République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Dr. Yahia Ferés de Médéa

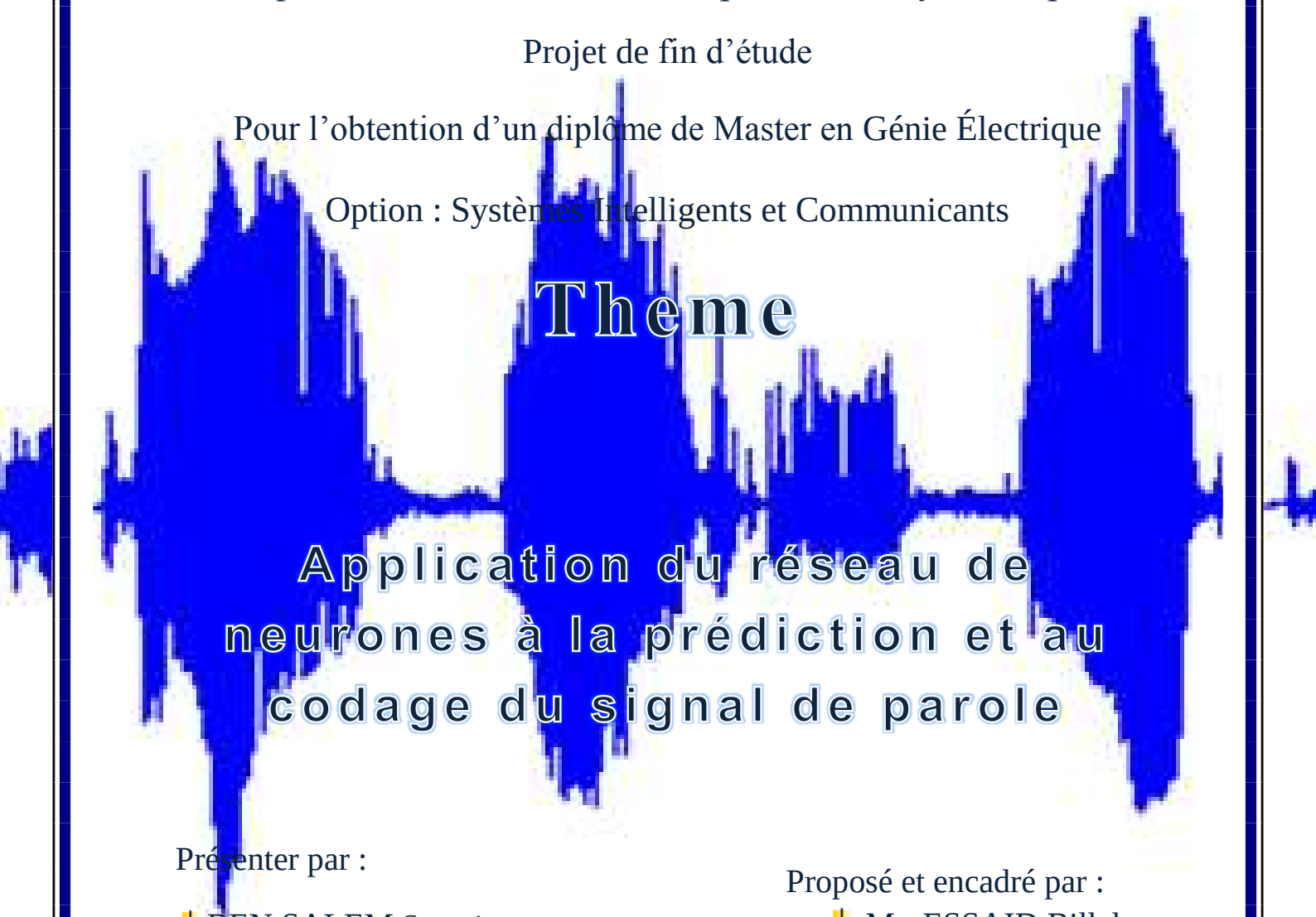
Département du Génie Électrique et de l'Informatique

Projet de fin d'étude

Pour l'obtention d'un diplôme de Master en Génie Électrique

Option : Systèmes Intelligents et Communicants

Theme



Application du réseau de neurones à la prédiction et au codage du signal de parole

Présenter par :

- BEN SALEM Soumia
- ZERGA Mohamed Oussama

Proposé et encadré par :

- Mr. ESSAID Billel

Juin 2013

ملخص :

إن التنبؤ بالكلام من المجالات المهمة في الذكاء الصناعي،(شبكات العصبونية) عبارة عن وسيلة مساعِدة من اجل التعرف على الكلام حيث اننا نجد أن الطرق الكلاسيكية قد استعملت بكثرة في هذا المجال MLP;TDNN;ATNN. ... الخ, حاليا لدينا شبكات العصبونية و التي اعطت نتائج جيدة مثل MLPفي. هذا المشروع سوف نقدم كادات من اجل التنبؤ بالكلام شبكة العصبونية التي تتميز أساسا بقدرتها على التخزين في الذاكرة.

كلمات البحث: التنبؤ الكلام، مطلب, الشبكات العصبية .

Résumé :

La prédiction de la parole est un des domaines les plus importants dans l'intelligence artificielle. Les réseaux de neurones est un outil d'aide à reconnaître la parole, on en trouve les méthodes classiques qui ont été largement utilisées mais actuellement nous avons les réseaux de neurones qui ont donné de bons résultats tels MLP, TDNN, ATNN, etc. Dans le cadre de ce projet, on présentera comme outil pour la prédiction de la parole, le réseau de neurones MLP caractérisé principalement par sa capacité de mémorisation.

Mots clefs: Prédiction de la parole, Matlab, Réseaux de Neurones.

Abstract :

The prediction of speech is one of the most important areas in artificial intelligence. Neural networks are a tool to recognize the word, we find the traditional methods which have been widely used, but now we have the neural networks that have been successful such MLP, TDNN, ATNN etc.. As part of this project will be presented as a tool for the prediction of speech, the neural network MLP mainly characterized by its storage capacity.

Keywords: Prediction of speech, Matlab Neural Networks.

Sommaire

Résumé

Abstract

ملخص

Table des figures

Introduction générale 1

Chapitre I : Généralités sur les réseaux de neurones artificiels

Introduction 3

I-Historique 4

II-Applications 5

III. Modèle de neurone et réseau 6

III.1 Modèle d'un neurone 6

III.2. Fonctions de transfert 7

III.3. Architecture de réseau 9

III.4. Connexions 11

III.5. Processus d'apprentissage 12

III.5.1. Par correction d'erreur 13

III.5.2. Par la règle de Hebb 14

III.5.3. Compétitif 15

III.5.4. Supervisé 16

III.5.5. Non-Supervisé 17

IV. procédure de développement d'un réseau de neurones 18

IV.1. Collecte des données 18

IV.2. Analyse des données 19

IV.3. Séparation des bases de données 19

IV.4. Choix d'un réseau de neurones 20

IV.5. Mise en forme des données pour un réseau de neurones 20

IV.6.apprentissage du réseau de neurones 20
IV.7. validation 21
Conclusion 21

Chapitre II : Prédiction du signal de parole

Introduction 22
I -définition signal de parole 22
II- le traitement de la parole 23
III -prédiction du signal de parole 24
III-1- définition de prédiction 25
III-2- principe de prédiction par apprentissage 25
III-3-le modèle de prédiction 25
III-4-l'estimation des paramètres 26
Conclusion 27

Chapitre III : Conception & Implémentations du réseau de neurones

Introduction 28
I. présentation de matlab et vhdl 28
I.1. matlab 28
I.2. vhdl 29
II. conception du réseau de neurones 30
II.1. configuration du réseau de neurones 31
II.2. test sans validation croisée 33
II.2.1 4 neurones en couche cachée 33
II.2.2 8 neurones en couche cachée 34
II.2.3 12 neurones en couche cachée 34
II.3 test avec validation croisée 34
II.3.1 4 neurones en couche cachée 35

II.3.2 8 neurones en couche cachée 35
II.3.3 12 neurones en couche cachée 36
II.4 discussion des résultats 36
III. réalisation d'une interface graphique 38
IV. modélisation Simulink de notre réseau de neurones 41
V. génération des codes vhdl et simulation modelSim du réseau de neurones 44
VI. simulation d'une chaine de communication utilisant las réseaux de neurones 47
Conclusion 50
Conclusion générale 51
Bibliographie

Table des figures

Figure I.1 : *Modèle d'un neurone artificiel*.....6
FigureI.2 *: Fonction 'purelin','logsig','tansig'*.....9
Figure I.3 *: Réseau de neurones à S couches* 10
Figure I.4 : *Représentation d'une couche de S neurones* 11
Figure I.5 : *Représentation d'un réseau de trois couches* 11
Figure I.6 : *connexion directe (a) et récurrentes (b)* 12
Figure I.7 : *connexions partielles (a) et totales (b)* 12
Figure I.8 : *Schéma bloc de l'apprentissage supervise* 17
Figure III.1 : *Processus de simulation et d'implémentation d'un code VHDL* 30
Figure III.2 : *Représentation du signal de la parole utilisé* 32
Figure III.3 : *signal original et signal prédit*..... 37
FigureIII.4 :*Signal d'erreur de prédiction*..... 38
Figure III.5 : *L'interface graphique de test du réseau de neurones* 38
Figure III.6 : *Chargement du fichier audio* 39
Figure III.7 : *Décomposition du signal audio* 40
Figure III.8 : *Déroulement du processus d'entraînement du réseau* 40
Figure III.9 : *le bloc du réseau de neurones crée en Simulink* 41
Figure III.10 : *représentation du réseau de neurones couches* 41
Figure III.11 : *la couche cachée*..... 42
Figure III.12 : *les 12 neurones en couche cachée*..... 42
Figure III.13 : *la couche de sortie* 43
Figure III.14 : *le modèle Simulink*..... 43
Figure III.15 : *le signal original et le signal prédit* 44
Figure III.16 : *Vérification de compatibilité VHDL* 45
Figure III.17 : *Générations des codes VHDL*..... 45
Figure III.18 : *Génération du testbench*..... 46

Figure III.19 : *Lancement du simulateur ModelSim par Matlab* **46**
Figure III.20 : *Les résultats de simulation ModelSim* **47**
Figure III.21 : *Le modèle Simulink de la chaine de communication* **47**
Figure III.22 : *La représentation temporelle du signal original et du signal bruité* **48**
Figure III.23 : *Le signal prédit* **49**
Figure III.24 : *Le signal original et le signal reçu* **50**

Liste des Tableaux

Tableau I.1 : *Fonctions de transfert a = f(n)* **8**
Tableau III.1 : *test de 4 neurone en couche cachée* **33**
Tableau III.2 : *test de 8 neurone en couche cachée* **34**
Tableau III.3 : *test de 12 neurone en couche cachée* **34**
Tableau III.4 : *test de 4 neurone en couche cachée* **35**
Tableau III.5 : *test de 8 neurone en couche cachée* **35**
Tableau III.6 : *test de 12 neurone en couche cachée* **36**

Introduction Générale

Ce projet porte sur la proposition de modèles pour ce que l'on appelle aujourd'hui la prédiction de signal de parole. Il s'agit d'un problème important aujourd'hui dans des domaines aussi divers que la bioinformatique, l'extraction d'information, la reconnaissance de la parole.

Les réseaux de neurones artificiels sont des réseaux fortement connectés de Processeurs élémentaires fonctionnant en parallèle. Chaque processeur élémentaire calcule une sortie unique sur la base des informations qu'il reçoit.

Nous utilisons ici ces réseaux de neurones dans un but tout à fait différent. Nous désirons les appliquer à la prédiction du signal de parole. Nous ne disposons d'aucune information quant au dimensionnement de ce réseau et devrons donc explorer l'ensemble des possibilités de manière à affiner notre choix de configuration de réseau. Le nombre de réseaux par unités acoustiques – on entend par unité acoustique un ensemble d'échantillons consécutifs - devra être également déterminé de manière à obtenir une prédiction efficace. Si les unités ne se résument pas au signal complet, il faudra alors envisager une adaptation dynamique des paramètres du réseau pour coder le signal complet. Ces deux étapes seront réalisées par un ensemble d'essais successifs. Nous possèderons alors un ou plusieurs types de configurations de réseaux possibles pour une unité acoustique bien définie.

Ce mémoire est décomposé en trois chapitres :

- Le premier chapitre décrit les notions de base sur les réseaux de neurones artificiels
- Dans le deuxième chapitre, nous avons présenté la procédure de la prédiction du signal, leur utilisation par les réseaux de neurones.
- Le dernier chapitre, décrit notre conception, avec les résultats de simulation obtenus.

Chapitre 1 :

Généralités sur les réseaux de neurones artificiels

INTRODUCTION

Les réseaux de neurones, fabriqués de structures cellulaires artificielles, constituent une approche permettant d'aborder sous des angles nouveaux les problèmes de perception, de mémoire, d'apprentissage et de raisonnement. Ils s'avèrent aussi des alternatives très prometteuses pour contourner certaines des limitations des ordinateurs classiques.

Grace à leur traitement parallèle de l'information et à leurs mécanismes inspires des cellules nerveuses (neurones), ils infèrent des propriétés émergentes permettant de solutionner des problèmes jadis qualifies de complexes.

Nous aborderons dans ce chapitre les principales architectures de réseaux de neurones que l'on retrouve dans la littérature. Il ne s'agit pas de les étudier toutes, car elles sont trop nombreuses, mais plutôt d'en comprendre les mécanismes internes fondamentaux et de savoir comment et quand les utiliser.

Le cerveau humain contient environ 100 milliards de neurones. Ces neurones nous permettent, entre autre, de lire ce texte tout en maintenant une respiration régulière permettant d'oxygéner notre sang, en actionnant notre corps qui assure une circulation efficace de ce sang pour nourrir nos cellules, etc.

Chacun de ces neurones est par ailleurs fort complexe. Essentiellement, il s'agit de tissu vivant et de chimie. Les spécialistes des neurones biologiques (ceux qui œuvrent en neurophysiologie) commencent à peine à comprendre quelques uns de leurs mécanismes internes. On croit en général que leurs différentes fonctions neuronales, y compris celle de la mémoire, sont stockées au niveau des connexions (synapses) entre les neurones. C'est ce genre de théorie1 qui a inspiré la plupart des architectures de réseaux de neurones artificiels.

L'apprentissage consiste alors soit à établir de nouvelles connexions, soit à en modifier des existantes.

Ceci nous amène à poser une question fondamentale : en ce basant sur nos connaissances actuelles, *peut-on construire des modèles approximatifs de neurones et les entrainer pour, éventuellement, réaliser des taches utiles ?*

La réponse courte **:** est «oui», même si les réseaux que nous allons développer ne possèdent qu'une infime fraction de la puissance du cerveau humain. [1]

I. HISTORIQUE

L'histoire des réseaux de neurones est donc tissée à travers des découvertes conceptuelles et des développements technologiques survenus à diverses époques.

Les premières recherches remontent à la fin du 19^{e} et au début du 20^{e} siècle. Ils consistent en de travaux multidisciplinaires en physique, en psychologie et en neurophysiologie par des scientifiques tels Hermann Von Helmholtz, Ernst Mach et Ivan Pavlov. À cette époque, il s'agissait de théories plutôt générales sans modèle mathématique précis d'un neurone.

On s'entend pour dire que la naissance du domaine des réseaux de neurones artificiels remonte aux années 1940 avec les travaux de Warren McCulloch et Walter Pitts qui ont montre qu'avec de tels réseaux, on pouvait, en principe, calculer n'importe quelle fonction arithmétique ou logique c'est que l'on appel Le neurone formel de Automates booléens et Connexions fixes .Vers la fin des années 1940, Donald Hebb a ensuite proposé une théorie fondamentale pour l'apprentissage.

La première application concrète des réseaux de neurones artificiels est survenue vers la fin des années 1950 avec l'invention du réseau dit «perceptron» par un dénomme Frank Rosenblatt. Rosenblatt et ses collègues ont construit un réseau et démontre ses habilités à reconnaitre des formes par Modèle linéaire à seuil et avec des Connexions modifiables.[1]

Environ au même moment, Bernard Widrow et Ted Hoff ont proposé un nouvel algorithme d'apprentissage pour entrainer un réseau adaptatif de neurones linéaires, dont la structure et les capacités sont similaires au perceptron par Modèle linéaire, des Connexions modifiables et avec Une seule couche.

En1967, Amari propose une théorie de Champs de neurone qui baser sur Continuum neural et Motifs d'activation. Grossberg construit un réseau consisté sur la Stabilité/Plasticité du Paramètre de vigilance et le Réentrance en 1968.

✓ ***Début des Années 1970 à aujourd'hui :***

Perceptrons (Minsky & Papert, 1969)

• Problème du OU EXCLUSIF

• Coup dur pour la recherche

Auto-organisation (Kohonen, 1970)

• Apprentissage non supervisé.

• Topologie dans le réseau.

Mémoires auto-associatives (Hopfield, 1982)

• Physique statistique.

• Apprentissage par cœur.

Perceptrons multi-couches

• Rétro-propagation du gradient.

• Werbos 1974, Parker 1982, Le Cun 1985, Rumelhart & McClelland 1986.[2]

II. APPLICATIONS

Les réseaux de neurones servent aujourd'hui à toutes sortes d'applications dans divers domaines.

Par exemple, on a développé un auto-pilote pour avion, ou encore un système de guidage pour automobile, on a conçu des systèmes de lecture automatique de chèques bancaires et d'adresses postales, on produit des systèmes de traitement du signal pour différentes applications militaires, un système pour **la synthèse de la parole**, des réseaux sont aussi utilisés pour bâtir des systèmes de vision par ordinateur, pour faire des prévisions sur les marches monétaires, pour évaluer le risque financier ou en assurance, pour différents processus manufacturiers, pour le diagnostic médical, pour l'exploration pétrolière ou gazière, en robotique, en télécommunication, ! Bref, les réseaux de neurones ont aujourd'hui un impact considérable et, il y a fort à parier, que leur importance ira grandissant dans le futur. [1]

III. MODELE DE NEURONE ET RESEAU

Nous présentons le modèle mathématique pour d´écrire, d'une part, un neurone artificiel et, d'autre part, un réseau de neurones complet, c'est à dire un ensemble de neurones reliés en réseau. Le modèle que nous présentons dans ce chapitre est celui de base, commun à beaucoup d'architectures. Il n'est cependant pas universel.[1]

III.1. Modèle d'un neurone

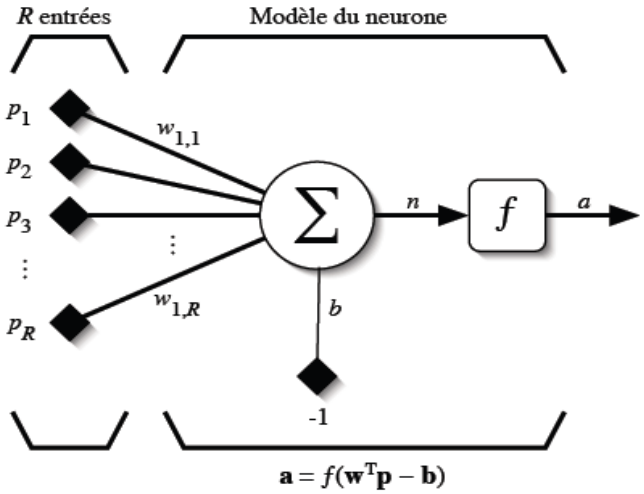


Figure I.1 : Modèle d'un neurone artificiel

Le modèle mathématique d'un neurone artificiel est illustré à la figure I.1. Un neurone est essentiellement constitue d'un intégrateur qui effectue la somme pondérée de ses entrées. Le résultat ***n*** de cette somme est ensuite transformée par une fonction de transfert ***f*** qui produit la sortie ***a*** du neurone. Les ***R*** entrées des neurones correspondent au vecteur $p = [p_1 \quad p_2 \cdots \quad p_R]^T$, alors que $w = [w_{1,1} \quad w_{1,2} \cdots \quad w_{1,R}]^T$ représente le vecteur des poids du neurone. La sortie *n* de l'intégrateur est donnée par l'équation suivante :

$$n = \sum_{j=1}^{R} w_{1j}p_j - b = w_{1,1}p_1 + w_{1,2}p_2 + \cdots + w_{1,R}p_R - b$$

Que l'on peut aussi écrire sous forme matricielle :

$$n = w^T p - b$$

Cette sortie correspond a une somme pondérée des poids et des entrées moins ce qu'on nomme le biais ***b*** du neurone. Le résultat ***n*** de la somme pondérée s'appelle le niveau

d'activation du neurone. Le biais ***b*** s'appelle aussi le seuil d'activation du neurone. Lorsque le niveau d'activation atteint ou dépasse le seuil ***b***, alors l'argument de ***f*** devient positif (ou nul). Sinon, il est négatif. [3]

III.2. Fonctions de transfert

Jusqu'à présent, nous n'avons pas spécifie la nature de la fonction d'activation de notre modèle.

Il se trouve que plusieurs possibilités existent. Différentes fonctions de transfert pouvant être utilisées comme fonction d'activation du neurone sont énumérées au tableau I.1. Les trois les plus utilisées sont les fonctions «seuil» (en anglais «hard limit»), «linéaire» et «sigmoïde».

Comme son nom l'indique, la fonction seuil applique un seuil sur son entrée. Plus précisément, une entrée négative ne passe pas le seuil, la fonction retourne alors la valeur 0 (on peut interpréter ce 0 comme signifiant *faux*), alors qu'une entrée positive ou nulle dépasse le seuil, et la fonction retourne 1 (vrai).

Nom de la fonction	Relation d'entrée/sortie	Icone	Nom Matlab
seuil	**a = 0 si n < 0** **a = 1 si n≥0**		**hardlim**
seuil symétrique	**a = −1 si n < 0** **a = 1 si n ≥ 0**		**hardlims**
linéaire	**a = n**		**purelin**
linéaire saturée	**a = 0 si n < 0** **a = n si 0 ≤ n≤1** **a = 1 si n > 1**		**satlin**
linéaire saturée symétrique	**a = −1 si n < −1** **a = n si −1 ≤ n ≤ 1** **a = 1 si n > 1**		**satlins**
linéaire positive	**a = 0 si n < 0** **a = n si n ≥ 0**		**poslin**
sigmoïde	**a =** $\frac{1}{1+\exp^{-n}}$		**logsig**
tangente hyperbolique	**a =** $\frac{e^{n}-e^{-n}}{e^{n}+e^{-n}}$		**tansig**
compétitive	**a = 1 si n** maximum **a = 0** autrement		**compte**

Tableau I.1 : Fonctions de transfert a = f(n)

Nous avons utilisé dans notre projet les fonctions qui sont coloré dans le tableau I.1.les graphes de ces fonctions sont représenté dans la figure suivante : [4]

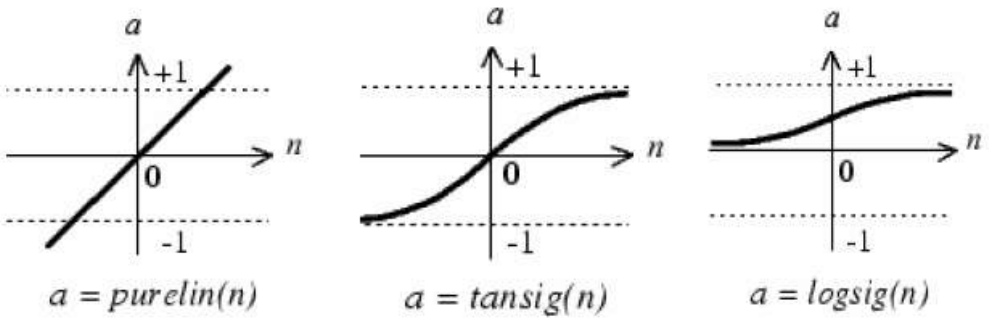


Figure I.2 : Fonctions 'purelin','tansig' et 'logsig'

III.3. Architecture de réseau

La plupart des réseaux de neurones ont une topologie définie sous forme de couches. Il existe quelques exceptions lorsque le réseau n'est pas explicitement défini sur plusieurs couches (comme par exemple certaines mémoires associatives), mais elles peuvent alors être considérées comme n'ayant qu'une seule couche. L'architecture du réseau peut alors être décrite par le nombre de couches et le nombre de neurones dans chaque couche.

Une terminologie similaire à celle présentée pour définir les différents types de neurones est utilisée pour définir les couches d'un réseau multicouche (figure I.3) :

- couche d'entrée pour une couche constituée de cellules perceptives,
- couche cachée pour une couche constituée de neurones cachés,
- couche de sortie pour une couche constituée de neurones de sorties.

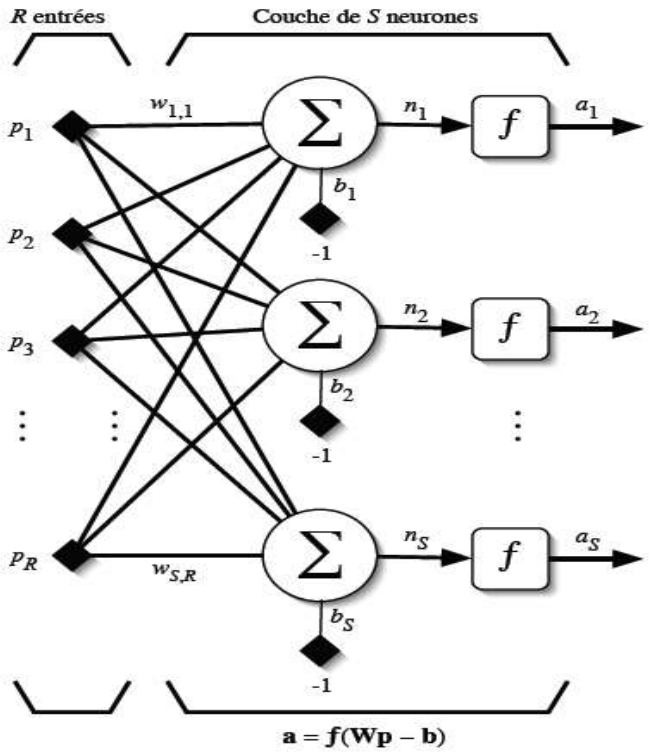


Figure I.3 : Réseau de neurones à S couches

Les réseaux multicouches sont beaucoup plus puissants que les réseaux simples à une seule couche. En utilisant deux couches (une couche cachée et une couche de sortie), a condition d'employer une fonction d'activation sigmoïde sur la couche cachée, on peut entrainer un réseau à produire une approximation de la plupart des fonctions, avec une précision arbitraire (cela peut cependant requérir un grand nombre de neurones sur la couche cachée). Sauf dans de rares cas, les réseaux de neurones artificiels exploitent deux ou trois couches.

Entrainer un réseau de neurones signifie modifier la valeur de ses poids et de ses biais pour qu'il réalise la fonction entrée/sortie désirée. [5]

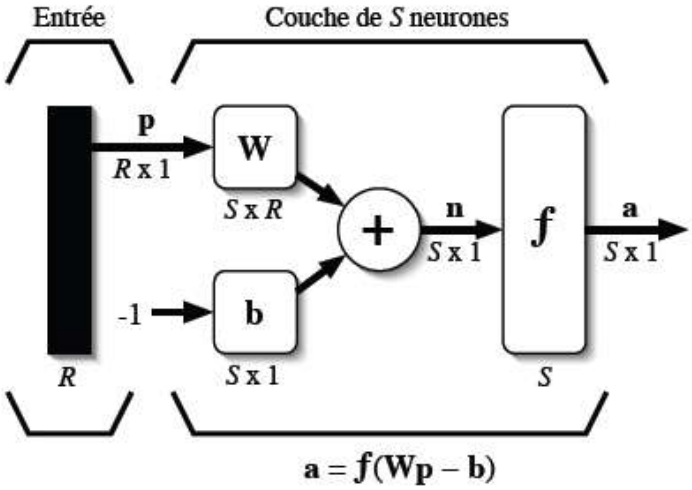


Figure I.4 : Représentation d'une couche de S neurones

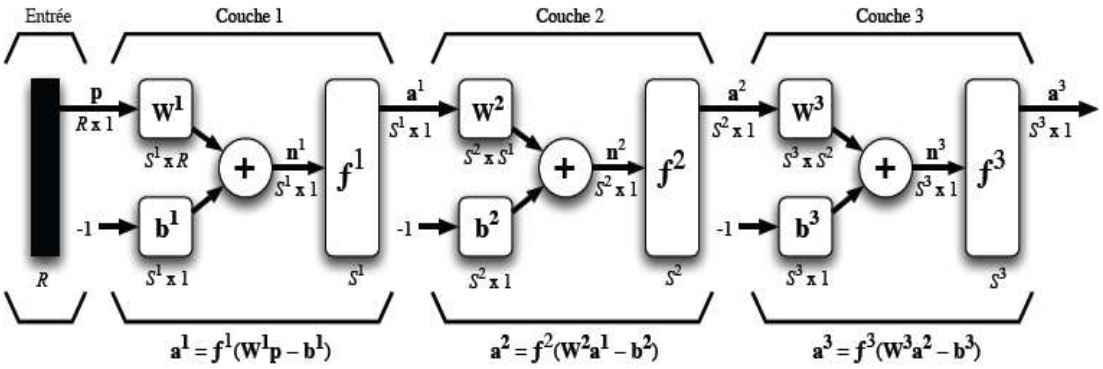


Figure I.5 : Représentation d'un réseau de trois couches

III.4. Connexions

Le modèle de connexion définit la manière dont sont interconnectés les neurones d'un réseau. En se basant sur la structure à couches, on distingue différents types de connexions : les connexions inter couches (interconnexion entre neurones de couches voisine), les connexions supracouche (lorsque les couches ne sont pas adjacentes), les connexions intracouches (connexions entre neurones d'une même couche) et l'autoconnexion (un neurone avec lui-même).

De manière générale, le sens de transfert de l'information dans un réseau est défini par la nature des connexions: directes ou récurrentes (figure I.6). Les connexions directes sont celles qui sont dirigées d'une couche d'indice inférieur vers une couche d'indice supérieur. Les connexions sont dites récurrentes lorsque des

sorties de neurones d'une couche sont connectées aux entrées d'une couche d'indice inférieur.



Figure I.6 : connexion directe (a) et récurrentes (b)

Par ailleurs, entre deux couches, les connexions peuvent être partielles ou totales (figure1-7). L'utilisation de connexions partielles permet de regrouper certaines zones du réseau pour effectuer une fonction spécifique. [6]

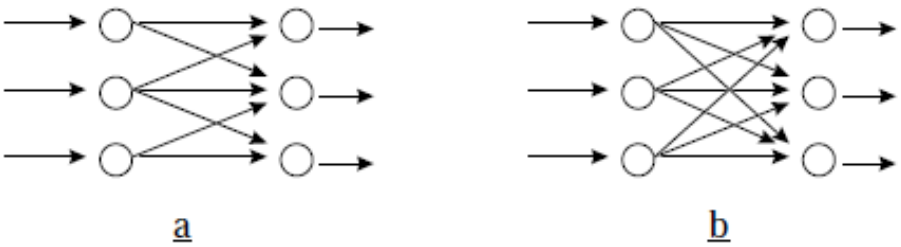


Figure I.7 : connexions partielles (a) et totales (b)

III.5. Processus d'apprentissage

Parmi les propriétés désirables pour un réseau de neurones, la plus fondamentale est surement la capacité d'apprendre de son environnement, d'améliorer sa performance à travers un processus d'apprentissage. *Mais qu'est-ce donc que l'apprentissage ?* Malheureusement, il n'existe pas de définition générale, universellement acceptée, car ce concept touche à trop de notions distinctes qui dépendent du point de vue que l'on adopte.

Dans le contexte des réseaux de neurones artificiels, nous adopterons un point de vue pragmatique en proposant la définition suivante : *« L'apprentissage est un processus dynamique et itératif permettant de modifier les paramètres d'un réseau en réaction avec les stimuli qu'il reçoit de son environnement. Le type d'apprentissage est déterminé par la manière dont les changements de paramètre surviennent. »*

Dans la plupart des architectures, l'apprentissage se traduit par une modification de l'efficacité synaptique, c'est à dire par un changement dans la valeur des poids qui relient les neurones d'une couche à l'autre. Soit le poids $w_{i,j}$ reliant le neurone $\boldsymbol{i}$ a son entrée $\boldsymbol{j}$. Au temps $\boldsymbol{t}$, un changement $\Delta w_{i,j}(t)$ de poids peut s'exprimer simplement de la façon suivante :

$$\Delta w_{i,j}(t) = w_{i,j}(t+1) - w_{i,j}(t),$$

Et, par conséquent, $w_{i,j}(t+1) = w_{i,j}(t) + \Delta w_{i,j}(t)$, avec $w_{i,j}(t+1)$ et $w_{i,j}(t)$ représentant respectivement les nouvelle et ancienne valeurs du poids $w_{i,j}(t)$.

Nous allons passer en revue différents types de règles ainsi que différents principes pouvant guider l'apprentissage d'un réseau de neurone.

III.5.1. Par correction d'erreur

La première règle que l'on peut utiliser est fondée sur la correction de l'erreur observée en sortie. Soit $a_i(t)$ la sortie que l'on obtient pour le neurone i au temps t. Cette sortie résulte d'un stimulus $p(t)$ que l'on applique aux entrées du réseau dont un des neurones correspond au neurone i. Soit $d_i(t)$ la sortie que l'on désire obtenir pour ce même neurone i au temps t. Alors, $a_i(t)$ et $d_i(t)$ seront généralement différents et il est naturel de calculer l'erreur $e_i(t)$ entre ce qu'on obtient et ce qu'on voudrait obtenir :

$$e_i(t) = a_i(t) - d_i(t)$$

Avec $e_i(t) = [e_1(t)e_2(t) \dots e_i(t) \dots e_s(t)]$ qui désigne le vecteur des erreurs observées sur les S neurones de sortie du réseau. L'apprentissage par correction des erreurs consiste à minimiser un indice de performance F basé sur les signaux d'érreur $e_i(t)$, dans le but de faire converger les sorties du réseau avec ce qu'on voudrait qu'elles soient. Un critère très populaire est la somme des erreurs quadratiques :

$$F\big(e_i(t)\big) = \sum_{i=1}^{s} e_i^2 = e(t)'e(t)$$

Pour minimiser $F(e_i(t)) = F(w(t)) = F(t)$ nous allons commencer par choisir des poids initiaux ($t = 0$) au hasard, puis nous allons modifier ces poids de la manière suivante :

$$w_{i,j}(t+1) = w_{i,j}(t) + \tau\, x(t)$$

Où le vecteur $x(t)$ désigne la direction dans laquelle nous allons chercher le minimum et τ est une constante positive déterminant l'amplitude du pas dans cette direction (la vitesse d'apprentissage).

La règle de la correction des erreurs est utilisée pour beaucoup de réseaux de neurones artificiels, bien qu'elle ne soit pas plausible biologiquement. En effet, comment le cerveau pourrait-il connaitre a priori les sorties qu'il doit produire ? Cette règle ne peut être utilisée que dans un contexte d'apprentissage supervise sur lequel nous reviendrons bientôt.

III.5.2. Par la règle de Hebb

Une telle synapse est dite «synapse hebbien». Il utilise un mécanisme interactif, dépendant du temps et de l'espace, pour augmenter l'efficacité synaptique d'une manière proportionnelle a la corrélation des activités pré- et post-synaptiques. De cette définition ressort les propriétés suivantes :

1. ***Dépendance temporelle***. Les modifications d'une synapse hebbien dépendent du moment exact des activités pré- et post-synaptiques ;

2. ***Dépendance spatiale*.** Etant donne la nature même de la synapse qui constitue un lieu de transmission d'information, l'apprentissage hebbien se doit de posséder une contiguïté spatiale. C'est cette propriété qui, entre autres, permet l'apprentissage dit non-supervise sur lequel nous reviendrons bientôt ;

3. ***Interaction***. L'apprentissage hebbien dépend d'une interaction entre les activités de part et d'autre de la synapse.

4. ***Conjonction ou corrélation***. Une interprétation de l'énonce de Hebb est que la condition permettant un changement dans l'efficacité synaptique est une conjonction des activités pré –et post-synaptiques. C'est la co-occurrente des activités de part et d'autre de la synapse qui engendre une modification de celui-ci. Une interprétation

plus statistique réfère à la corrélation de ces activités. Deux activités positives simultanées (corrélation positive) engendrent une augmentation de l'efficacité synaptique, alors que l'absence d'une telle corrélation engendre une baisse de cette efficacité.

On peut exprimer la règle de Hebb sous sa forme la plus simple par la formule suivante :

$$\Delta w_j(t-1) = \tau\, p_j(t) a(t),$$

Ou τ est une constante positive qui détermine la vitesse de l'apprentissage, $p_j(t)$ correspond à l'activité pré-synaptique (l'entrée j du neurone) au temps t, et $a(t)$ à l'activité post-synaptique (sortie du neurone) à ce même temps$t$. Cette formule fait ressortir explicitement la corrélation entre le signal qui entre et celui qui sort.

Et si l'on fixe $a = \tau$ pour simplifier (on pose un rythme d'apprentissage égale a celui de l'oubli), on obtient la règle dite «instar» :

$$\Delta w(t-1) = \tau\, a(t)[p(t) - w(t-1)],$$

Une façon d'aborder cette règle, est de regarder ce qui se passe lorsque $a(t) = 1$:

$$w(t) = w(t-1) + \tau[p(t) - w(t-1)],$$

$$= (1-\tau)w(t-1) + \tau\, p(t).$$

Une propriété particulièrement intéressante de la règle instar est qu'avec des entrées normalisées, suite au processus d'apprentissage, les poids $\boldsymbol{w}$ convergeront également vers des vecteurs normalises.

III.5.3. Compétitif

L'apprentissage compétitif, comme son nom l'indique, consiste à faire competitionner les neurones d'un reseau pour déterminer lequel sera actif à un instant donné. Contrairement aux autres types d'apprentissage où, généralement, tous les neurones peuvent apprendre simultanément et de la même manière, l'apprentissage compétitif produit un «vainqueur» ainsi que, parfois, un ensemble de neurones

«voisins» du vainqueur, et seuls ce vainqueur et, potentiellement, son voisinage bénéficient d'une adaptation de leur poids.

On dit alors que l'apprentissage est local car limite a un sous-ensemble des neurones du reseau.

Une règle d'apprentissage compétitif comporte les éléments suivants :

- Un ensemble de neurones identiques (même type) sauf pour les valeurs de leurs poids synaptiques;
- Une limite imposée à la «force» d'un neurone ;
- Un mécanisme permettant aux neurones de competitionner pour le droit de répondre a un certain sous-ensemble des stimuli d'entrée, de manière a ce qu'un seul neurone de sortie soit actif à la fois.

III.5.4. Supervisé

L'apprentissage dit supervise est caractérisé par la présence d'un «professeur» qui possède une connaissance approfondie de l'environnement dans lequel évolue le reseau de neurones.

En pratique, les connaissances de ce professeur prennent la forme d'un ensemble de *Q* couples de vecteurs d'entrée et de sortie que nous noterons *{(p1 ;d1),(p2,d2),..,(pQ,dQ)},* ou pi désigne un stimulus (entrée) et *di* la cible pour ce stimulus, c'est-`a-dire les sorties désirées du reseau.

Chaque couple *(pi, di)* correspond donc à un cas d'espèce de ce que le reseau devrait produire (la cible) pour un stimulus donne. Pour cette raison, l'apprentissage supervisé est aussi qualifié d'apprentissage par des exemples.

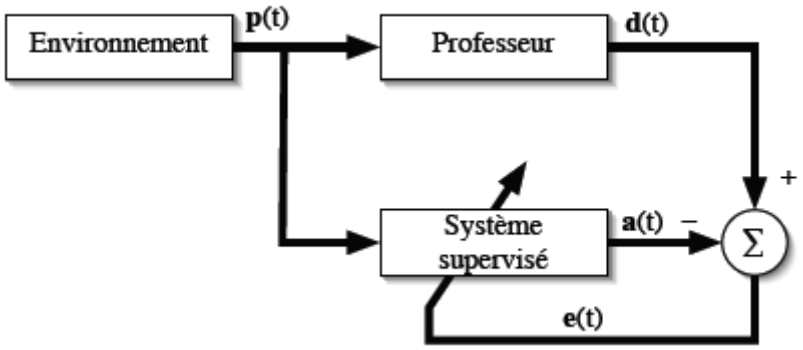


Figure I.8 : Schéma bloc de l'apprentissage supervise

L'apprentissage supervisé est illustré d'une manière conceptuelle a la figure I.8 L'environnement est inconnu du reseau. Celui-ci produit un stimulus *p* qui est acheminé à la fois au professeur et au reseau. Grâce à ses connaissances intrinsèques, le professeur produit une sortie désirée *d(t)* pour ce stimulus. On suppose que cette réponse est optimale. Elle est ensuite comparée (par soustraction) avec la sortie du reseau pour produire un signal d'érreur *e(t)* qui est réinjecté dans le réseau pour modifier son comportement via une procédure itérative qui, éventuellement, lui permet de simuler la réponse du professeur. Autrement dit, la connaissance de l'environnement par le professeur est graduellement transférée vers le réseau jusqu'à l'atteinte d'un certain critère d'arrêt.

III.5.5. Non-supervise

La dernière forme d'apprentissage que nous abordons est dite «non-supervisée» ou encore «auto-organisée». Elle est caractérisée par l'absence complète de professeur, c'est à dire qu'on ne dispose pas d'un signal d'érreur, comme dans le cas supervisé. Nous ne disposons donc que d'un environnement qui fournit des stimuli, et d'un reseau qui doit apprendre sans intervention externe. En assimilant les stimuli de l'environnement à une description de son état interne, la tâche du réseau est alors de modéliser cet état le mieux possible. Pour y arriver, il importe d'abord de définir une mesure de la qualité pour ce modèle, et de s'en servir par la suite pour optimiser les paramètres libres du réseau, c'est à dire ses poids synaptiques. A la fin de l'apprentissage, le réseau a développé une habilité à former des représentations

internes des stimuli de l'environnement permettant d'encoder les caractéristiques de ceux-ci et, par conséquent, de créer automatiquement des classes de stimuli similaires.

L'apprentissage non-supervise s'appuie généralement sur un processus compétitif permettant d'engendrer un modèle ou les poids synaptiques des neurones représentent des prototypes de stimuli. La qualité du modèle résultant doit s'évaluer a l'aide d'une métrique permettant de mesurer la distance entre les stimuli et leurs prototypes.

IV.PROCEDURE DE DEVELOPPEMENT D'UN RESEAU DE NEURONES.

Le cycle classique de développement peut être séparé en sept étapes :[6]

1. la collecte des données,
2. l'analyse des données,
3. la séparation des bases de données,
4. le choix d'un réseau de neurones,
5. la mise en forme des données,
6. l'apprentissage,
7. la validation.

IV.1. Collecte des données:

L'objectif de cette étape est de recueillir des données, à la fois pour développer le réseau de neurones et pour le tester. Dans le cas d'applications sur des données réelles, l'objectif est de rassembler un nombre de données suffisant pour constituer une base représentative des données susceptibles d'intervenir en phase d'utilisation du système neuronal.

La fonction réalisée résultant d'un calcul statistique, le modèle qu'il constitue n'a de validité que dans le domaine où on l'a ajusté. En d'autres termes, la présentation de données très différentes de celles qui ont été utilisées lors de l'apprentissage peut entraîner une sortie totalement imprévisible.

IV.2. Analyse des données

Il est souvent préférable d'effectuer une analyse des données de manière à déterminer les caractéristiques discriminantes pour détecter ou différencier ces données. Ces caractéristiques constituent l'entrée du réseau de neurones. Notons que cette étude n'est pas spécifique aux réseaux de neurones, quelque soit la méthode de détection ou de classification utilisée, il est généralement nécessaire de présenter des caractéristiques représentatives. Cette détermination des caractéristiques a des conséquences à la fois sur la taille du réseau (et donc le temps de simulation), sur les performances du système (pouvoir de séparation, taux de détection), et sur le temps de développement (temps d'apprentissage).

Une étude statistique sur les données peut permettre d'écarter celles qui sont aberrantes et redondantes. Dans le cas d'un problème de classification, il appartient à l'expérimentateur de déterminer le nombre de classes auxquelles ses données appartiennent et de déterminer pour chaque donnée la classe à laquelle elle appartient.

IV.3. Séparation des bases de données

Afin de développer une application à base de réseaux de neurones, il est nécessaire de disposer de deux bases de données : une base pour effectuer l'apprentissage et une autre pour tester le réseau obtenu et déterminer ses performances. Afin de contrôler la phase d'apprentissage, il est souvent préférable de posséder une troisième base de données appelée « base de validation croisée ». Les avantages liés à l'utilisation de cette troisième base de données seront présentés dans le chapitre III.

Il n'y a pas de règle pour déterminer ce partage de manière quantitatif. Il résulte souvent d'un compromis tenant compte du nombre de données dont on dispose et du temps imparti pour effectuer l'apprentissage. Chaque base doit cependant satisfaire aux contraintes de représentativité de chaque classe de données et doit généralement refléter la distribution réelle, c'est à dire la probabilité d'occurrence des diverses classes.

IV.4. Choix d'un réseau de neurones

Il existe un grand nombre de types de réseaux de neurones, avec pour chacun des avantages et des inconvénients. Le choix d'un réseau peut dépendre :

- de la tâche à effectuer (classification, association, contrôle de processus, séparation aveugle de sources...).
- de la nature des données (dans notre cas, des données présentant des variations au cours du temps),
- d'éventuelles contraintes d'utilisation temps-réel (certains types de réseaux de neurones, tels que la 'machine de Boltzmann' . nécessitant des tirages aléatoires et un nombre de cycles de calculs indéfini avant stabilisation du résultat en sortie, présentent plus de contraintes que d'autres réseaux pour une utilisation temps-réel),

Ce choix est aussi fonction de la maîtrise ou de la connaissance que l'on a de certains réseaux, ou encore du temps dont on dispose pour tester une architecture prétendu plus performante.

IV.5. Mise en forme des données pour un réseau de neurones

De manière générale, les bases de données doivent subir un prétraitement afin d'être adaptées aux entrées et sorties du réseau de neurones. Un prétraitement courant consiste à effectuer une normalisation appropriée, qui tienne compte de l'amplitude des valeurs acceptées par le réseau.

IV.6. Apprentissage du réseau de neurones

Tous les modèles de réseaux de neurones requièrent un apprentissage. Plusieurs types d'apprentissages peuvent être adaptés à un même type de réseau de neurones. Les critères de choix sont souvent la rapidité de convergence ou les performances de généralisation.

Le critère d'arrêt de l'apprentissage est souvent calculé à partir d'une fonction de coût, caractérisant l'écart entre les valeurs de sortie obtenues et les valeurs de références. La technique de validation croisée, qui sera précisée par la suite, permet un

arrêt adéquat de l'apprentissage pour obtenir de bonnes performances de généralisation.

Certains algorithmes d'apprentissage se chargent de la détermination des paramètres architecturaux du réseau de neurones. Si on n'utilise pas ces techniques, l'obtention des paramètres architecturaux optimaux se fera par comparaison des performances obtenues pour différentes architectures de réseaux de neurones.

IV.7. Validation

Une fois le réseau de neurones entraîné (après apprentissage), il est nécessaire de le tester sur une base de données différente de celles utilisées pour l'apprentissage ou la validation croisée. Ce test permet à la fois d'apprécier les performances du système neuronal et de détecter le type de données qui pose problème. Si les performances ne sont pas satisfaisantes, il faudra soit modifier l'architecture du réseau, soit modifier la base d'apprentissage (caractéristiques discriminantes ou représentativité des données de chaque classe). [7]

CONCLUSION

Dans ce chapitre, nous avons présenté quelques notions de réseaux de neurones artificiels, afin de clarifier l'utilité de ces derniers dans différentes domaines d'applications.

Chapitre : 2

Prédiction du signal de parole

INTRODUCTION

La prédiction de séries temporelles est un problème très important, dont la solution trouve de nombreuses applications dans des domaines aussi variés que la prise de décision financière ou économique, la météorologie, la prévision de ventes .Dans la mise en œuvre des méthodes, on considère que le signal est numérisé et qu'il est représenté par une série temporelle. Ainsi, le problème de la prédiction en temps réel revient à estimer, à chaque instant t, les valeurs futures de la série temporelle, à partir de ses valeurs passées.

Il existe de nombreuses méthodes permettant l'apprentissage et la prédiction de séries temporelles. Les méthodes classiques utilisent un modèle d'approximation paramétré. Ces modèles ont une structure simple et linéaire. Toutefois, ils ne suffisent pas pour réaliser une prédiction correcte de séries temporelles à comportement complexe et non-linéaire.

Des approches plus récentes, basées sur les techniques de « soft computing » (réseaux neuronaux, contrôleurs à logique floue), ont été développées avec succès. Ces outils apparaissent, en effet, comme des approximateurs universels. De plus, ils peuvent opérer dans un environnement variable et ont donc d'excellentes capacités d'adaptation. Une difficulté subsiste toutefois : les temps de calculs nécessaires à la recherche de la solution optimale ne permettent pas toujours d'opérer en temps réel. [8]

I -DEFINITION SIGNAL DE PAROLE

L'information portée par le signal de parole peut être analysée de bien des façons. On en distingue généralement plusieurs niveaux de description non exclusifs : *acoustique, phonétique, phonologique, morphologique, syntaxique, sémantique,* et *pragmatique*. [9]

II- LE TRAITEMENT DE LA PAROLE

Le traitement de la parole est aujourd'hui une composante fondamentale des sciences de l'ingénieur. Située au croisement du traitement du signal numérique et du traitement du langage (c'est-à-dire du traitement de données symboliques), cette discipline scientifique a connu depuis les années 60 une expansion fulgurante, liée au développement des moyens et des techniques de télécommunications.

L'importance particulière du traitement de la parole dans ce cadre plus général s'explique par la position privilégiée de la parole comme vecteur d'information dans notre société humaine. L'extraordinaire singularité de cette science, qui la différencie fondamentalement des autres composantes du traitement de l'information, tient sans aucun doute au rôle fascinant que joue le cerveau humain à la fois dans la production et dans la compréhension de la parole et à l'étendue des fonctions qu'il met, inconsciemment, en œuvre pour y parvenir de façon pratiquement instantanée.

Les techniques modernes de traitement de la parole tendent cependant à produire des systèmes automatiques qui se substituent à l'une ou l'autre de ces fonctions :

- **Les *analyseurs*** de parole sont utilisés soit comme composant de base de systèmes de codage, de reconnaissance ou de synthèse, soit en tant que tels pour des applications spécialisées, comme l'aide au diagnostic médical ou l'étude des langues.

- Les ***reconnaisseurs*** ont pour mission de décoder l'information portée par le signal vocal à partir des données fournies par l'analyse. On distingue fondamentalement deux types de reconnaissance, en fonction de l'information que l'on cherche à extraire du signal vocal : la ***reconnaissance du locuteur***, dont l'objectif est de reconnaître la personne qui parle, et la ***reconnaissance de la parole***, où l'on s'attache plutôt à reconnaître ce qui est dit. On classe

également les reconnaisseurs en fonction des hypothèses simplificatrices sous lesquelles ils sont appelés à fonctionner.

- Les ***synthétiseurs*** ont quant à eux la fonction inverse de celle des analyseurs et des reconnaisseurs de parole : ils produisent de la parole artificielle. On distingue fondamentalement deux types de synthétiseurs : les ***synthétiseurs de parole à partir d'une représentation numérique***, inverses des analyseurs, dont la mission est de produire de la parole à partir des caractéristiques numériques d'un signal vocal telles qu'obtenues par analyse, et les ***synthétiseurs de parole à partir d'une représentation symbolique***¸ inverse des reconnaisseurs de parole et capables en principe de prononcer n'importe quelle phrase sans qu'il soit nécessaire de la faire prononcer par un locuteur humain au préalable.[9]

III -PREDICTION DU SIGNAL DE PAROLE

Dans le domaine scientifique, une prédiction s'appuie sur un modèle et sur des observations. Pour être valide, une prédiction doit donc être accompagnée du modèle qui la sous-tend. Ce modèle doit être accessible et discutable, afin d'être éventuellement affiné pour améliorer la prédiction. Les erreurs de prédiction, c'est-à-dire les écarts entre les prédictions du modèle et les observations expérimentales, jouent un rôle fondamental pour les tâches de validation et d'amélioration du modèle. La validation est donc une étape très importante, mais aussi très difficile. En effet, la confrontation des prédictions du modèle et des mesures est très délicate, puisque le modèle est par nature simplifié et les mesures par nature bruitées ! Les erreurs sont-elles dues à la simplicité du modèle ou au bruit des données ? Il n'est pas toujours simple de conclure. [10]

III-1- définition de prédiction

Prédiction est une opération mathématique où les valeurs futures d'un signal à temps discret sont estimées en fonction des échantillons précédents. Dans le traitement du signal numérique, la prédiction est souvent appelé codage

prédictif linéaire (LPC) et peut donc être considéré comme un sous-ensemble de la théorie du filtre. Dans l'analyse de système (un sous-champ des mathématiques), prédiction linéaire peut être considéré comme une partie de la modélisation mathématique ou d'optimisation. [11]

III-2- Principe de prédiction par apprentissage

On considère :

– Un signal y(t) ∈ R,

– A le point de coordonnées (t, y(t)),

–C le cercle osculateur à la courbe y(t) en A, de centre O = (tO, yO) et de rayon R(t).

On désigne alors par ˆy(t + h) la prédiction de y(t), faite à l'instant t, h représentant l'horizon de prédiction.

Dans de telles conditions, le problème de prédiction, qui a pour objectif de trouver le point P de coordonnées (t + h, ˆy(t + h)), peut se ramener, à l'instant t, à la recherche d'un coefficient λ(t) ∈ R tel que le point P puisse s'écrire P = λ(t)M +(1-λ(t))N , où le point N appartient à la tangente à la courbe en A, et le point M est l'intersection de C et de ON.

D'après cette définition, le coefficient λ(t) peut être interprété comme étant la pondération optimale entre une prédiction sur le cercle osculateur (point M) et une prédiction le long de la tangente en A (point).[8]

III-3-Le modèle de prédiction

Les déférentes méthodes de prédiction parent de l'hypothèse qu'un échantillon X(n)=X(Te), Te étant la période d'échantillonnage, d'un signal est prédictible à partir des P échantillons précédents sous la forme :

$$\hat{x}(n) = \sum_{i=1}^{P} a_i x(n-i) \qquad \text{.....1}$$

Où$\hat{x}(n)$ est la valeur du signal prédit, $x(n-i)$ les valeurs observées des précédents, et a_i des coefficients de prédicteur. L'erreur générée par cette estimation est :

$$e(n) = x(n) - \hat{x}(n) \quad \text{.....2}$$

Où x(n) est la valeur de signal originale. Ces équations sont valables pour tous les types de prédiction (unidimensionnelle) linéaire. Les différences se trouvent dans la façon dont les paramètres a_i sont choisis.

Pour les signaux multidimensionnels la métrique d'erreur est souvent définie comme :

$$e(n) = \|x(n) - \hat{x}(n)\| \quad \text{.....3}$$

Où $\|\cdot\|$ est une norme du vecteur choisi approprié.

III-4-L'estimation des paramètres

Le choix le plus courant dans l'optimisation des paramètres a_i est la moyenne de la racine carrée critère qui est aussi appelé le critère d'auto-corrélation. Dans ce procédé, on minimise la valeur attendue de l'erreur quadratique E [e2 (n)], ce qui donne l'équation.

$$\sum_{i=1}^{p} a_i R(j-i) = -R(j), \quad \text{.....4}$$

Pour $1 \leq j \leq p$, où R est l'auto-corrélation des signaux X_n, définie comme :

$$R(i) = E\{x(n)x(n-i)\} \quad \text{,.....5}$$

Et E est la valeur attendue. Dans le cas multidimensionnel ce qui correspond à minimiser la norme.

Les équations ci-dessus sont appelés les équations normales ou des équations de Yule-Walker. Sous forme matricielle les équations peuvent être écrites comme équivalente.

$$Ra = -r, \quad \text{.....6}$$

CONCLUSION

Dans ce chapitre, nous avons présenté l'opération de prédiction du signal, essentiellement le signal de parole, qui est notre objectif dans notre travail.

Chapitre 3 :

Conception & Implémentations du réseau de neurones

INTRODUCTION

Les réseaux de neurones sont déjà largement exploités dans les domaines de la classification et reconnaissance de forme ou de la parole. Nous l'appliquons également ici au signal de parole mais dans un but différent. Par un nombre d'entrées suffisantes du réseau de neurones, nous tenterons de réaliser une prédiction de l'échantillon suivant immédiatement cet ensemble. Nous comparerons les écarts type de l'erreur en vue de juger la meilleure configuration du réseau. Ensuite nous essayons d'utiliser un codage pour simuler la transmission du signal à travers un canal de communication.

I. PRESENTATION DE MATLAB ET VHDL :

Dans ce paragraphe, nous présentons les outils de programmation utilisés, afin de simuler notre conception.

I.1. Matlab :

MATLAB est un environnement puissant, complet et facile à utiliser destiné au calcul scientifique. Il apporte aux ingénieurs, chercheurs et à tout scientifique un système interactif intégrant calcul numérique et visualisation. C'est un environnement performant, ouvert et programmable qui permet de remarquables gains de productivité et de créativité.

Comme il est ouvert et extensible pour le calcul, Dispose de plusieurs centaines (voire milliers, selon les versions et les modules optionnels autour du noyau Matlab) de fonctions mathématiques, scientifiques et techniques. L'approche matricielle de MATLAB permet de traiter les données sans aucune limitation de taille et de réaliser des calculs numériques et symboliques de façon fiable et rapide. Grâce aux fonctions graphiques de MATLAB, il devient très facile de modifier interactivement les différents paramètres des graphiques pour les adapter selon nos souhaits.

MATLAB possède son propre langage, intuitif et naturel qui permet des gains de temps de CPU spectaculaires par rapport à des langages comme le C, le Turbo Pascal et le Fortran. Avec MATLAB, on peut faire des liaisons de façon dynamique, à

des programmes C ou Fortran, échanger des données avec d'autres applications (via la DDE : MATLAB serveur ou client) ou utiliser MATLAB comme moteur d'analyse.

MATLAB comprend aussi un ensemble d'outils spécifiques à des domaines, appelés Toolboxes (ou Boîtes à Outils). Indispensables à la plupart des utilisateurs, les Boîtes à Outils sont des collections de fonctions qui étendent l'environnement MATLAB pour résoudre des catégories spécifiques de problèmes. Les domaines couverts sont très variés et comprennent notamment le traitement du signal, l'automatique, l'identification de systèmes, les réseaux de neurones, la logique floue, le calcul de structure, les statistiques, … etc.

MATLAB permet le travail interactif soit en mode commande, soit en mode programmation ; tout en ayant toujours la possibilité de faire des visualisations graphiques. Considéré comme un des meilleurs langages de programmations (C ou Fortran), MATLAB possède les particularités suivantes par rapport à ces langages :

- la programmation facile.
- la continuité parmi les valeurs entières, réelles et complexes.
- la gamme étendue des nombres et leurs précisions.
- la bibliothèque mathématique très compréhensive.
- l'outil graphique qui inclut les fonctions d'interface graphique et les utilitaires.
- la possibilité de liaison avec les autres langages classiques de programmations (C ou Fortran). [13]

I.2. VHDL :

VHDL est un langage de description matériel destiné à représenter le comportement ainsi que l'architecture d'un système électronique numérique. VHDL est une abréviation de: « VHSIC (Very High Speed Integrated Circuit) Hardware Description Language ».

- *Language de programmation :* au même titre que le C, Java, PHP, etc...
- *Description matérielle :* description physique.
- *Descritpion du comportement et architecture :* description du fonctionnement du matériel.

De plus, le VHDL permet de générer du code exécutable :

- Il peut être simulé,
- Il peut être traduit en schéma de portes logiques (implémentation)

A ce titre, certaines instructions sont réservées à la simulation. De plus, la notion fondamentale de *séquentiel* peut trouver deux significations suivant le contexte :

- le code simulé va être exécuté, instructions par instructions, les unes après les autres,
- le code implémenté va être conçus grâce à de la logique combinatoire, et/ou, de la logique séquentielle. [14]

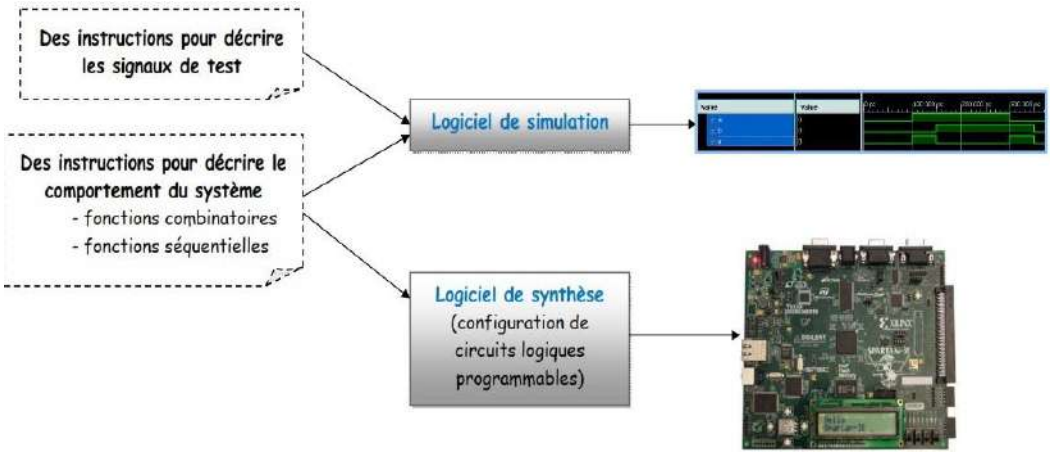


Figure III.1 : Processus de simulation et d'implémentation d'un code VHDL

II. CONCEPTION DU RESEAU DE NEURONES :

Dans le cadre de ce projet nous avons choisi comme application l'utilisation des réseaux de neurones artificiels dans la prédiction du signal de parole, leurs mises en œuvre nécessitent :

- La détermination des entrées et des sorties pertinentes, c'est à dire des grandeurs qui ont une influence significative sur le phénomène que l'on cherche à modéliser.
- La détermination du nombre de neurones cachés nécessaires pour obtenir une approximation satisfaisante.
- La réalisation de l'apprentissage.

- L'évaluation des performances du réseau de neurones à l'issue de l'apprentissage.

La structure du réseau dépend étroitement du type des échantillons. Il faut d'abord choisir le type de réseau : un perceptron standard, un réseau de Hopfield, un réseau à décalage temporel (TDNN), un réseau de Kohonen, etc...

Dans notre projet nous avons choisi d'utiliser le perceptron multicouche, mis à part les couches d'entrée et de sortie, il faut décider du nombre de couches cachées. Sans couche cachée, le réseau n'offre que de faibles possibilités d'adaptation. Il a été démontré qu'un Perceptron Multicouches avec une seule couche cachée pourvue d'un nombre suffisant de neurones, peut approximer n'importe quelle fonction avec la précision souhaitée.

Chaque neurone peut prendre en compte des profils spécifiques de neurones d'entrée. Un nombre plus important permet donc de mieux ″coller″ aux données présentées mais diminue la capacité de généralisation du réseau. Il faut alors trouver le nombre adéquat de neurones cachés nécessaire pour obtenir une approximation satisfaisante.

Il n'existe pas, à ce jour, de résultat théorique permettant de prévoir le nombre de neurones cachés nécessaires pour obtenir une performance spécifique du modèle, compte tenu des modèles disponibles. Il faut donc nécessairement mettre en œuvre une procédure numérique de conception de modèle.

II.1. Configuration du réseau de neurones :

Nous avons utilisé dans notre projet un signal de parole extrait de la base de données « TIMIT », enregistré à 8000Hz sur 16bits, dont la représentation temporelle est donnée par la figure III.2.

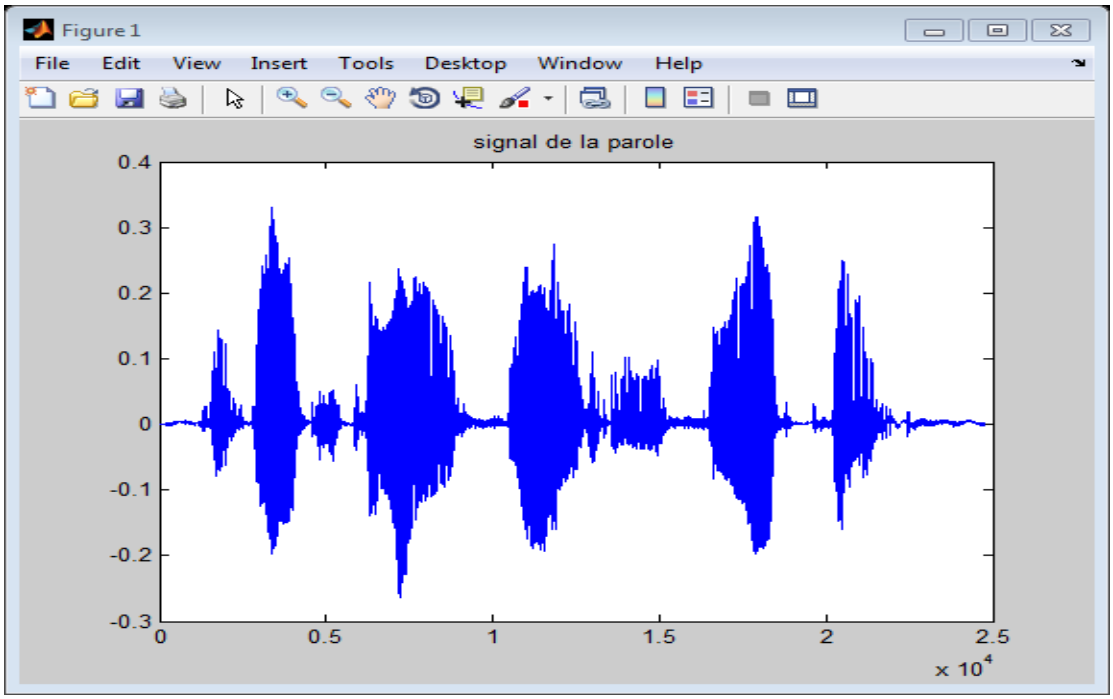


Figure III.2 : Représentation du signal de la parole utilisé

Après plusieurs essais, nous choisissons d'utiliser 25 échantillons en entrée du réseau, en d'autres termes, cela revient à estimer un échantillon sur la base des 25 précédents. De manière à ne pas augmenter de façon inutile la complexité du réseau, nous choisissons une seul couche cachée et un neurone à la sortie puisqu'il n'y a qu'un échantillon à prédire, Nous testerons nos réseaux sur (4, 8,12) neurones en couche cachée.

Trois fonctions de transfert seront envisagées : une fonction linéaire ('purelin') et deux fonctions sigmoïdes, ('tansig'), ('logsig').

A partir de ses fonction nous souhaitons une sortie signée, nous excluons la sortie en 'logsig', en fin on a 6 possibilité de configuration pour 2 couches de neurones et les 3 fonctions adoptées. Nous simplifions donc en ne considérant qu'un seul type de fonction par couche .les 6 couples sont donc les suivants :

- 'tansig'-'tansig'.
- 'logsig'-'tansig'.
- 'purelin'-'tansig'.
- 'purelin'-'purelin'.
- 'logsig'-'purelin'.
- 'tansig'-purelin'.

Nous annonçons tout de suite les résultats de la prédiction, nous augmentons le nombre d'essais pour tirer le maximum de conclusions.

Dans un premier temps, nous choisissons des vecteurs d'entrainement (d'entrées), de validation et de test.

Nous commençons les tests avec (4, 8, 12) neurones en couche cachée, en changeant le nombre d'itérations. La 1ere valeur indiquée est l'ecart-type (STD) de l'erreur de prédiction, la 2ème est le temps d'entrainement « apprentissage » (TIME).

Les tableaux suivants résument quelques résultats obtenus après une séquence des tests.

II.2. Test sans validation croisée :

Dans les paragraphes suivants, nous présentons les valeurs de l'écart type et du temps d'apprentissage, pour les différentes configurations.

II.2.1. 4 neurones en couche cachée :

STD TIME	1	2	3	4	5	6
200	0.0060 9.9944	0.0063 8.0117	0.0099 0.5029	0.0099 0.5747	0.0063 7.0222	0.0063 7.2986
400	0.0062 14.4895	0.0060 15.1825	0.0099 0.5420	0.0099 0.4821	0.0059 15.3370	0.0059 16.6763
600	0.0059 22.9816	0.0060 21.5719	0.0099 0.5555	0.0099 0.4485	0.0058 20.8862	0.0060 21.8823
800	0.0058 32.0828	0.0059 28.1519	0.0099 0.5496	0.0099 0.4989	0.0059 28.8638	0.0058 30.0284
1000	0.0060 45.3256	0.0059 41.6270	0.0099 0.6028	0.0099 0.5099	0.0060 34.8920	0.0057 36.1047

II.2.2. 8 neurones en couche cachée :

STD TIME	1	2	3	4	5	6
200	0.0046 14.5241	0.0043 14.4542	0.0099 0.6486	0.0099 0.6627	0.0051 14.0998	0.0046 13.8265
400	0.0046 41.0980	0.0045 29.2315	0.0099 0.7306	0.0099 0.6730	0.0045 28.4430	0.0041 27.1923
600	0.0044 45.9334	0.0037 48.7104	0.0099 0.7621	0.0099 1.0187	0.0046 45.7248	0.0044 45.2365
800	0.0043 59.0104	0.0041 60.6125	0.0099 0.7041	0.0099 0.5681	0.0040 57.1559	0.0041 55.4282
1000	0.0040 94.3336	0.0039 94.2428	0.0099 0.7157	0.0099 0.7393	0.0040 89.9041	0.0036 83.3559

II.2.3. 12 neurones en couche cachée :

STD TIME	1	2	3	4	5	6
200	0.0050 32.6072	0.0045 37.7761	0.0099 3.4194	0.0099 0.9372	0.0039 34.9794	0.0038 34.8289
400	0.0036 61.4414	0.0039 57.3076	0.0099 2.3982	0.0099 0.9811	0.0036 55.5184	0.0036 55.2153
600	0.0034 128.7196	0.0032 73.3818	0.0099 1.5625	0.0099 1.0085	0.0032 77.9756	0.0035 79.6028
800	0.0033 135.8585	0.0034 142.0616	0.0099 1.0527	0.0099 0.9393	0.0032 69.1436	0.0032 98.5842
1000	0.0033 135.8585	0.0032 155.1296	0.0099 1.1085	0.0099 0.8583	0.0032 122.7673	0.0032 82.7914

II.3. Test avec validation croisée :

On trouve dans la littérature deux principales techniques pour limiter le sur-apprentissage : la régularisation bayésienne et la validation croisée. Nous avons choisi d'utiliser dans notre projet la technique de validation croisée.

Cette technique consiste à calculer l'erreur d'une trame supplémentaire appelée trame de validation pendant l'apprentissage. Cette erreur devrait diminuer avec les époques. Mais dès qu'elle commence à augmenter, on arrête l'apprentissage, car cela veut dire qu'il y a sur-apprentissage.

Dans les paragraphes suivants, nous présentons les valeurs de l'écart type et du temps d'apprentissage, pour les différentes configurations, en utilisant la validation croisée.

II.3.1. 4 neurones en couche cachée :

STD TIME	1	2	3	4	5	6
200	0.0090 1.5655	0.0096 1.3771	0.0099 0.5786	0.0099 0.5329	0.0099 0.7028	0.0084 0.8669
400	0.0088 1.1071	0.0090 0.9176	0.0099 0.5160	0.0099 0.4933	0.0091 1.2893	0.0091 1.0118
600	0.0089 0.8652	0.0093 0.9222	0.0099 0.5659	0.0099 0.4927	0.0087 1.3331	0.0074 0.8052
800	0.0068 0.9814	0.0095 0.7738	0.0099 0.7996	0.0099 0.4899	0.0088 0.7957	0.0108 0.7937
1000	0.0093 1.0766	0.0093 1.7939	0.0099 0.6453	0.0099 0.4906	0.0091 1.2629	0.0079 1.0835

II.3.2. 8 neurones en couche cachée :

STD TIME	1	2	3	4	5	6
200	0.0089 2.4853	0.0087 1.1692	0.0099 0.8331	0.0099 0.6878	0.0090 1.0576	0.0084 1.0459
400	0.0093 1.6267	0.0089 1.3638	0.0099 0.7448	0.0099 0.6886	0.0063 1.2621	0.0078 1.1516
600	0.0083 1.3847	0.0089 2.2967	0.0099 1.0346	0.0099 0.7438	0.0091 1.3399	0.0084 1.2271
800	0.0086 1.4770	0.0091 2.9043	0.0099 0.8185	0.0099 0.7432	0.0091 1.4014	0.0087 1.3577
1000	0.0089 1.3372	0.0091 1.5237	0.0099 0.8911	0.0099 0.7755	0.0088 1.7878	0.0096 1.2438

II.3.3. 12 neurones en couche cachée :

STD TIME	1	2	3	4	5	6
200	0.0076 1.9659	0.0079 1.8188	0.0099 1.2849	0.0099 1.0897	0.0092 1.9759	0.0076 2.1749
400	0.0087 2.0153	0.0090 1.8348	0.0099 1.2679	0.0099 1.0441	0.0136 1.7793	0.0071 2.1432
600	0.0088 3.0627	0.0093 2.4225	0.0099 1.0963	0.0099 0.9544	0.0085 1.8701	0.0069 2.1519
800	0.0101 2.2690	0.0077 2.3833	0.0099 1.2411	0.0099 1.1124	0.0085 2.0105	0.0074 1.8129
1000	0.0094 1.5084	0.0096 2.5685	0.0099 1.0913	0.0100 0.9676	0.0093 1.6127	0.0067 1.9291

II.4. Discussion des Résultats :

Nous constatons d'après les différents tableaux présentés dans les paragraphes précédents, une légère amélioration de l'écart type et du temps d'entrainement si le nombre d'itération augmente.

En fin, à partir les tests que nous avons fait-on peut tirer les conclusions suivantes :

- Nous remarquons une diminution de l'écart type de l'erreur de prédiction, en augmentant le nombre de neurones en couche cachée (4, 8 puis 12).
- Les 2 réseaux linéaires (purelin/tansig et purelin-purelin) – on entend par réseau linéaire, un réseau dont la couche cachée est linéaire – paraissent converger plus rapidement.
- La convergence des réseaux non linéaires est plus longue ; on pourrait cependant arrêter l'algorithme beaucoup plus tôt tout en conservent Un écart-type acceptable. On ne pourrait cependant égaler les performances Temporelles des réseaux linéaires.
- Les réseaux non linéaires fournissent finalement la meilleure valeur de l'écart type mais aucun ne se distingue.
- Les 2 réseaux linéaires sont moins sensibles au nombre de vecteurs d'entraînement (et de validation) que les réseaux non linéaires.

On constate une petite divergence des échantillons prédits par rapport au signal original. Les Figures suivantes montrent cette petite divergence. La première

superpose les échantillons prédits à ceux attendus (signal original) et la deuxième donne le signal d'erreur de prédiction. Ces échantillons mal prédits ont en effet servi à la validation croisée. Pour rappel, un entraînement sans validation croisée est d'abord mis en œuvre pour éviter un arrêt trop rapide dû à un minimum local. Ensuite l'entraînement se poursuit, mais cette fois-ci combiné à une validation croisée. Celui-ci s'arrête assez rapidement de manière générale. Mais n'est cependant pas valable pour les 2 réseaux linéaires qui y sont moins sensibles. Ceux-ci s'adaptent également plus vite pour les mêmes raisons évoquées plus haut.

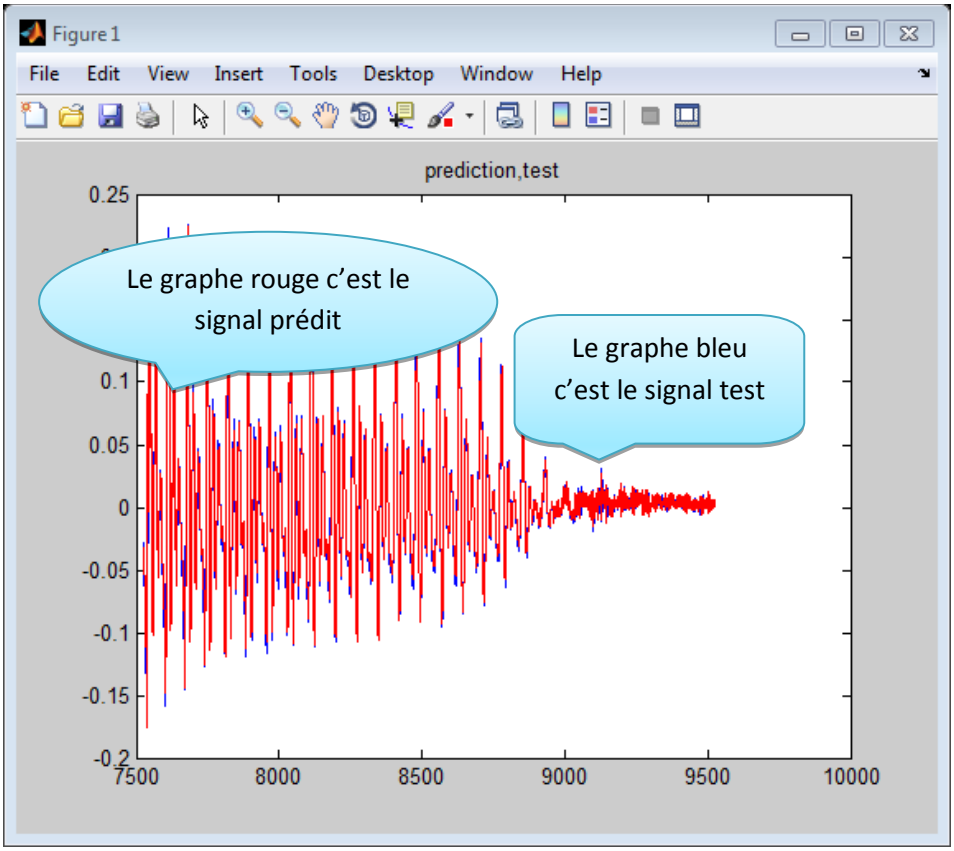


Figure III.3 : signal original et signal prédit

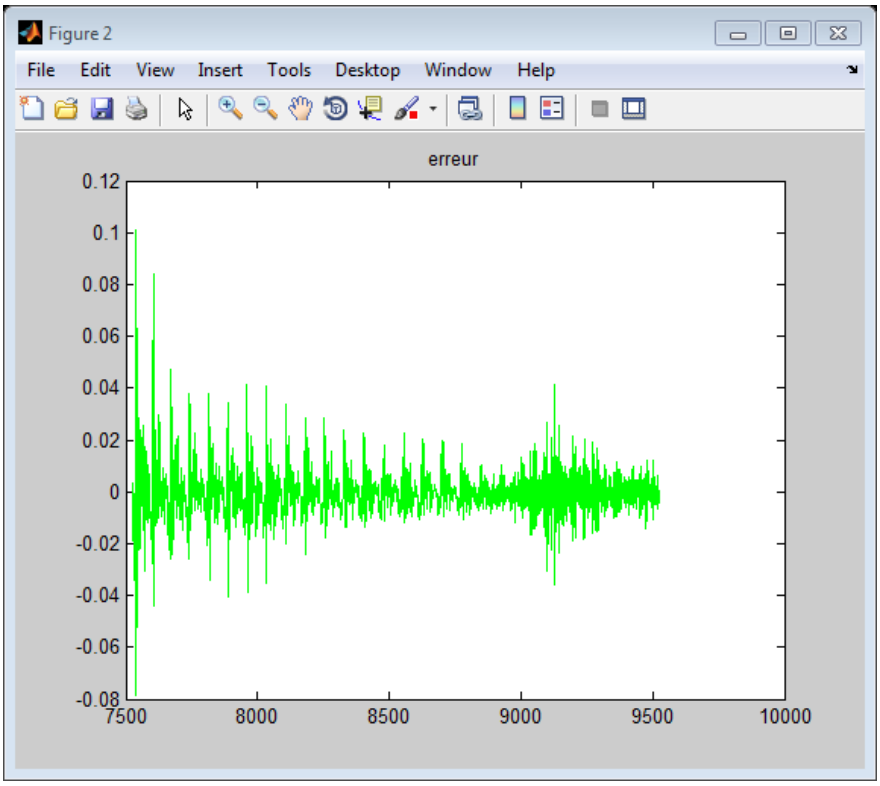


Figure III.4 : Signal d'erreur de prédiction

III. REALISATION D'UNE INTERFACE GRAPHIQUE :

Pour faciliter la séquence de tests, nous avons essayé de réaliser une interface graphique en Matlab, et qui est montrée dans la figure suivante :

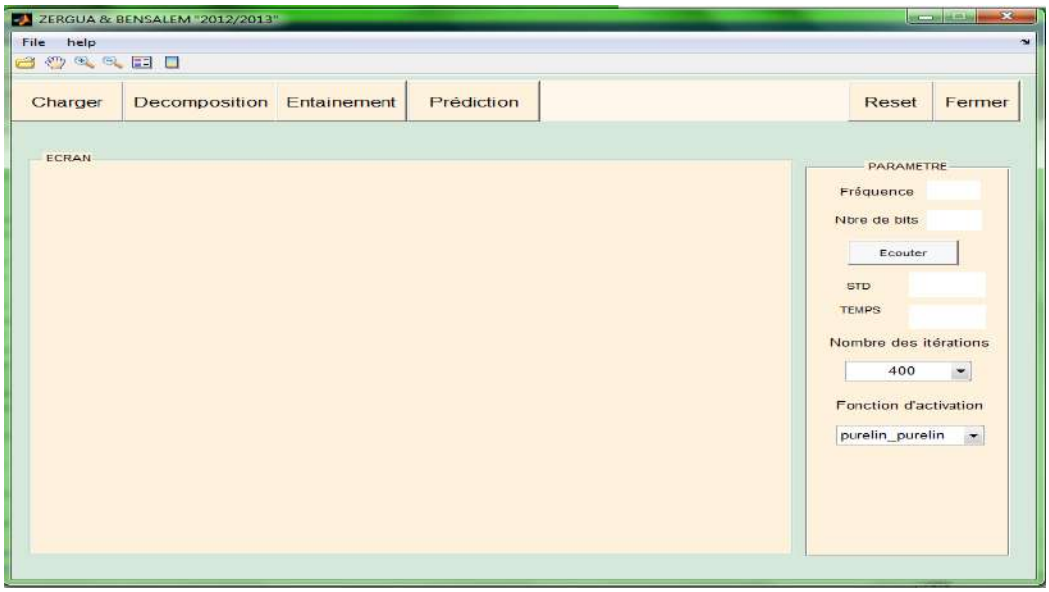


Figure III.5 : L'interface graphique de test du réseau de neurones

Les boutons en haut permettent de :

- Charger : import le signal de parole souhaité qui doit être d'extension « *.wav* ».
- Décomposition : permet de décomposer le signal audio à un signal d'entrée, signal cible, de validation et un signal test_cible.
- Entrainement : permet d'entrainer le réseau de neurones.
- Prédiction : permet d'estimer le signal de parole, en traçant le signal original, le signal prédit et l'erreur de prédiction.

Nous obtenons après chaque essai, les valeurs de l'écart type de l'erreur de prédiction, ainsi que le temps d'apprentissage.

L'interface graphique précédente, nous offre la possibilité de modifier l'architecture du réseau de neurones à chaque essai.

Nous présentons maintenant, la démarche d'utilisation de notre interface :

Nous commençons par charger le signal audio comme le montre la figure suivante :

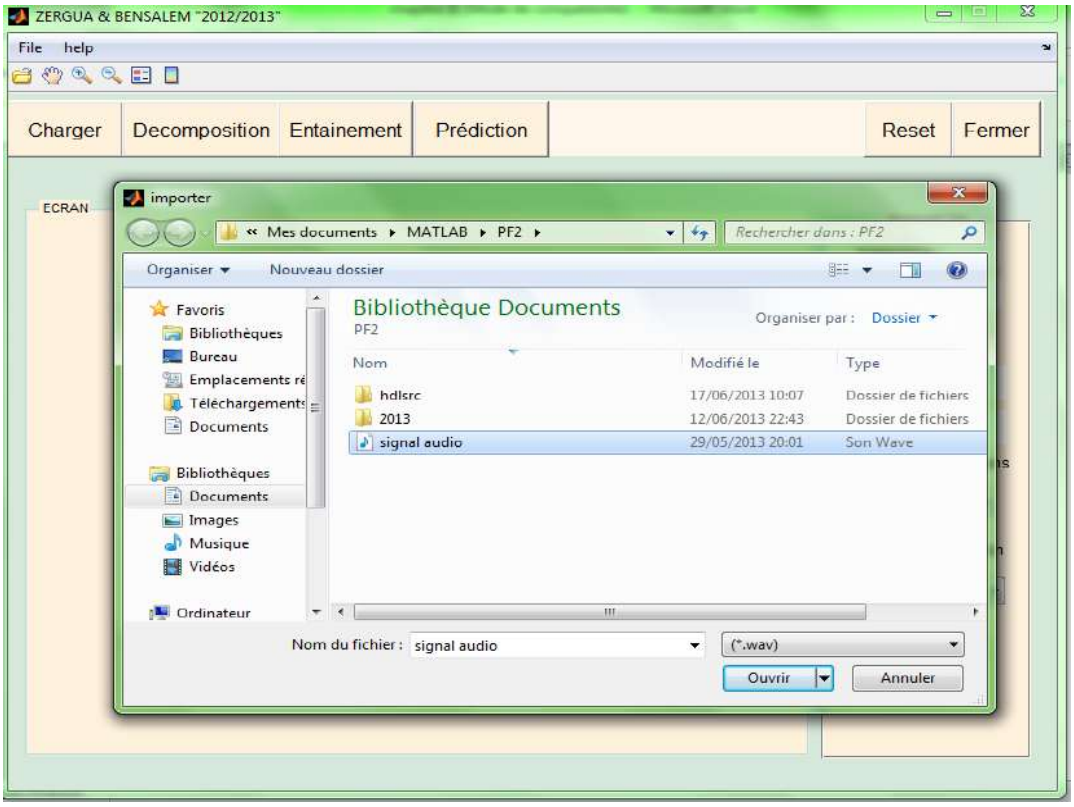


Figure III.6 : Chargement du fichier audio

Ensuite, nous devons décomposer notre signal, comme le montre la figure suivante :

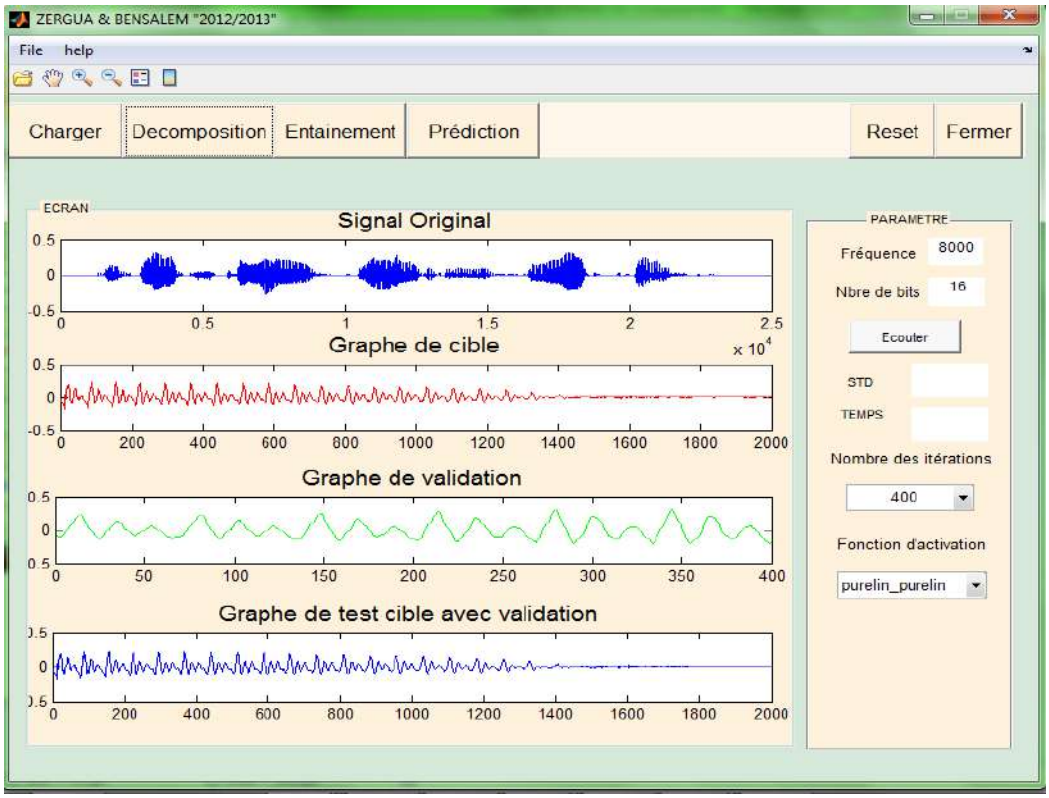


Figure III.7 : Décomposition du signal audio

En cliquant sur le bouton « Entrainement », l'interface de l'entraînement du Matlab apparaît :

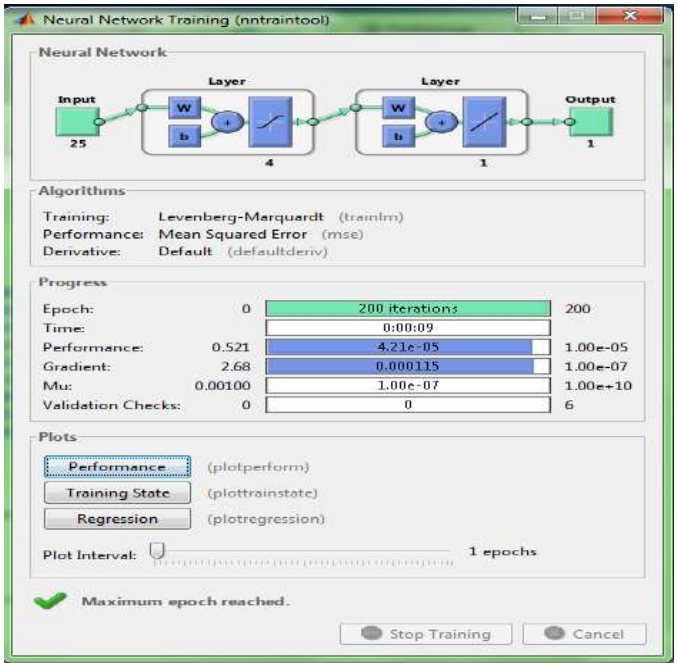


Figure III.8 : Déroulement du processus d'entraînement du réseau

Après nous pouvons visualiser les signaux originaux, prédit et l'erreur de prédiction en cliquant sur le bouton « Prédiction ».

IV.MODELISATION SIMULINK DE NOTRE RESEAU DE NEURONES :

D'après les résultats de test précédents, nous choisissons la meilleure configuration du réseau, afin de réaliser un modèle Simulink, qui va nous permet de simuler le réseau de neurones.

Nous avons choisi un réseau avec 12 neurones en couche caché, avec les fonctions d'activation tansig/purelin, et en déroulant l'apprentissage avec 200 itérations.

En descendant en hiérarchie dans le bloc du réseau de neurones nous pouvons voire la couche d'entrée, la couche cachée, et la couche de sortie.

Nous devons en premier temps réaliser le bloc Simulink de la configuration choisie. La figure III.9 présente le réseau de neurone de la configuration choisie.

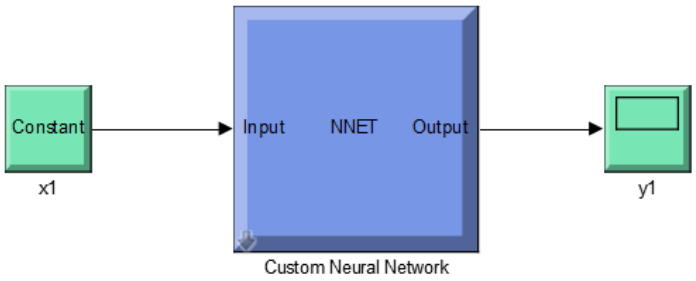


Figure III.9 : le bloc du réseau de neurones crée en Simulink

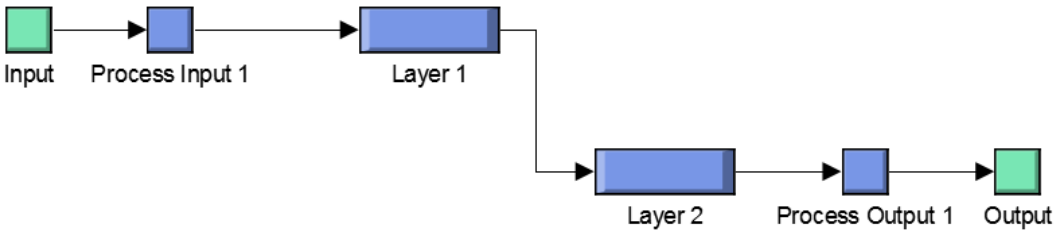


Figure III.10 : représentation du réseau de neurones en couches

C'est un réseau comportant trois couches, une d'entrée, une cachée et une de sortie. La figure suivante, montre le schéma bloc de la couche cachée :

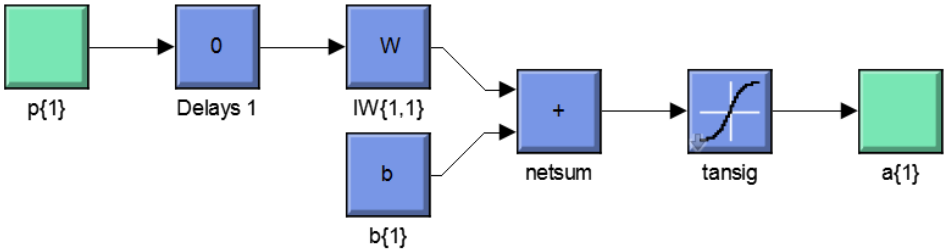


Figure III.11 : la couche cachée

En développant le bloc des poids de la couche cachée, nous obtiendrons la figure III.12 :

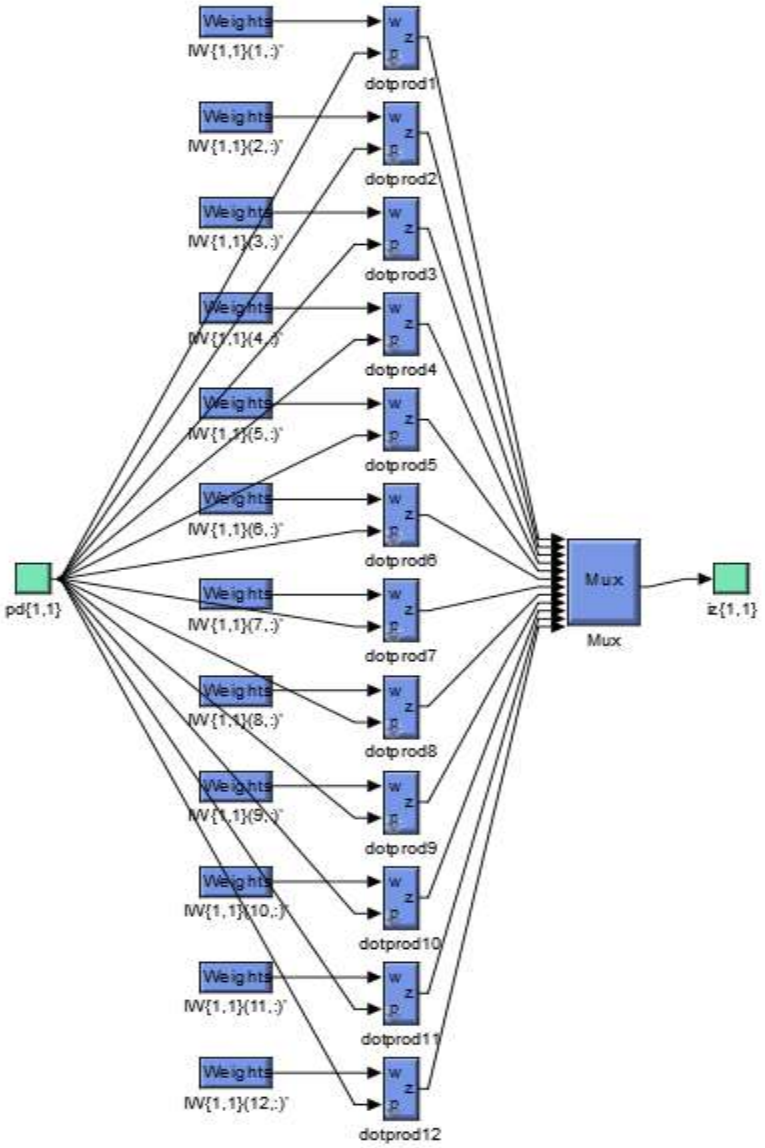


Figure III.12 : les 12 neurones en couche cachée

De la même manière, nous présentons la couche de sortie.

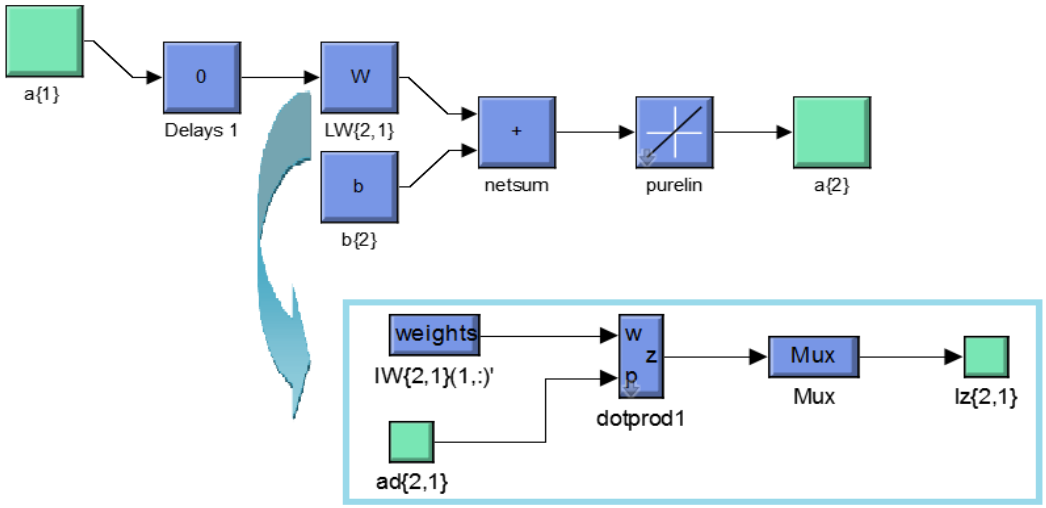


Figure III.13 : la couche de sortie

Afin de simuler notre réseau de neurones en utilisant un signal de parole, nous devons générer l'entrée du réseau qui doit comporter 25 échantillons.

Pour cela nous avons écrit un script Matlab, qui permet d'échantillonner le signal de parole en 25 échantillons par trame.

Nous avons aussi, placé un oscilloscope dans le modèle Simulink, pour visualiser les différents signaux.

Le modèle ainsi réalisé est le suivant :

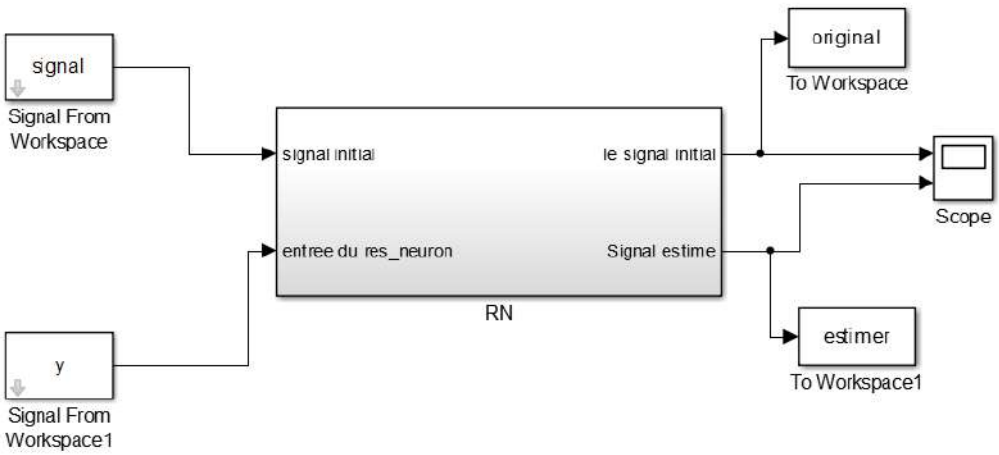


Figure III.14 : le modèle Simulink

Après le déroulement de la simulation, nous avons obtenu la figure suivante :

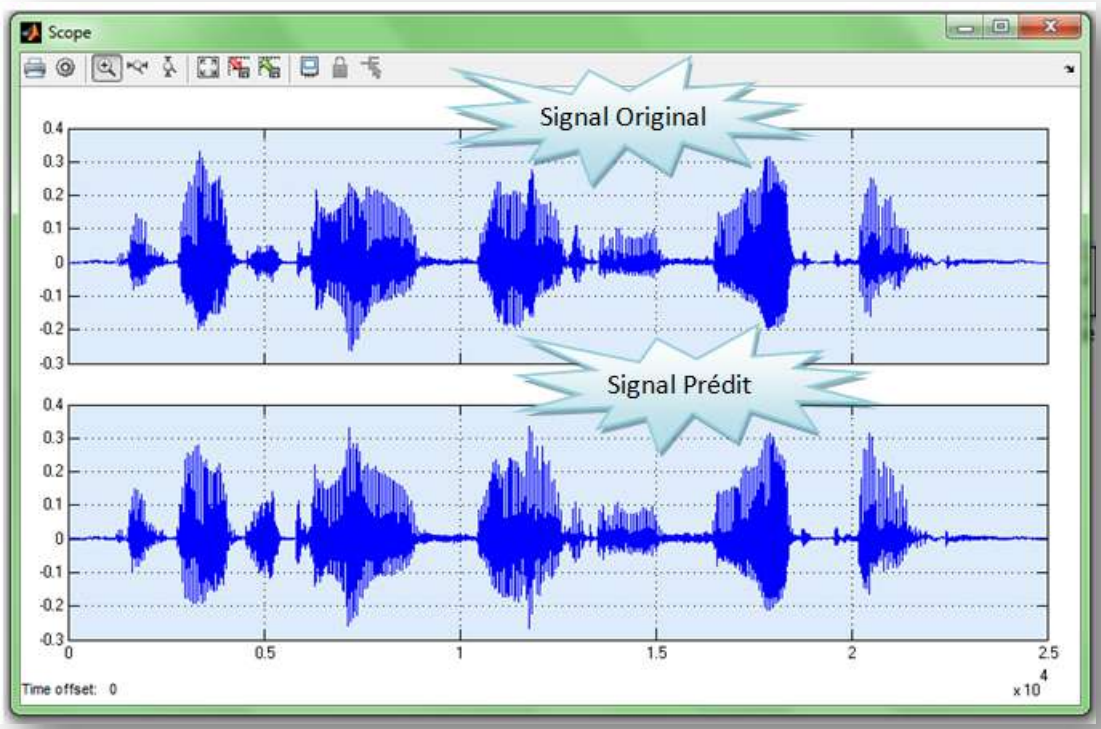


Figure III.15 : le signal original et le signal prédit

D'après cette dernière figure, nous pouvons dire que notre estimateur donne des résultats acceptables.

V. GENERATION DES CODES VHDL ET SIMULATION MODELSIM DU RESEAU DE NEURONES :

Dans ce paragraphe, nous présentons les résultats de simulation VHDL en utilisant ModelSim, dans le but de voire la possibilité d'implémentation matérielle des réseaux de neurones sur circuits logiques programmables.

Pour cela, nous avons utilisé le logiciel Matlab pour généré des codes VHDL correspondant au modèle Simulink réalisé dans la partie précédente.

Dans un premier temps, nous avons vérifié la compatibilité de notre modèle Simulink avec le langage VHDL grâce à la commande Matlab « ***checkhdl*** ».

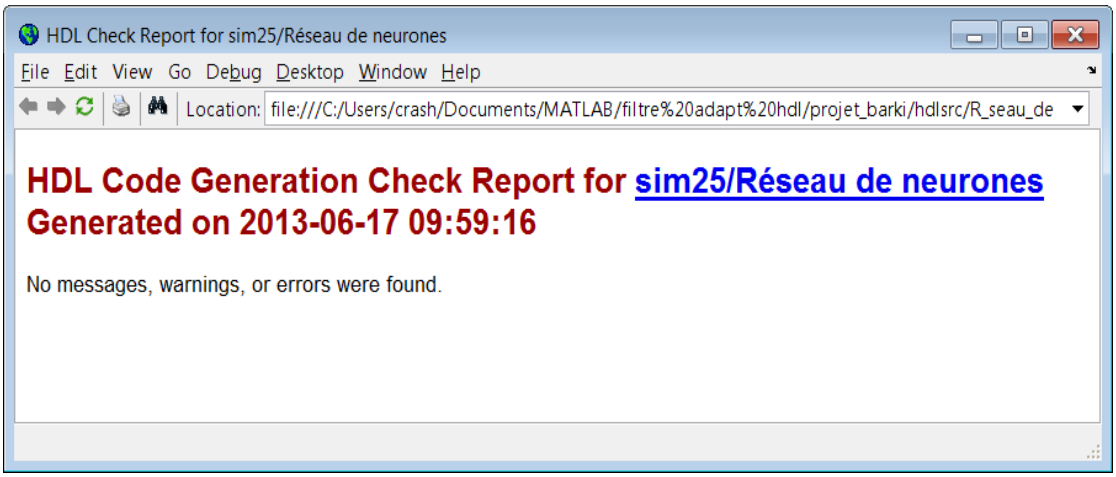


Figure III.16 : Vérification de compatibilité VHDL

Après vérification de compatibilité du modèle Simulink, nous avons exécuté la commande « ***makehdl*** », afin de générer les codes VHDL correspondantes.

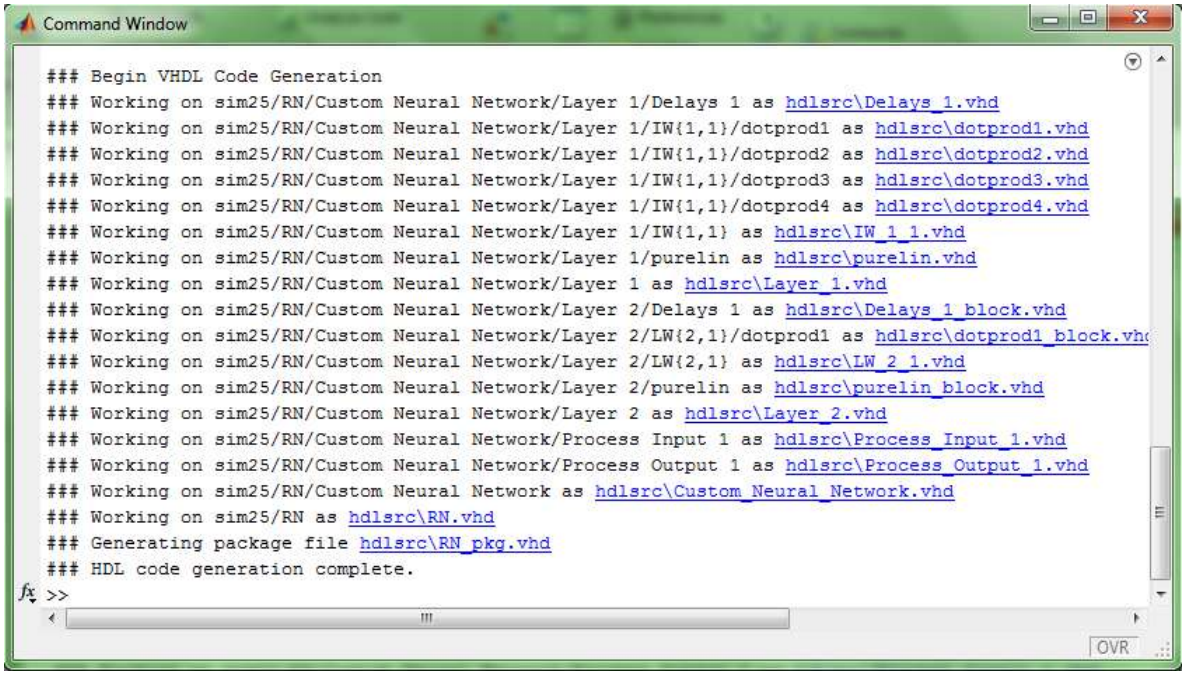


Figure III.17 : Générations des codes VHDL

Finalement, nous avons créé un ***testbech***, en exécutant la commande Matlab « ***makehdltb*** », afin de simuler les codes VHDL en ModelSim.

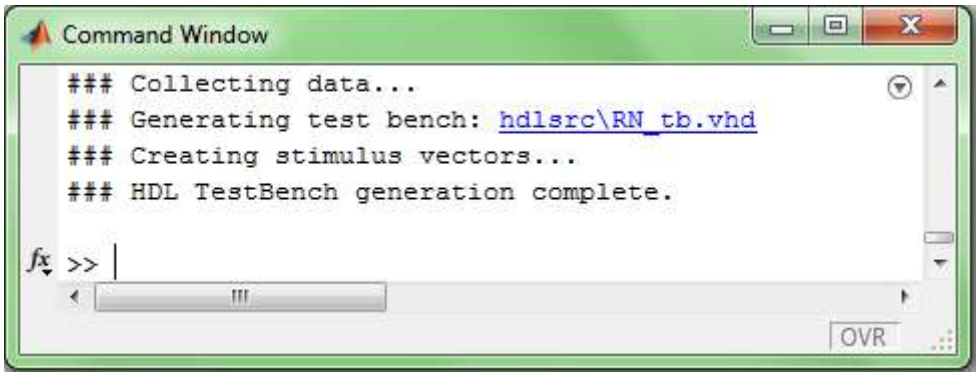


Figure III.18 : Génération du testbench

Après ces étapes, nous avons lancé le simulateur ModelSim à partir du Matlab, grâce à la commande « ***vsim*** », comme le montre la figure suivante :

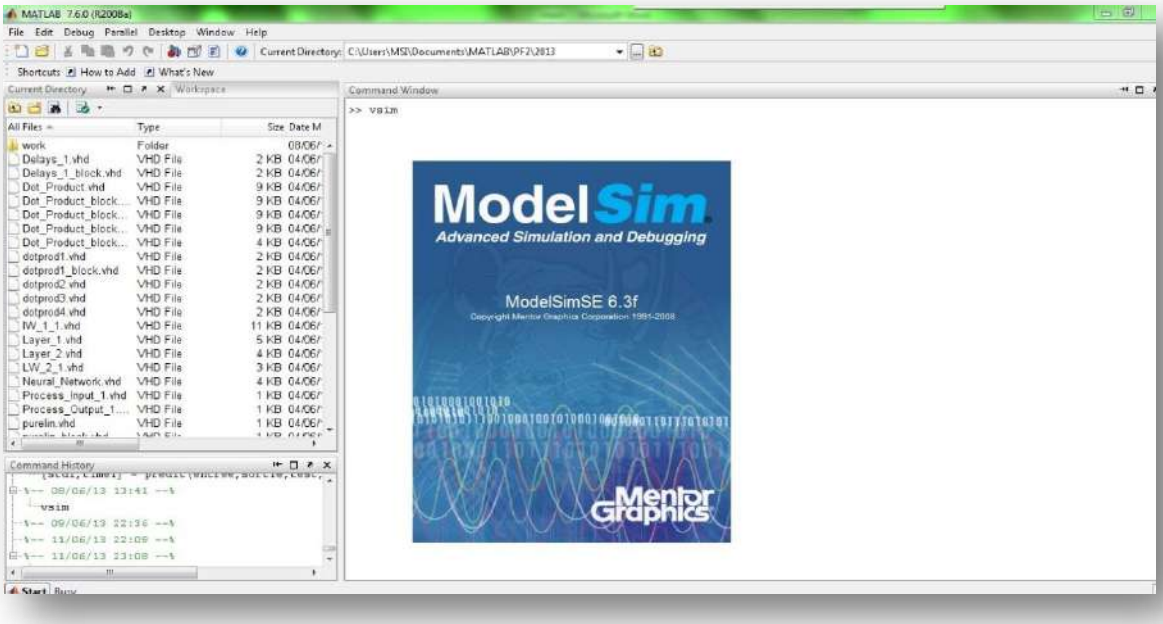


Figure III.19 : Lancement du simulateur ModelSim par Matlab

Enfin, nous avons obtenus les résultats de simulation suivants :

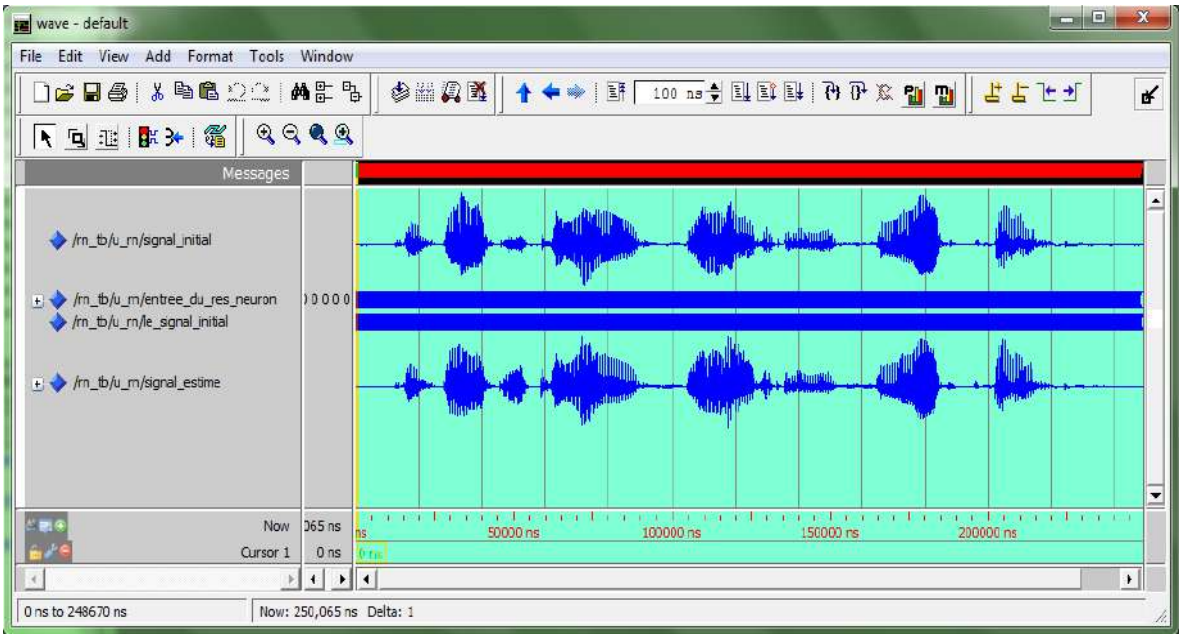


Figure III.20 : Les résultats de simulation ModelSim

Cette figure, nous montre le signal estimé qui est quasi identique au signal original.

VI. SIMULATION D'UNE CHAINE DE COMMUNICATION UTILISANT LAS RESEAUX DE NEURONES :

Nous avons ensuite, simulé notre réseau de neurone de prédiction en utilisant Simulink dans une chaine de communication, en utilisant un canal de communication de type AWGN (Additif White Gaussien Noise), et un codeur/décodeur uniforme.

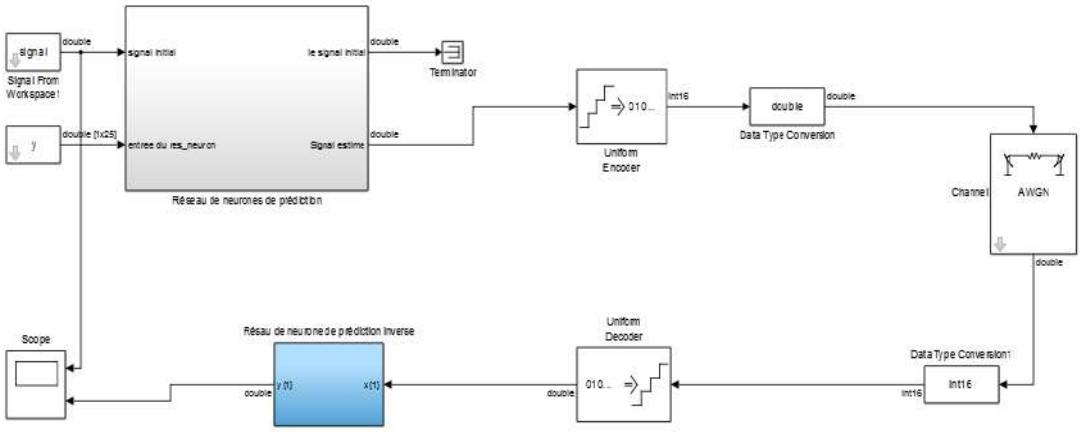


Figure III.21 : Le modèle Simulink de la chaine de communication

Dans notre modèle Simulink, nous avons codé le signal prédit par un codeur uniforme, puis nous l'avons transmis par un canal de communication de type AWGN, et en réception nous avons utilisé un décodeur uniforme, afin de décoder le signal transmis, où nous avons implémenté un autre réseau de neurones de prédiction inverse qui est utilisé pour la prédiction du signal original.

Dans le but de réaliser un réseau de neurones de prédiction inverse, nous avons écrit un script Matlab, qui nous a permis de simuler ce réseau tout seule.

Les instructions suivantes permettent de générer un signal audio, puis de le bruité par un bruit normal de variance 0.02 :

```
[signal,fe,nbits]=wavread('signal audio');
signal_bruit=signal+.02*randn(length(signal),1)';
```

La représentation temporelle de ces signaux est la suivante :

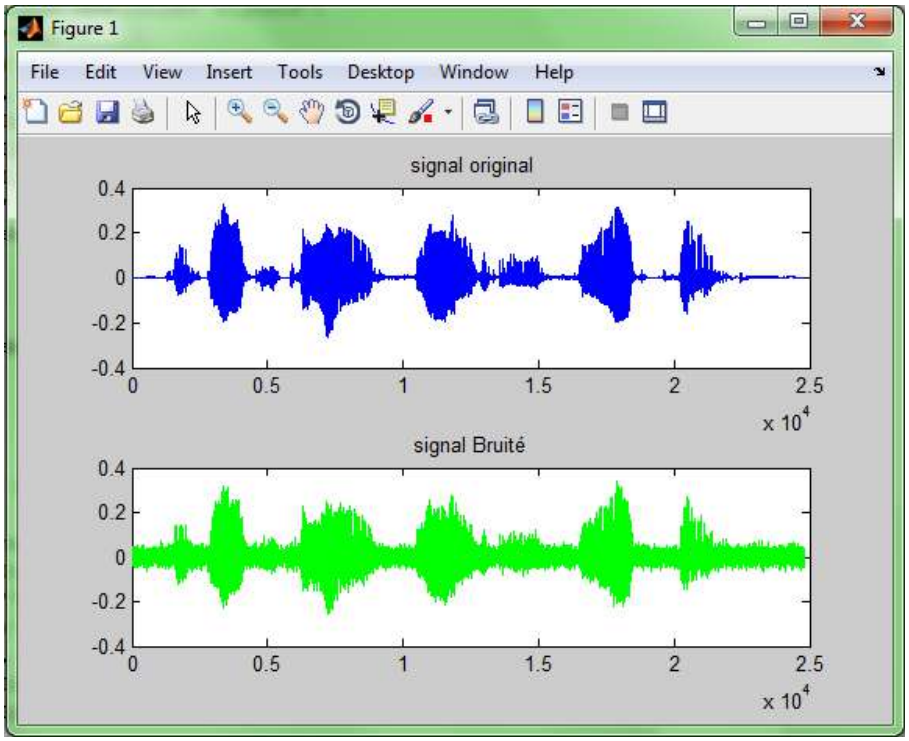


Figure III.22 : La représentation temporelle du signal original et du signal bruité

Après, nous avons créé un réseau linéaire, dont code est le suivant :

```
net = newlin([-3 3],1,[0 1],0.1);
```

Ensuite, nous avons adapté notre réseau par l'instruction suivante :

```
[net,Y,E,Pf]=adapt(net,signal,signal_bruit);
```

En traçant le signal de sortie du réseau Y, nous pouvons voire le signal débruité :

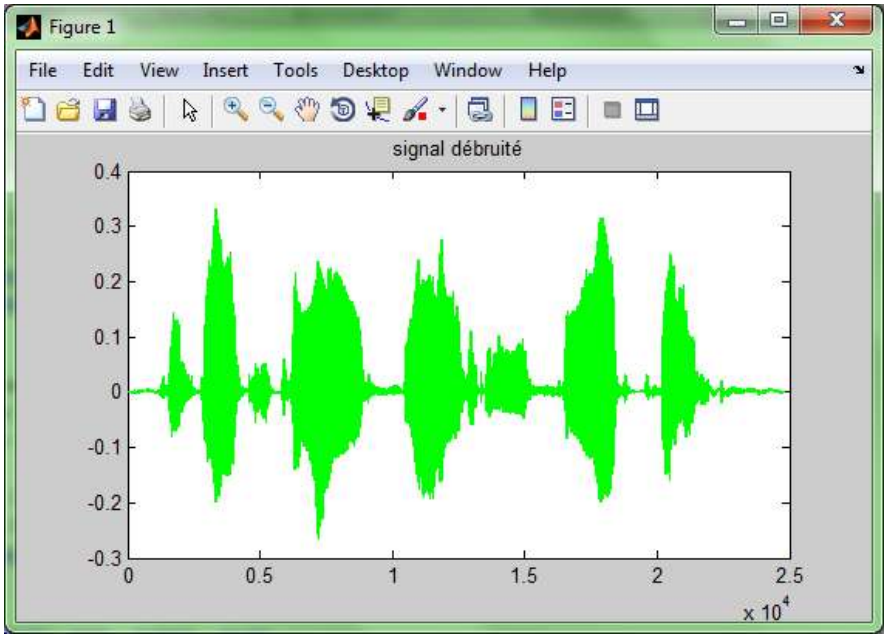


Figure III.23 : Le signal prédit

Après cette étape, nous avons réalisé le bloc Simulink correspondant à ce réseau, que l'on nomme réseau de prédiction inverse.

Après déroulement de simulation de notre modèle Simulink, nous avons obtenus les résultats suivants :

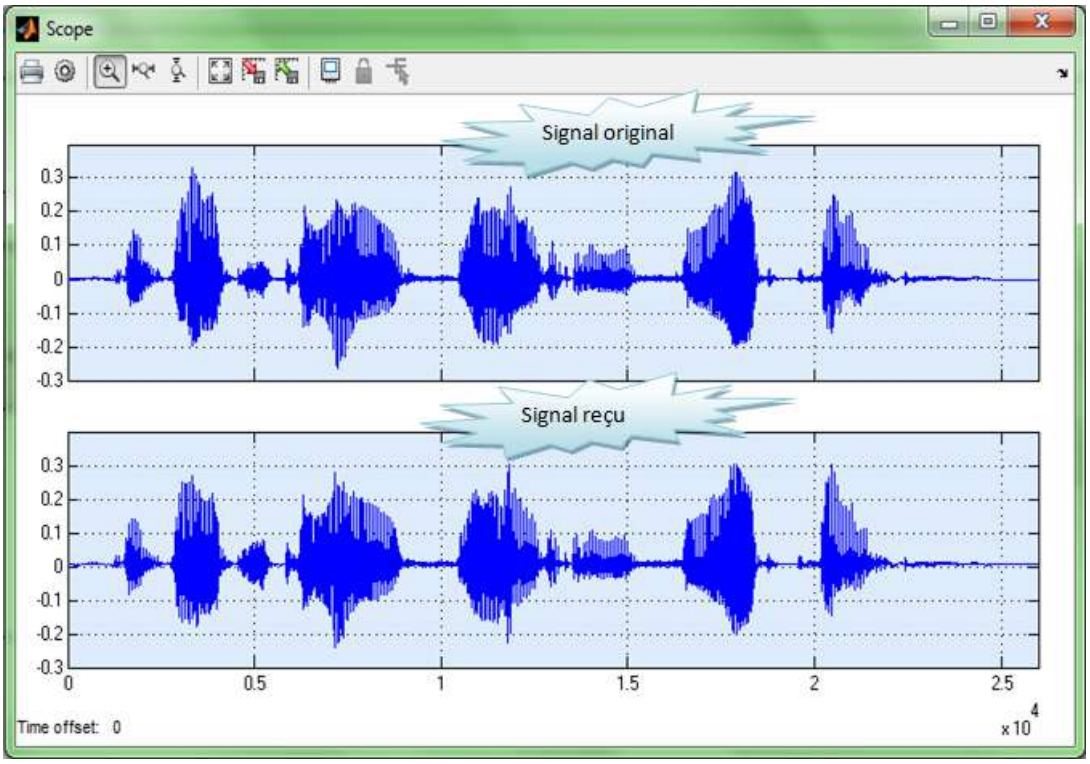


Figure III.24 : Le signal original et le signal reçu

Cette partie de simulation nous montre la possibilité d'utiliser les réseaux de neurones dans une chaine de communication.

CONCLUSION

Dans ce chapitre, nous avons présenté les différentes manipulations de réseau de neurones, afin de prédire un signal de parole. Nous avons aussi réalisé un réseau de neurones en Simulink, dans le but de simuler notre réseau en utilisant un canal de communication. Nous avons aussi simulé en ModelSim les codes VHDL de notre réseau de neurones pour les comparer avec les résultats de simulation Matlab, en vue d'une implémentation dans un circuit logique programmable. Les résultats obtenus dans ce chapitre sont satisfaisantes.

Conclusion Générale

Ce projet a été consacré à l'application des réseaux de neurones dans la prédiction du signal de parole.
Les réseaux de neurones ont d'énorme avantage de pouvoir être appliqués sur un large éventail de domaines de l'apprentissage.

Dans ce travail, nous avons donné des généralités sur les réseaux de neurones .En premier lieu, nous avons introduit un rappel sur l'apprentissage d'un réseaux de neurones . Puis nous avons fait un rappel sur les différents types et leurs domaines d'application. Dans le deuxième chapitre nous avons présenté l'opération de prédiction des signaux, essentiellement le signal de parole, où nous avons présenté l'utilisation des réseaux de neurones artificiels dans la prédiction.

Dans le chapitre conception et implémentation, nous avons fait plusieurs simulations sur les réseaux de neurones pour prédire un signal de parole, afin de tirer le maximum d'informations. Les résultats obtenus, sont efficaces, et satisfaisantes.

Enfin, les réseaux de neurones restent un champ très important dans la prédiction des signaux, où plusieurs configurations sont possibles.

Bibliographie

[1] Marc Parizeau, RESEAUX DE NEURONES GIF-21140 et GIF-64326.

[2] Yann Boniface, Nicolas P. Rougier, Apprentissage et Mémoires (Introduction aux réseaux de neurones artificiels), Janvier 2012.

[3] M.T. Hagan, H.B. Demuth, M. Beale, «Neural Network Design», PWS Publishing Compagny, 1995.

[4] Claude TOUZET,LES RESEAUX DE NEURONES ARTIFICIELS, Juillet 1992.

[5] Simon Haykin, «Neural Networks : A Comprehensive Foundation», IEEE Press, 1994.

[6] technique neurales adaptées aux données spation-temporelles.

[7] Introduction aux réseaux de neurones 'CONSIDÉRATIONS ACTUARIELLES'.

[8] Prédiction en ligne par apprentissage et exploitation des caractéristiques géométriques des signaux.

[9] introduction au traitement automatique de la parole

[10] Christian Jutten (Enseignant-chercheur), 23/09/2011.

[11] Emmanuel Ravelli, PRÉDICTION NON-LINÉAIRE POUR LE CODAGE AUDIO SANS PERTES

[12] Présentation de Matlab.

[13] Remacle Matthieu, Electronique numérique 'Le language VHDL', 28 février 2013.