to all that participated.				
0.843733333...	1	Wassim Insat	218	5 months ago
0.8406	2	team tuni-jp	84	6 months ago
0.839266666...	3	issam	79	8 months ago
0.838333333...	4	Shiro	81	5 months ago
0.8382	5	team Zahzouha	54	6 months ago
0.838	6	team PeePeePooPoo	117	6 months ago
0.836533333...	7	ghofranerz	23	6 months ago
0.8362	8	Muhamed_Tuo	107	6 months ago
0.835333333...	9	king_of_death	49	6 months ago
0.8342	10	Brainiac	40	6 months ago

FIGURE 4.1 – Top 10 des meilleurs algorithmes pour la classification du dataset “darija tunisien” la plupart basés sur des technologies “Transformers”.

à supporter l’ajout de nouvelles classes à un classifieur déjà entraîné, très rapidement et sans avoir à le réentrainer entièrement. Cela semble plus logique et plus proche de ce qui se passe dans la nature, en effet, un humain ne réapprend pas tout un corpus à l’occurrence d’une nouvelle information.
Malgré les résultats modestes, nous restons confiants qu’au final, avec une base d’apprentissage soigneusement sélectionnée est présentée, le taux de précision du système peut dépasser les 90% .

4.4 Conclusion

Dans ce chapitre, nous avons en particulier discuter de notre application, les classes et les outils utilisés, ainsi que la base d’apprentissage que nous avons utilisée. Nous avons présenté les tests et les résultats que nous avons pu effectué et obtenu.
Finalement, les résultats obtenus sont modestes et quelque peu satisfaisants. Nous sommes conscients que la marge de manœuvre pour les améliorer reste conséquente et que cette voie est très prometteuse.

Conclusion générale

Tenter d'imiter la nature est un domaine de recherche très fructueux, bien que notre capacité à cloner des systèmes biologiques en soit encore à ses balbutiements, les résultats obtenus sont considérés comme encourageants pour de futurs efforts. Dans ce mémoir et lors de la création de ce classifieur , nous nous sommes appuyés sur la prochaine génération de réseaux de neurones, qui sont des réseaux de neurones impulsionnels, qui se caractérisent par le fait d'être plus proches des cellules biologiques dans le principe de leur activité et de leur connexion les uns aux autres.
Afin que notre classifieur acquière une connaissance approfondie qui lui permettrait de classer correctement ses entrées, nous avons d'abord emprunté le principe de travail de ces neurones dans le cortex et la façon dont la connaissance est représentée dans celui-ci. Ensuite, nous avons choisi le type de neurones pulsés de type Izhikevich pour notre classifieur, car ce type de neurones a une capacité de calcul très efficace lorsque le nombre de ces cellules est grand , et un dataset tiré des sites de réseaux sociaux pour le dialecte tunisien. Pour représenter la connaissance nous avons appliqué le principe STDP afin de créer des synapses entre ces neurones impulsionnels du classifieur.
Bien que ce type de neurone manque de fonctions d'apprentissage appropriées et de fonctions de correction d'erreur efficaces, le pourcentage de précision du classifieur atteint environ 75%, et les résultats sont considérés comme très satisfaisants par rapport aux réseaux de neurones classiques dans lesquels le pourcentage de précision des classifieurs d'analyse des sentiments sont d'environ 84%, bien qu'ils aient des fonctions qui leur sont assignées pour l'apprentissage et la correction des erreurs.
Ajoutez à celà le fait que l'apprentissage de notre classifieur est beaucoup plus rapide (quelques secondes) par rapport aux systèmes les plus récents qui nécessite des heures et parfois des jours d'entraînement. De plus, notre système étant hautement modulaire, il serait possible d'adjoindre de nouvelles classes en ajoutant de nouveaux connectomes sans avoir à réentraîner entièrement le réseau.
Cette recherche présentée dans ce mémoire est considérée comme un prélude à d'autres travaux basés sur ce type de réseau de neurones à l'avenir, et nous nous réjouirons de fournir des fonctions d'apprentissage et de correction d'erreurs dédiées à ce type de réseau.

BIBLIOGRAPHIE

[1] Snehashish Chakraverty, Deepti Moyi Sahoo, and Nisha Rani Mahato. Concepts of soft computing fuzzy and ann with programming. 2019.

[2] P. Nakamoto. Neural networks and deep learning. 2017.

[3] S.N. Sivanandam, S. Sumathi, and S.N. Deepa. Introduction to neural networks using matlab 6.0. 2006.

[4] F. Laurene. Fundamentals of neural networks—architectures, algorithms, and applications. *Prentice Hall*, 1994.

[5] Alexandre Bergel. Agile artificial intelligence in pharo. 2020.

[6] Hani Zerzaihi et Fouad Zarour. Reconnaissance d images par les réseaux de neurones convolutifs. 2020.

[7] Krine Zohra and Abdi Ahlem. Utilisation des réseaux de neurones convolutifs pour la suppression automatique des filigranes. 2020.

[8] Marc Parizeau. Le perceptron multicouche et son algorithme de rétropropagation des erreurs. 2004.

[9] Nikola K. Kasabov. Time-space, spiking neural networks and brain-inspired artificial intelligence. 2019.

[10] RITA CARTER, Susan aldridge, Martyn Page, and Steve Parker. Human brain the book. 2019.

[11] Jiang Wang, Liangquan Chen, and Xianyang Fei. Analysis and control of the bifurcation of hodgkin–huxley model. 2005.

[12] W.M. Kistler W. Gerstner. Spiking neuron models : Single neurons, populations, plasticity. *Cambridge University Press, Cambridge*.

[13] J. Rinzel M. Nelson. The hodgkin-huxley model. in the book of genesis. *Springer, New York*.

[14] I. Segev C. Meunier. Playing the devil's advocate : is the hodgkin—huxley model useful ? *Trends Neurosci*.

[15] C.M. Armstrong F. Bezanilla. Inactivation of the sodium channel. 1997.

[16] FitzHugh R. Mathematical models of threshold phenomena in the nerve membrane. 1955.

[17] FitzHugh R. Impulses and physiological states in theoretical models of nerve membrane. 1961.

[18] FitzHugh R. Motion picture of nerve impulse propagation using computer animation. *Applied Physiology*, 1968.

[19] Izhikevich E. M. Dynamical systems in neuroscience : The geometry of excitability and bursting. *The MIT Press, Cambridge, MA*, 2007.

[20]

[21] John W. Milnor. Attractor. scholarpedia. 2006.

[22] Eric T. Shea-Brown Jeff Moehlis, Kresimir Josic. Periodic orbit. scholarpedia. 2006.

[23] Philip Holmes and Eric T. Shea-Brown n. Stability, scholarpedia. 2006.

[24] E.M. Izhikevich. Simple model of spiking neurons. *IEEE Trans. Neural Netw*, 2003.

[25] W. Maass. On the role of time and space in neural computation. *Math. Found. Comput. Sci*, 1998.

[26] S.M. Bohte. The evidence for neural information processing with precise spike-times : a survey. *Nat. Comput*, 2004.

[27] D.O. Hebb. The first stage of perception : Growth of the assembly. 2002.

[28] Wulfram Gerstner and Werner M. Kistler. Mathematical formulations of hebbian learning. *Biological Cybernetics*, 2002.

[29] W. Bechtel and A Abrahamsen. Connectionism and the mind. *Oxford, UK : Blackwell*, 1993.

[30] D.O Hebb. The organization of behavior. *New York, Wiley.*, 1949.

[31] Caporale N and Dan Y. Spike timing-dependent plasticity :a hebbian learning rule. *Annual Review of Neuroscience.*

[32] H. La Poutre S.M. Bohte, J.N. Kok. Spikeprop : Backpropagation for networks of spiking neurons error-backpropagation in a network of spiking neurons. *ESANN*, 2000.

[33] Sander M. BohteJoost N. KokHan La Poutré. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, 2002.

[34] Jessell TM 2000 Kandel ER, Schwartz JH. Principles of neural science. *New York. McGraw-Hill*, 2013.

[35] Ritz R. Hemmen J.L Gerstner, W. Why spikes ? hebbian learning and retrieval of timeresolved excitation patterns. *Biol. Cybern*, 1993.

[36] Kempter R. Leo van Hemmen-J. Wagner H. Gerstner, W. A neuronal learning rule for sub-millisecond temporal coding. *Lett. Nat*, 1996.

[37] J.M. Young. Cortical reorganization consistent with spike timing-but not correlationdependent plasticity. *Nat. Neurosc.*, 2007.

[38] Haney S. Bazhenov M. Stopfer-M. Sejnowski T.J. Finelli, L.A. Synaptic learning rules and sparse coding in a model sensory system. *PLoS Comput*, 2008.

[39] Guyonneau R. Thorpe S.J. Masquelier, T. Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains. *PLoS ONE*, 2008.

[40] Guyonneau R. Thorpe S.J. Masquelier, T. Competitive stdp-based spike pattern learning. *Neural Comp*, 2009.

[41] Lbke J. Frotscher M. Sakmann-B. Markram, H. Regulation of synaptic efficacy by coincidence of postsynaptic aps and epsps. *Science*, 1997.

[42] Poo M. Bi, G. Synaptic modifications in cultured hippocampal neurons : dependence on spike timing, synaptic strength, and postsynaptic cell type. j. *Neurosci*, 1998.

[43] Poo M.M. Bi, G. Synaptic modification by correlated activity : Hebb's postulate revisited. ann. rev. *Neurosci*, 2001.

[44] Tao H. Holt C. Harris-W. Poo M. Zhang, L. A critical window for cooperation and competition among developing retinotectal synapses. *Nature*, 1998.

[45] D. : Feldman. Timing-based ltp and ltd at vertical inputs to layer ii/iii pyramidal cells in rat barrel cortex. *Neuron*, 2000.

[46] Poo M.M. Mu, Y. Spike timing-dependent ltp/ltd mediates visual experience-dependent plasticity in a developing retinotectal system. *Neuron*, 2006.

[47] Laurent G. Cassenaer, S. Hebbian stdp in mushroom bodies facilitates the synchronous flow of olfactory information in locusts. *Nature 448(7154), 709–713.*

[48] et al. Jacob, V. Spike-timing-dependent synaptic depression in the in vivo barrel cortex of the rat. *Neurosc*, 2007.

[49] Gerkin R.C. Bi G.-Q.-Chow C.C. Rubin, J.E. Calcium time course as a signal for spiketiming-dependent plasticity. *Neuroph*, 2005.

[50] Mohamed Nadjib Zennir. Spike-time dependant plasticity in a spiking neural network for robot path planning. 2015.

[51] Abir Hussain (Eds.) De-Shuang Huang, Kyungsook Han. Intelligent computing methodologies. 2016.

[52] Sirakoulis Andrew Adamatzky Editors Leon Chua, Georgios Ch. Handbook of memristor networks. 2019.

[53] Maass. Networks of spiking neurons : The third generation of neural network models. *Wolfgang.*

Résumé

Dans cette étude nous avons utilisé un classifieur basé sur les réseaux de neurones impulsionnels. Dans ce travail, nous utilisons un type de neurone impulsionnel basé sur le modèle d'Izhikevich. Ce classifieur a la particularité de faire correspondre chaque classe à un connectome autonome. La discrimination entre les classes se fait alors par l'analyse de la propagation d'une onde d'activité neuronale dans chaque connectome. La classe élue sera celle dont le connectome est le plus actif. Nous avons appliqué ce classifieur sur une base de commentaires en dialecte tunisien (70000 commentaires) dans une optique d'analyse de sentiments et que nous avons divisé à 90% pour l'entraînement et à 10% pour le test .Les résultats des expériences atteignent 75% de classifications correctes des commentaires positifs et 70% des commentaires négatifs .Les résultats ont dépassé le stade de la spéculation (50%), bien qu'il s'agisse de recherche. Nous espérons atteindre de meilleurs résultats dans l'avenir en optimisant davantage ce classifieur.

Mots-clés *:Classification, Réseaux de neurones impulsionnels, Modèle d'Izhikevich , STDP, Analyse des sentiments.*

Abstract

In this study we used a classifier based on spiking neural networks. In this work, we use a type of spiking neuron based on the Izhikevich model. This classifier has the particularity to map each class to an autonomous connectome. The discrimination between classes is then done by analyzing the propagation of a neural activity wave in each connectome. The elected class will be the one whose connectome is the most active. We applied this classifier on a base of comments in Tunisian dialect (70000 comments) in a perspective of sentiment analysis and that we divided at 90% for training and at 10% for testing. The results of the experiments reach 75% of correct classifications of positive comments and 70% of negative comments. The results exceeded the stage of speculation (50%), although it is research. We hope to achieve better results in the future by further optimizing this classifier.

Keywords : *Classification, Spiking Neural Networks, Izhikevich Model, STDP, Sentiment Analysis.*