الجمهورية الجزائرية الديمقراطية الشعبية
LA REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Saad Dahlab Blida 1

Institut d'Aéronautique et des Études Spatiales
Département Navigation Aérienne

Mémoire de fin d'études
En vue de l'obtention du diplôme de Master en Aéronautique

Option : **CNS/ATM**

THEME

Reconnaissance de la modulation impulsionnelle du signal radar Basé sur les Réseaux de Neurones Convolutionnels

Proposé et dirigé par :

Dr. AZMEDROUB Boussad

Réalisé par :

M. SAADI Ala
M. DERHAMOUNE Mohammed El Amin

Promotion: 2019 / 2020

Remerciement :

Tout d'abord, nous remercions Dieu, le Créateur de l'univers qui nous a donné la force et nous a maintenus en santé pour accomplir et réaliser ce modeste travail.

J'offre premièrement de sincères et chaleureux remerciements à notre promoteur ***M AZMEDROUB Boussad*** *pour nous avoir encadrés durant cette période critique d'épidémie, pour son aide , son attention exceptionnels, ainsi que le soutien moral qu'il nous a accordés.*

Nous remercions les membres de jury qui nous ont fait l'honneur de participer au jugement de ce travail.

Finalement, ce travail n'aurait pas vu le jour sans la présence, l'écoute, la confiance de nombreuses personnes

Nos sincères remerciement et nos profondes gratitudes sont destinés a

Nos chers parents

Nos frères et sœurs

Nos amis et nos familles

Ainsi que tous ceux dont nous avons oublié de mentionner le nom.

Résumé :

Dans ce mémoire une approche de reconnaissance de modulation intra-impulsion pour le signal radar a été proposé dans le but d'augmenter le taux de reconnaissance par rapport à un projet de travail déjà existé , le travail se base sur des techniques d'analyse temps-fréquence, de traitement d'image et de réseau de neurone convolutionnel .

Les signaux reçus transformèrent à des représentations temps-fréquence grâce à la distribution de Wigner-Ville ou de la transformée de Fourier à court terme, ces représentations tournèrent a des images de même taille grâce à des techniques de traitement d'image, Nous concevons un classificateur a réseau de neurone convolutionnel pour identifier les images Temps-Fréquence traitées.

l'approche proposée peut identifier 4 types de modulation, y compris la modulation de fréquence et la modulation de phase pour différent rapport signal sur bruit.

Abstract :

In this thesis an intra-pulse modulation recognition approach for the radar signal was proposed in order to increase the recognition rate compared to other already existing approaches. The approach is based on time- frequency analysis techniques, image processing and convolutional neuron network

The signals received transformed to time-frequency representations thanks to the Wigner-Ville distributions or the short-term Fourier transform, these representations turned to images of the same size thanks to image processing techniques, We design a Convolutional neural network classifier to identify processed Time-Frequency Images.

The proposed approach can identify 4 types of modulation, including frequency modulation and phase modulation for different signal-to-noise ratio

ملخـــــص :

في هذه الأطروحة تم اقتراح نهج التعرف على التضمين داخل النبضة لإشارة الرادار من أجل زيادة معدل التعرف مقارنة بالنُهج الأخرى الموجودة بالفعل ، ويستند هذا النهج إلى تقنيات تحليل **وقت - التردد** ومعالجة الصور وشبكة الخلايا العصبية الالتفافية.

تحولت الإشارات المستقبلة إلى تمثيلات تردد زمنية بفضل توزيعات Wigner-Ville أو تحويل فوريه قصير المدى ، وتحولت هذه التمثيلات إلى صور من نفس الحجم بفضل تقنيات معالجة الصور ، نحن نصمم مصنف الشبكة العصبية الالتفافي لتحديد صور **وقت - تردد** التي تمت معالجتها.

يمكن للنهج المقترح تحديد 4 أنواع من التضمين ، بما في ذلك تضمين التردد وتضمين الطور لنسبة إشارة على ضوضاء مختلفة

Table des matières

ABREVIATIONS :

HF : HAUTE FREQUENCE

BF : BASSE FREQUENCE

CW : ONDE CONTINUE (CONTINUOUS WAVE)

EW : GUERRE ELECTRONIQUE (ELECTRONIC WAR)

FMCW : ONDE CONTINUE MODULEE EN FREQUENCE (FREQUENCY MODULATION CONTINUOUS WAVE)

MTI : VISUALISATION DES CIBLES MOBILES (MOVING TARGET INDICATION)

ELINT :INTELLIGENCE ELECTRONIQUE (ELECTRONIC INTELLIGENCE)

SNR : RAPPORT SIGNAL SUR BRUIT (SIGNAL-NOISE RATIO)

BPSK : MODULATION PAR CHANGEMENT DE PHASE (BINARY PHASE SHIFT KEY)

LFM : MODULATION EN FREQUENCE LINEAIRE (LINEAR FREQUENCY MODULATION)

NLFM : MODULATION EN FREQUENCE NON LINEAIRE (NON LINEAR FREQUENCY MODULATION)

NRZ : NON-RETOUR AU ZERO (NON RETURN TO ZERO)

CW : ONDE CONTINUE (CONTINUOS WAVE)

TFA : ANALYSE TEMPS-FREQUENCE (TIME FREQUENCY ANALYSIS)

TFIS : IMAGES TEMPS FREQUENCE (TIME FREQUENCY IMAGE)

FFT : TRANSFORMEE DE FOURIER RAPIDE (FAST FOURIER TRANSFORM)

STFT : TRANSFORMEE DE FOURIER A COURT TERME (SHORT TIME FOURIER TRANSFORMATION)

WVD : DISTRIBUTION DE WIGNER-VILLE (WIGNER-VILLE DISTRIBUTION)

CNN : RESEAU NEURONAL CONVOLUTIONNEL (CONVOLUTIONAL NEURAL NETWORK)

PMC : LE PERCEPTRON MULTICOUCHE

LSTM : MEMOIRE LONG/COURT TERME (LONG-SHORT TERM MEMORY)

GPU : PROCESSEUR GRAPHIQUE (GRAPHICS PROCESSING UNIT)

BFGS : METHODE DE BROYDEN-FLETCHER-GOLDFARB-SHANNO

LISTE DES FIGURES :

LISTE DES TABLEAUX

INTRODUCTION GENERALE

Les systèmes de détection et de télémétrie radio (RADAR), comme on les appelait à l'origine, se sont transformés en une vaste gamme d'équipements indispensables à des fins militaires et civiles. Aujourd'hui, il existe de nombreux types de radars conçus pour de nombreuses applications. Radars à balayage, indicateurs de cibles mobiles (MTI), radars météorologiques, chercheurs de missiles guidés, radars pénétrants au sol, radars d'enquête par satellite à ouverture synthétique, radars anticollision automobiles et une multitude d'autres radars définissent l'industrie en pleine croissance d'aujourd'hui.

Dans le domaine militaire, la guerre est numérisée de plus en plus, où la technologie est continuellement évolutive et la motivation d'avoir des moyens performants et efficaces est fondamentale et c'est pour des attaques et des défenses autonomes que les différentes armées du monde utilisent l'intelligence artificielle.

Avec ce développement des systèmes radar, souvent à des fins militaires, le renseignement électronique qui pouvait être exploité dans des signaux radar était d'une grande valeur pour faire face aux menaces potentielles souvent attachées au radar (navires, avions et missiles) et tous cela entre dans le domaine de la guerre électronique (EW : Electronic WAR). Indépendamment de la complexité de chaque système.

L'exploitation des signaux émis par les systèmes d'un adversaire, permet de dégrader sa capacité à utiliser ses systèmes et appliquer des techniques et des procédures pour préserver la capacité des forces amies à utiliser efficacement leurs propres systèmes. Cela n'est pas possible sauf à travers la détection et la classification des signaux radar, mais le problème est de concevoir un tel système qui peut faire la tâche d'une façon rapide et automatique avec de bonne performance.

La reconnaissance de la modulation intra-impulsion du signal radar basé sur le réseau de neurone convolutionnel répond aux critères demandés, on intercepte d'abord le signal radar qui présente un rapport signal sur bruit (SNR). Ce signal a modulation intra-impulsionnelle qui est de plus en plus diversifié, Un vaste type de modulation existe et qui doivent également être reconnus, l'idée est de transformer le signal reçu vers la représentation temps-fréquence pour le représenter par une image puis de faire la classification de ses images grâce au réseau de neurone convolutionnel, chaque signal est classifié selon son type de modulation.

L'objectif principal de notre projet est la reconnaissance et la classification des types de modulation d'une impulsion radar par usage de l'intelligence artificielle basé sur le réseau de neurone convolutionnel.

La suite de ce document est organisée en quatre chapitres. On commence le premier chapitre par une représentation des techniques utilisées pour la compression d'impulsion radar en utilisant déférentes types de modulation tel que la modulation en fréquence et en

phase. Le deuxième chapitre est consacré pour la transformation d'un signal temporel à une représentation temps-fréquence en présentant différentes distributions existantes. Dans le troisième chapitre nous parlerons des réseaux de neurones en général et de réseau de neurones convolutionnel en particulier. Le dernier chapitre présente les résultats des classifications des modulations radar avec un réseau de neurones convolutionnel. L'opération de reconnaissance et de classification des différents types des modulations a été réalisée avec le logiciel MATLAB. A la fin, on termine par une conclusion générale et les futures perspectives.

CHAPITRE 01 : MODULATION DE SIGNAL RADAR

1 CHAPITRE 1 : MODULATION DE SIGNAL RADAR

1.1 Définition :

La réception d'un signal nécessite des antennes dont les dimensions dépendent de la longueur d'onde du signal (de l'ordre de 1/2). Un signal haute fréquence HF sera facilement transmissible (des fréquences F ≥ 100 MHz) soit des longueurs d'onde $\lambda = C/F$ donc λ =3m, soit une antenne de longueur inférieure à 3m.

Par contre, pour les signaux BF (F <300 kHz) la longueur d'onde sera beaucoup plus grande et cela nécessiterait des antennes géants en plus que le signal serait rapidement atténue pour ces fréquences, la solution est donc la modulation.

La modulation est la mise en forme d'un signal électrique contenant une information afin de l'adapter au canal de transmission, Le signal en basse fréquence à transmettre qui contient l'information est appelé le signal modulant. Ce signal est utilisé pour modifier une des caractéristiques d'un signal haute fréquence. Le signal haute fréquence appelé onde porteuse.

Il existe différentes techniques de codage pour la modulation et la compression des impulsions radar comme dans la figure (1.1)

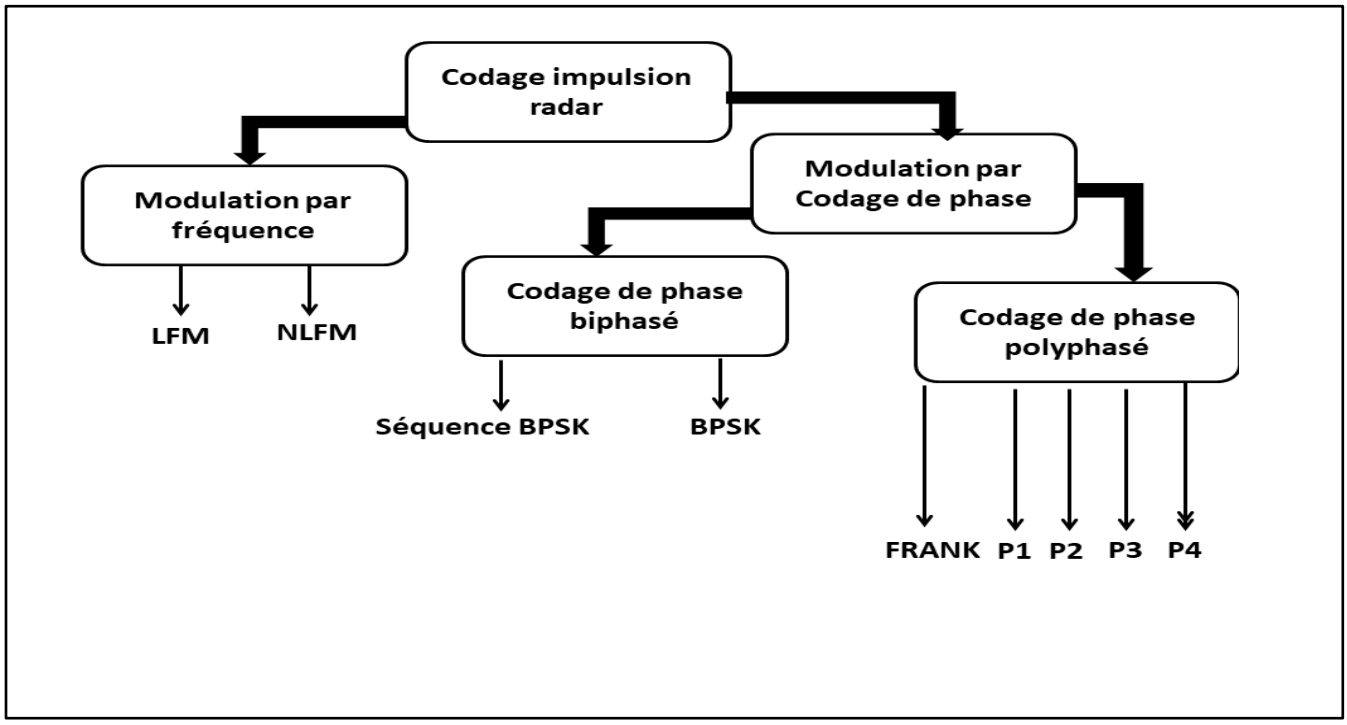


Figure 1.1 – différentes techniques de codage pour la compression des impulsions radar

1.2 Type de modulations :

1.2.1 Modulations avec le code Barker :

Un code de Barker ou séquence de Barker est une suite finie de N valeurs de +1 et -1 avec A = [a0, a1, ..., aN] et N ≥2, l'impulsion de durée T est divisée en n segments de durée $\tau = T/n$ (avec $T \gg \tau$) la phase de chaque segment est modifiée selon ce code elle se décale de 180 dégrée en fonction d'un flux binaire numérique un ''+1'' provoque une transition de phase, et un ''-1'' ne produire pas de transition, Le schéma de codage numérique utilisé dans cette implémentation est appelé non-retour à zéro (NRZ).
Le signal x (t) est une sinusoïde à onde continue (CW), Après un échantillonnage, le signal modulé est criée en ajoutant un code Barker a n bits, la figure (1-2) montre un schéma fonctionnel de base de la conception de l'émetteur.

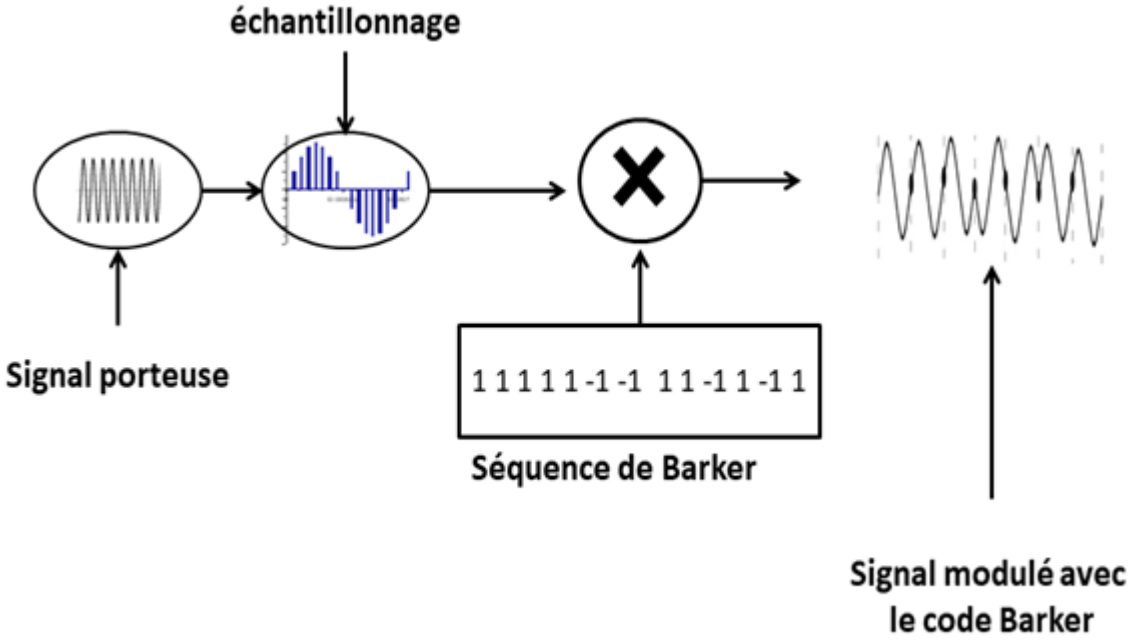


Figure 1.2 – Schéma fonctionnel d'implémentation BPSK

Le code de Barker possède une fonction d'auto corrélation qui présente un pic étroit qui est très avantageux dans telles opérations, sa fonction d'autocorrélation peut prendre que 3 valeurs 0,-1 et N qui est le nombre de bits de la s´séquence, l'équation (1-1) suivante montre les coefficients d'auto corrélation de code :

$$r_k = \sum_{j=0}^{n-k} a_j a_{j+k} \qquad (1.1)$$

$|r_k| \leq 1$ est satisfaite pour $K \neq 0$ et $r_k = r_{-k}$ la figure (1.3) montre la fonction d'auto corrélation d'un code de Barker composé de 7 bits

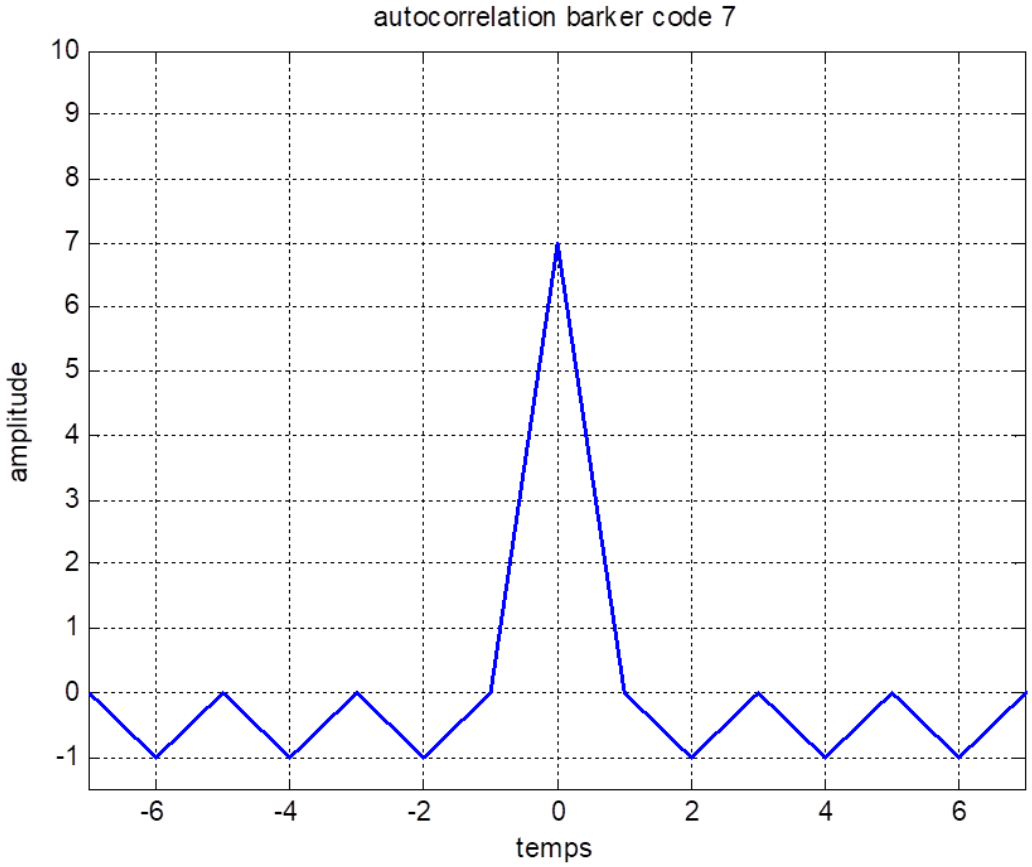


Figure 1.3 – fonction d'auto corrélation de code Barker 7 bits

Il n'existe que neuf séquences de Barker connues qui satisfait les propriétés précédentes. La liste des codes est donnée dans le tableau (1-1) ci-dessous.

Nombre de bits	Code Barker
2	11 ou 00
3	110
4	1110 ou 1101
5	11101
7	1110010
11	11100010010
13	1111100110101

Tableau 1.1 – code de Barker

Pour montrer comment une impulsion est compressée (modulée) les figures (1.4) montre un signal de nature porteuse avec (fc = 1000hz) modulée avec le code Barker=7bit

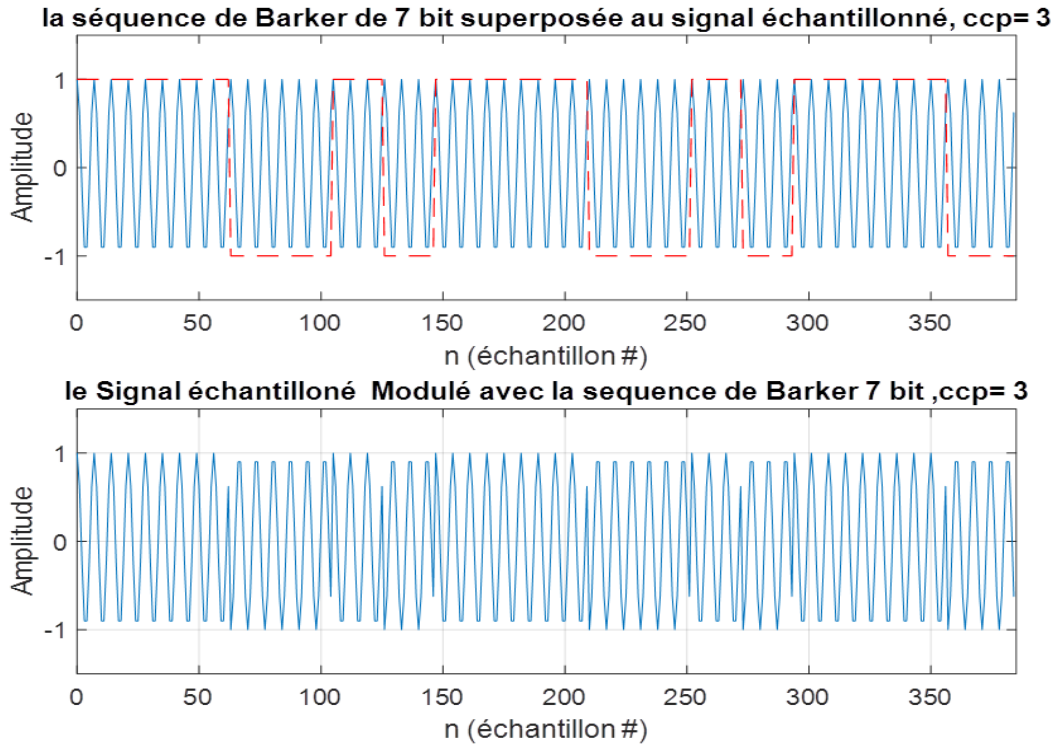


Figure 1.4 – (a) : séquence de Barker 7 bit et la représentation de signal avant modulation
(b) : signal modulée avec le code Barker 7

1.2.2 Modulations avec le code Frank :

Le code de Frank est un format de modulation polyphasée utilisée pour la compression des impulsions, une impulsion de largeur T est divisée en N groupes égaux, chacun de ces groupe est divisés en N sous-impulsions, chacune de ces sous-impulsions possède une largeur de $\delta\tau$.
Cette forme de codage de phase comporte N^2 éléments d´définis comme suit :

$$\emptyset_{i,j} = \frac{2\pi}{N}(i-1)(j-1) \qquad (1\text{-}2)$$

Où i et j varient de 1 à N

La figure ci-dessous (1-5) montre comment une impulsion se devise

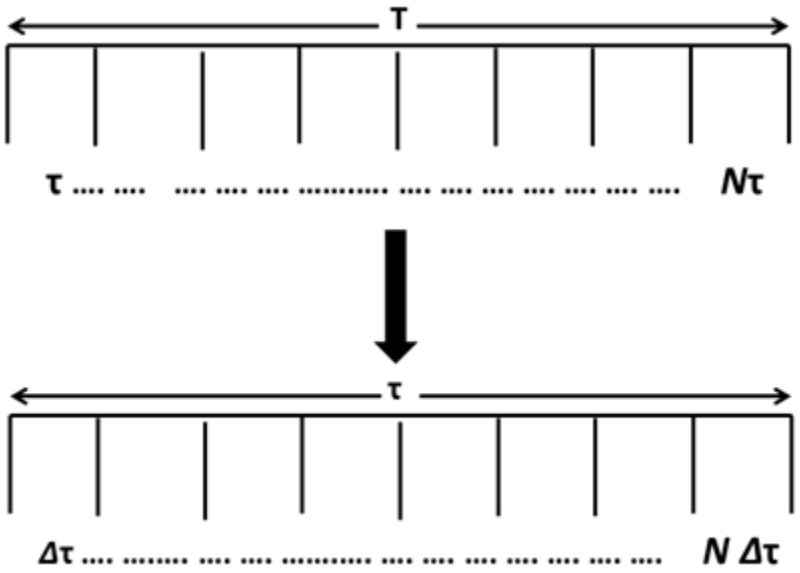


Figure 1.5 – division d'une impulsion pour codage de Frank

La première étape dans le calcul d'un tel code consiste à diviser 360 derges par N, le décalage de phase de chacune des sous-impulsions est tiré de la matrice suivante comme mentionné dans l´équation (1-1)

$$\emptyset_{i,j} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & (N-1) \\ 0 & 2 & 4 & 2(N-1) \\ \vdots & \vdots & \ddots & \vdots \\ 0 & (N-1) & \cdots & (N-1)^2 \end{pmatrix} \delta\emptyset \ldots\ldots (1.1)$$

Chaque rang représente les phases des sous-impulsions d'un groupe. L'incrément de phase fondamental est $\delta\emptyset = \frac{360}{N}$ cet incrément de phase est ensuite multiplié par tous les nombres de la matrice, rappelez-vous, les déphasages supérieurs à 360 degré peuvent être réduits sans influence sur la forme d'onde en soustrayant 360 degré.
Comme exemple un code de Frank N = 4 l'incrément de phase est de $\delta\emptyset = \frac{360}{4} = 90°$ comme montre la matrice et la figure (1-6) ci-dessous

$$\emptyset_{4,4} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 90 & 180 & 270 \\ 0 & 180 & 0 & 180 \\ 0 & 270 & 180 & 90 \end{pmatrix} \ldots\ldots\ldots (1.2)$$

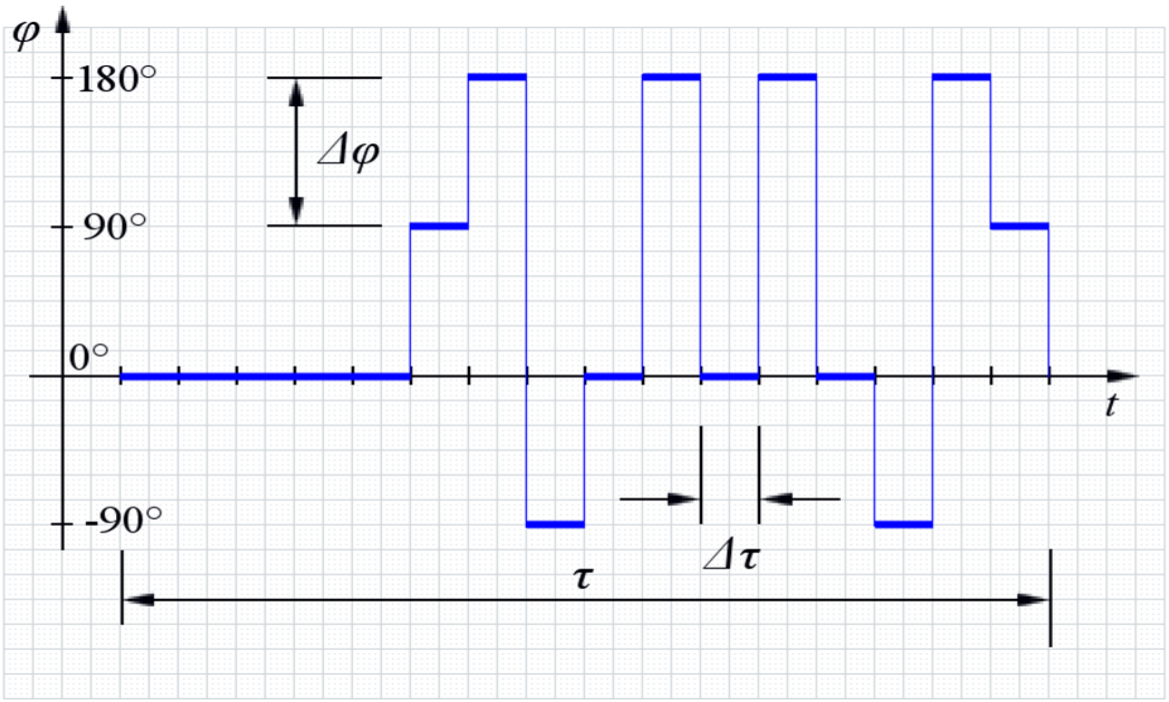


Figure 1.6 – exemple d'un code de Frank avec n=4 (16 phases).

Pour montrer comment une impulsion est compressé (modulé) les figures suivantes montre un signal porteuse avec (fc = 1200hz) avant et après modulation avec le code de Frank avec N=8

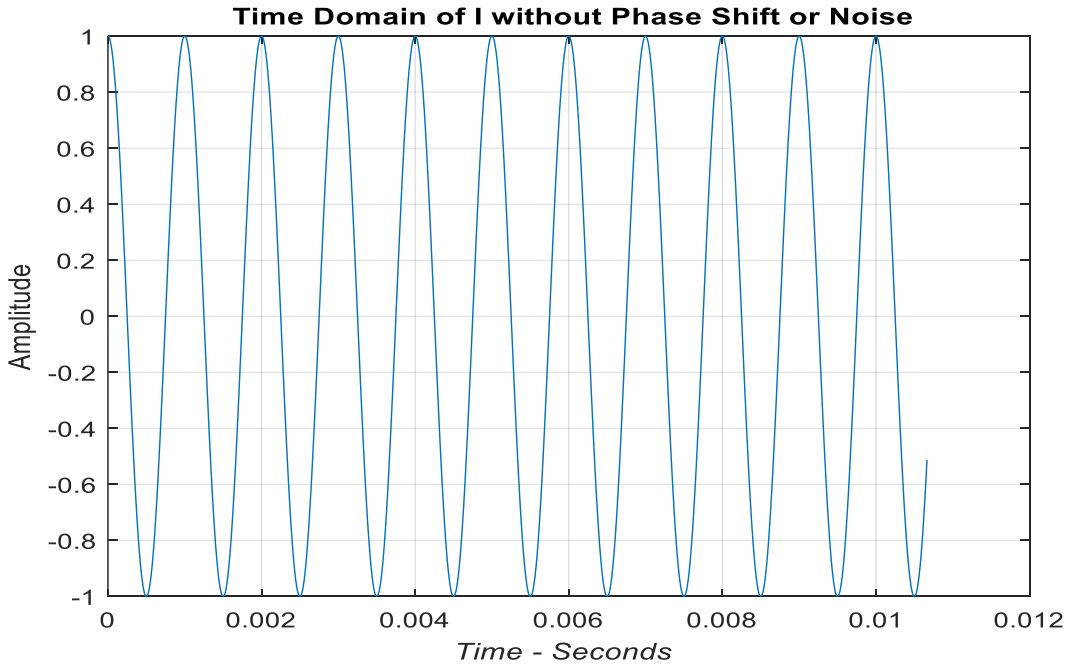


Figure 1.7 – signal porteuse (fc = 1200hz) avant modulation avec le code de Frank

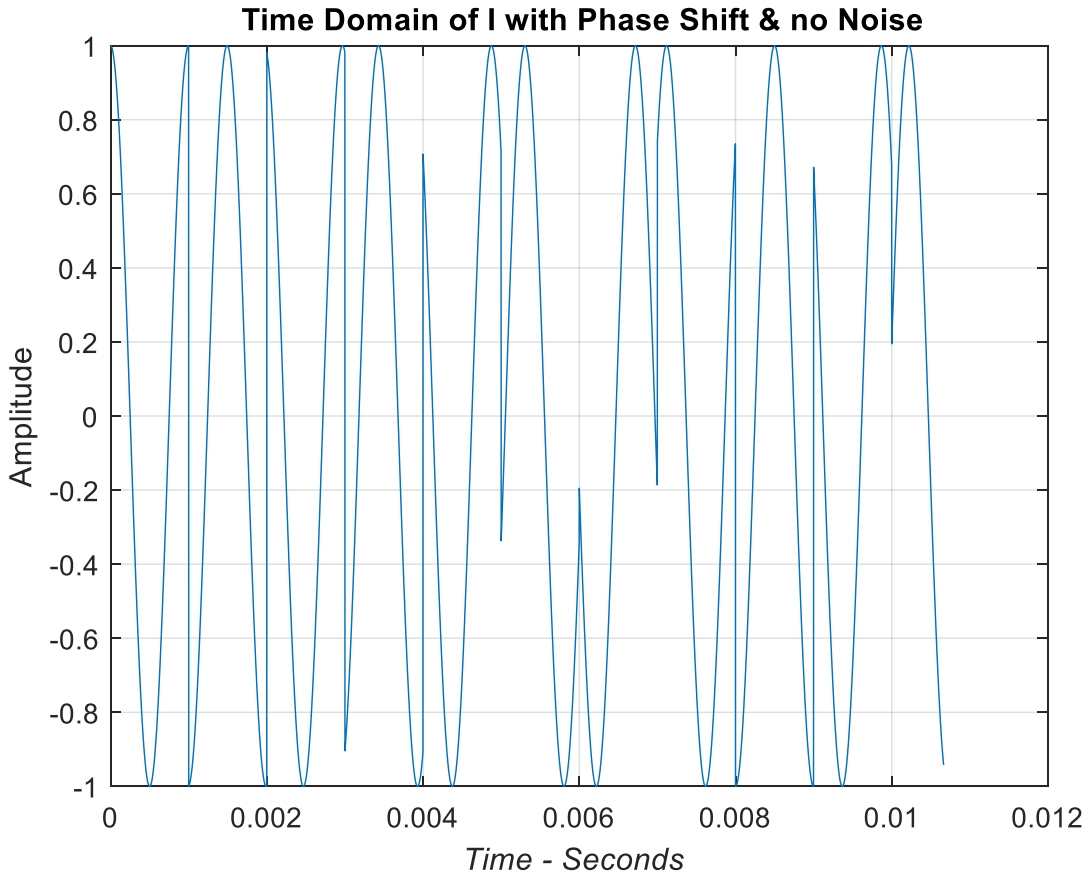


Figure 1.8 – signal (fc = 1200hz) avec modulé avec le code de Frank à N=8 (64 phases)

1.2.3 Modulations avec le code P4 :

Le code P4 se compose des phases discrètes de la forme d'onde de chirp linéaire, ce code ne nécessite pas de matrice carrée comme le code Frank, L'ordre des codes (N) est le nombre de valeurs dans le modèle de phase d´défini comme :

$$\emptyset_i = \frac{\pi(i-1)^2}{N} - \pi(i-1) \qquad (1.3)$$

Où i varie de 1 à N.

Par exemple, le code P4 avec N = 8, en prenant la valeur de phase modulo (2π) est donné par la séquence suivante :

$$\emptyset_8 = \left(0, \frac{-7\pi}{8}, \frac{-12\pi}{8}, \frac{-15\pi}{8}, \frac{-16\pi}{8}, \frac{-15\pi}{8}, \frac{-12\pi}{8}, \frac{-7\pi}{8}\right) \qquad (1.4)$$

La figure (1.9) montre comment la phase change avec la variation de l'indice i.

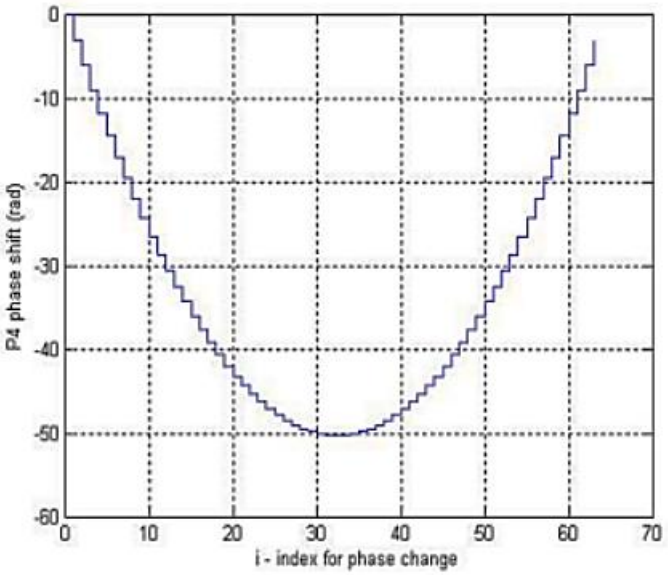


Figure 1.9 –variation de phase avec l'indice i pour le code P4 avec N=64.

Pour montrer comment une impulsion est compressée (modulée) les figures (1.8) et (1.9) montre un signal porteuse avec (fc = 1000hz) modulé avec le code P4 a N=64.

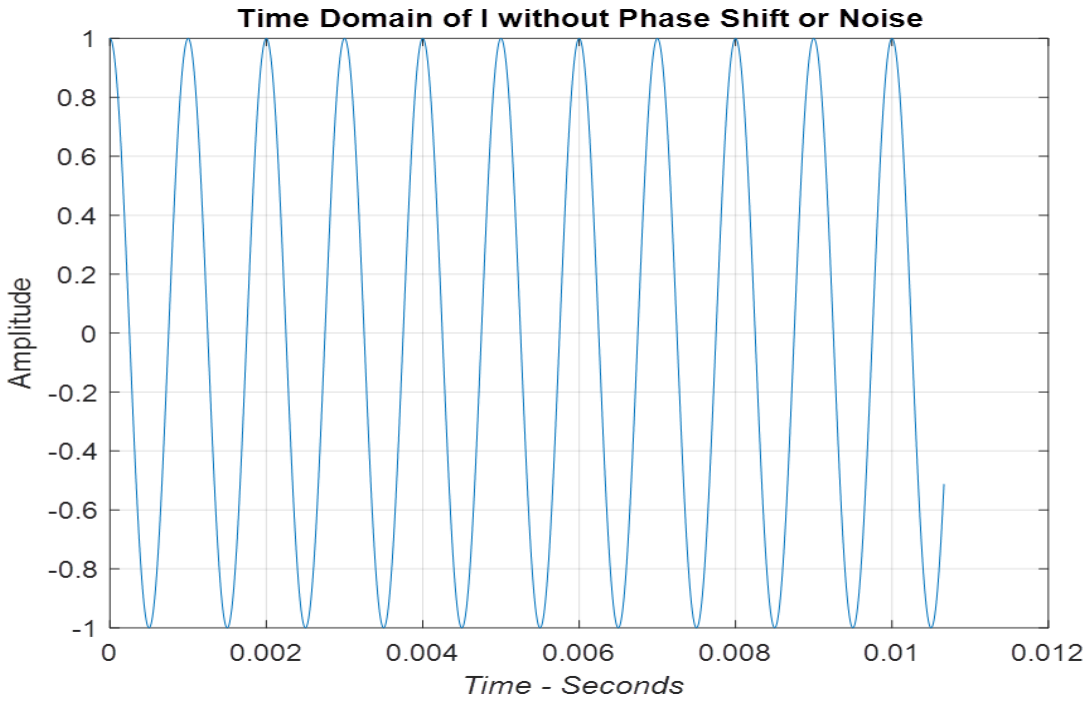


Figure 1.10 – signal porteuse (fc = 1000hz) avant modulation.

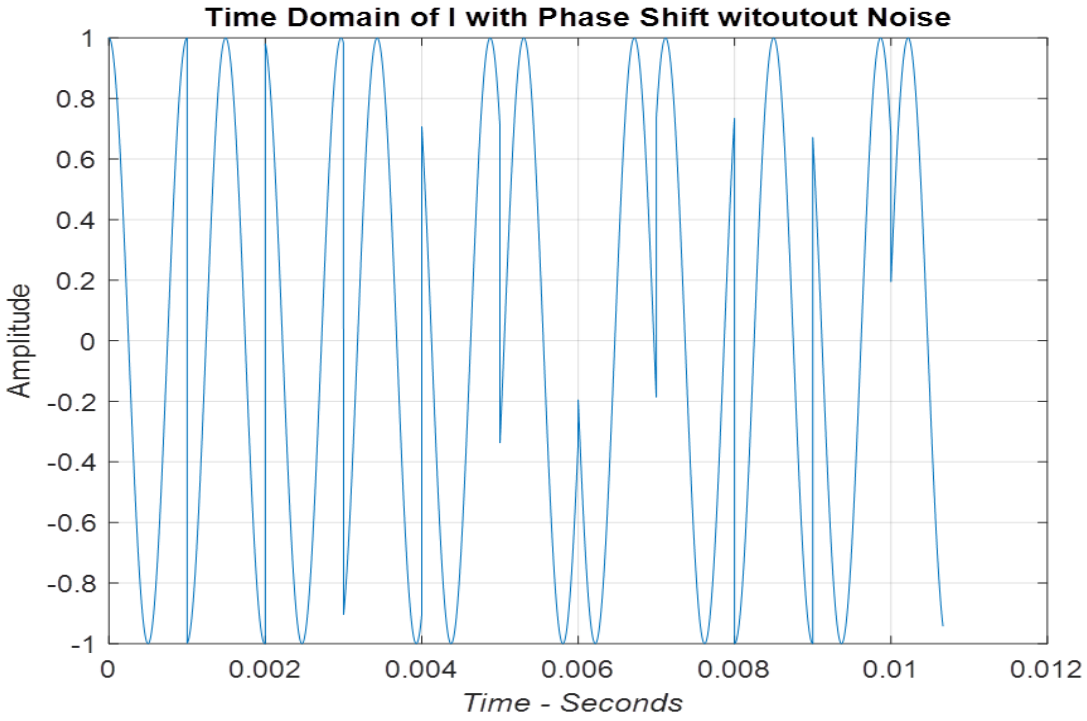


Figure 1.11 – signal porteuse (fc = 1000hz) modulé avec le code P4 à N=64 phases.

1.2.4 Modulations de fréquence FMCW :

Les radars à onde continue (CW) qui utilisent des formes d'onde non modulées ne peuvent pas mesurer la portée d'une cible.
Pour la mesurer on doit varier la fréquence d'émission dans le temps, La modulation de fréquence est alors la solution de problème, La corrélation du signal de retour avec le signal d'émission peut donner une mesure à la fois de la distance et de la fréquence Doppler de la cible, la plus utilisée est la modulation triangulaire.

La modulation triangulaire d'une impulsion radar se compose de deux sections de modulation de fréquence linéaire une de ces modulations avec des pentes positive et l'autre avec une pente négative.

On utilise une forme d'onde triangulaire pour que la fréquence Doppler de la cible détectée puisse être extraite sans ambiguïté en prenant, respectivement, la somme et la différence des deux fréquences de battement.

La fréquence de l'onde transmise pour la première section est donnée par :

$$f_1(t) = f_c - \frac{\Delta F}{2} + \frac{\Delta F}{t_m}.t \qquad (1.5)$$

Avec ΔF est la largeur de bande-passante, f_c est la fréquence de porteuse, t_m est la période de modulation.

La deuxième section est générée de même façon sauf que sa pente est différente de la première, elle est donnée par :

$$f_1(t) = f_c + \frac{\Delta F}{2} - \frac{\Delta F}{t_m}.t \qquad (1.6)$$

Cette forme d'onde est plus facile à mettre en œuvre que la modulation par code de phase tant qu'il n'y a pas de demande stricte sur la linéarité sur la bande passante de modulation son bloc d'implémentation est représenté par la figure (1.12)

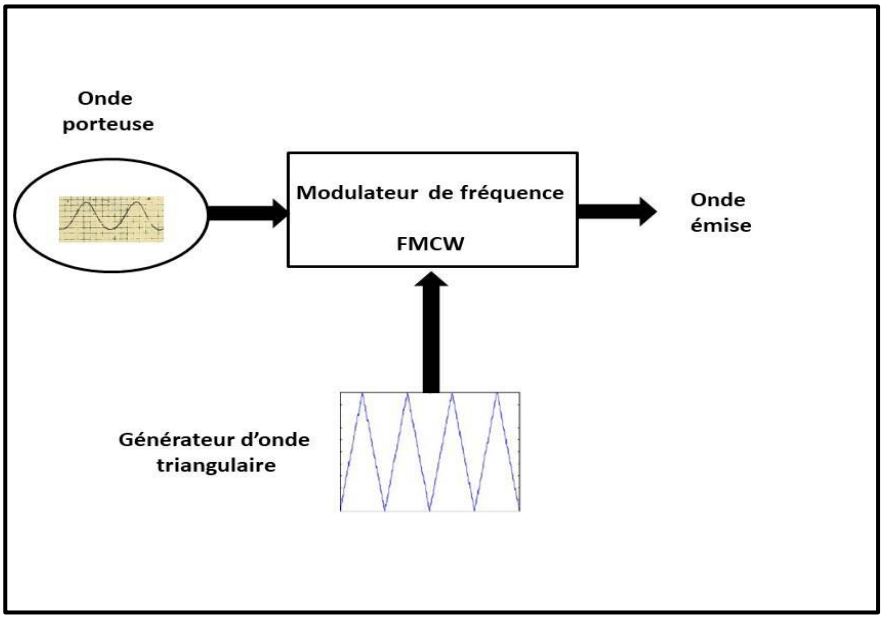


Figure 1.12-Un schéma fonctionnel d'une onde modulé en fréquence avec une onde triangulaire (FMCW)

Pour montrer comment une impulsion est compressée (modulée) les figures (1.13), (1.14) et (1.15) montre un signal porteuse avec (fc = 1000hz) modulé avec l'onde triangulaire a deltaF = 250 HZ et tm = 0.05 s

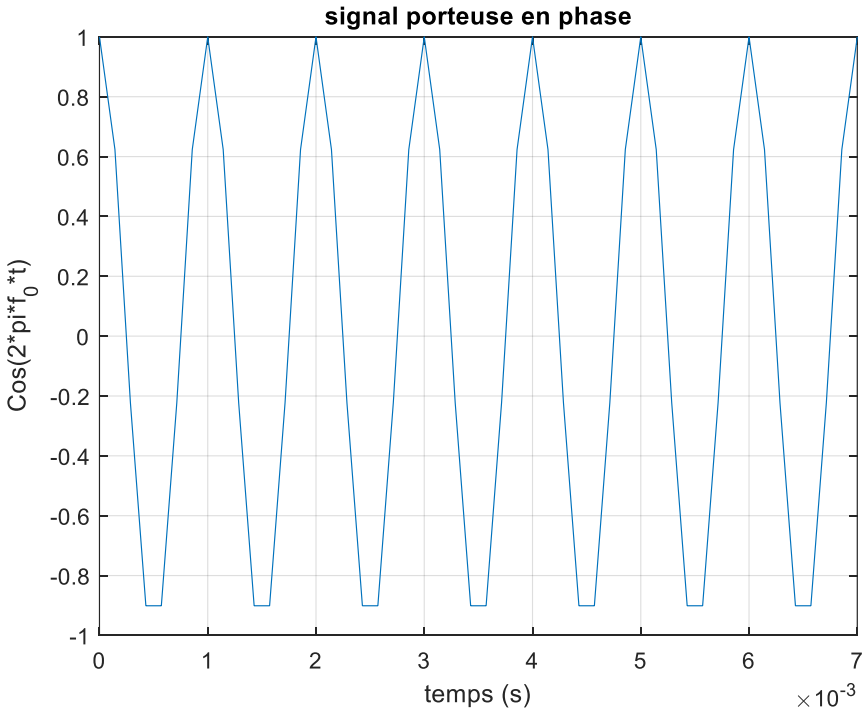


Figure 1.13-le signal porteuse avant la modulation avec fc = 1000hz

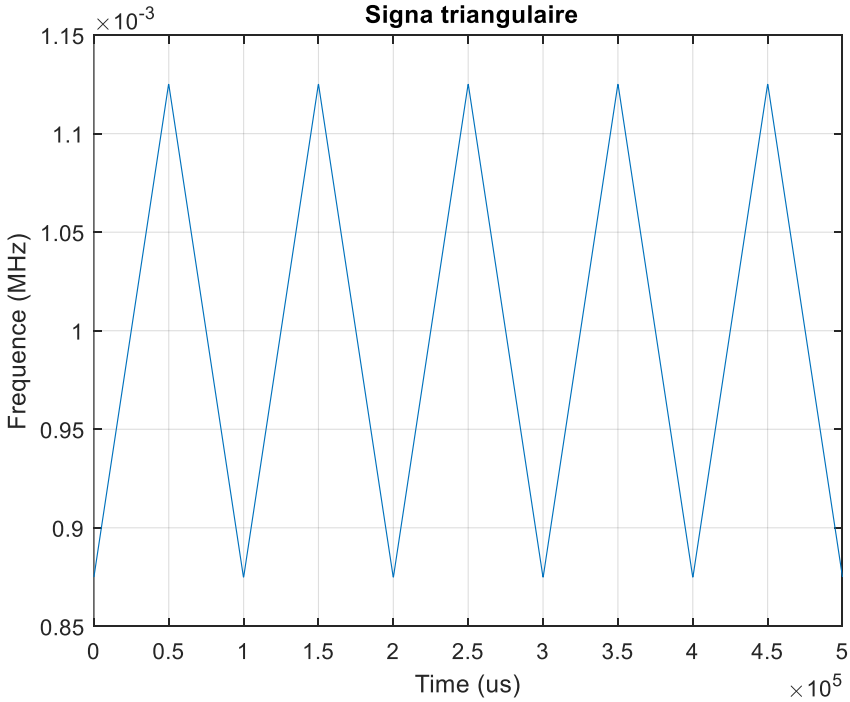


Figure 1.14-l'onde triangulaire avec deltaF = 250 HZ et tm = 0.05 s

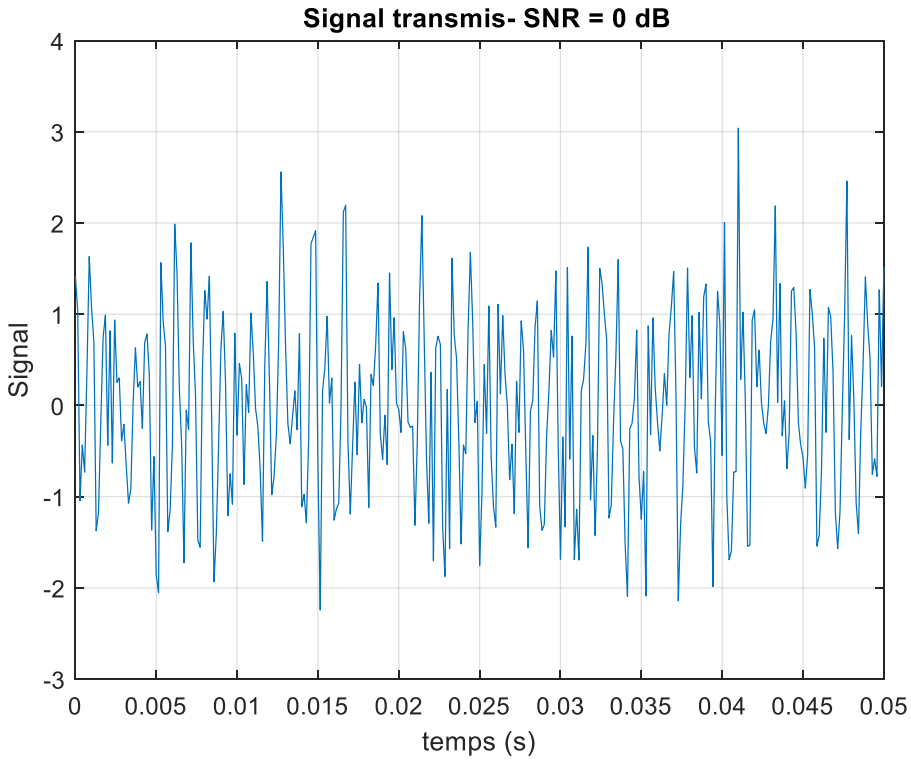


Figure 1.15-le signal transmis avec SNR =0 dB

1.3 Conclusion

On a abordé le Mémoire par ce chapitre car il contient la matière première du projet, la modulation des signaux et le bruit à plusieurs niveau de valeurs permettent de concevoir un environnement très varié contenant des signaux différents. Cela présente un défi pour l'intelligence de système qu'on va construire.

CHAPITRE 02 :

La Transformation Temps-Fréquence

2 CHAPITRE 2 : TRANSFORMEE TEMPS-FREQUENCE

2.1. Introduction :

Les représentations les plus connues pour un signal sont soit la représentation temporelle ou sa représentation fréquentielle, en effet la représentation temporelle n'a pas la description fréquentiel de signal et de même pour l'analyse de Fourier classique qui ne peut pas fournir d'informations localisées dans le temps.

L'analyse temps-fréquence fournirait des informations directes sur les composantes fréquentielles qui se produisent à un instant donné.

Donc L'analyse temps-fréquence qui donne une représentation conjointe temps-fréquence du signal est alors la solution.

L'analyse temps-fréquence a commencé par la transformée de Fourier de courte terme et de la distribution de Wigner-Ville en 1932, qui ont toutes deux contribué à divers domaines d'application de l'ingénierie et de la science à l'aide des progrès de la technologie informatique numérique.

Il existe plusieurs représentations temps-fréquence et on va citer les trois transformée les plus célèbre :

-la transformation de Fourier à court terme

-la distribution temps-fréquence de la classe Cohen

-la distribution de Winner-Ville

2.2 La Transformée de Fourier à court terme :

La transformée de Fourier à court terme ou La transformée de Fourier locale est une transformation liée aux transformées de Fourier classique elle est utilisée comme distribution temps-fréquence pour afficher la variation de la fréquence dans le temps dans l'analyse des signaux, elle fournit une amplitude complexe en fonction de ces deux paramètres.

La STFT est calculé comme une sécession de FFT, en divisant le signal d'entrée en segments, chaque segment comporte un bloc de données fenêtrés, le signal dans chaque fenêtre peut être estimé comme stationnaire ou la fenêtre ''glisse'' ou ''saute''.

La fonction mathématique qui décrit la STFT (Transformée de Fourier à Court Terme) est l'équation (2.1)

$$STFT\big(x(t)\big) = X(\tau, w) = \int_{-\infty}^{+\infty} x(t).w(t-\tau).e^{-j\omega t}\, dt \qquad (2.1)$$

$x(t)$: Signal d'entrée.

$w(t-\tau)$: La fenêtre glissante où w est la fonction de la fenêtre.

ω est la fréquence de signal

$X(\tau, w)$ est la transformée de Fourier

$x(t).w(t-\tau)$ est une fonction complexe représentant la phase et l'amplitude du signal dans le temps et la fréquence .

τ : Le décalage temporel (translation).

Notez que la STFT peut donner des résultats précis lorsque le signal x est concentré au maximum dans un intervalle de temps et un intervalle de fréquence donnés, donc le choix de la fenêtre est critique, la grandeur quadratique du STFT $\left|STFT\big(x(t)\big)\right|^2$ est appelée spectrogramme.

Pour démontrer la performance de la STFT on utilise un signal qui se compose de deux fonction sinus, le premier sinus à 75 Hz commençant à t = 0,2, montant avec une rampe gaussienne jusqu'à une amplitude de 0,8 à t = 0,3, restant à amplitude constante jusqu'à t = 1,1, puis s'éteignant à t = 1,2. Il contient également un sinus 150 Hz, commençant à t = 0,8, s'élevant à une amplitude complète de 1,5 à t = 0,9, restant constant à t = 1,5, puis s'éteignant à t = 1,6.

La figure (2.1) montre le signal de test utilisé :

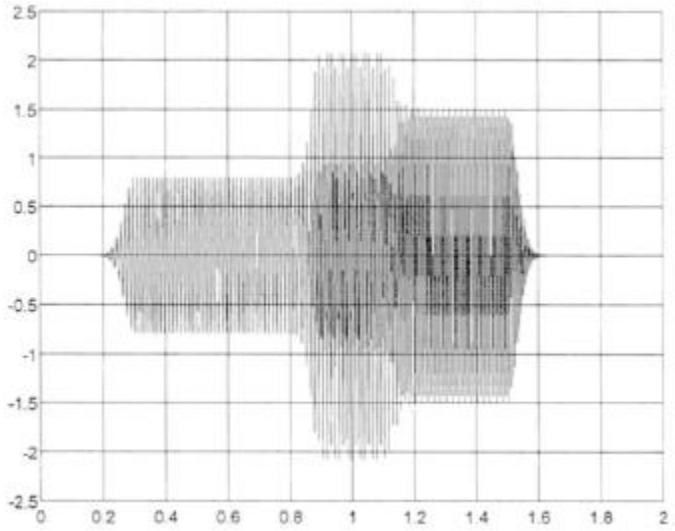


Figure (2.1) : représentation de signal de test avec 75 Hz et 150 Hz

On présente la STFT de signal de test sur la figure (2.2) sachant que la fenêtre utilisée est la fenêtre de Hanning.

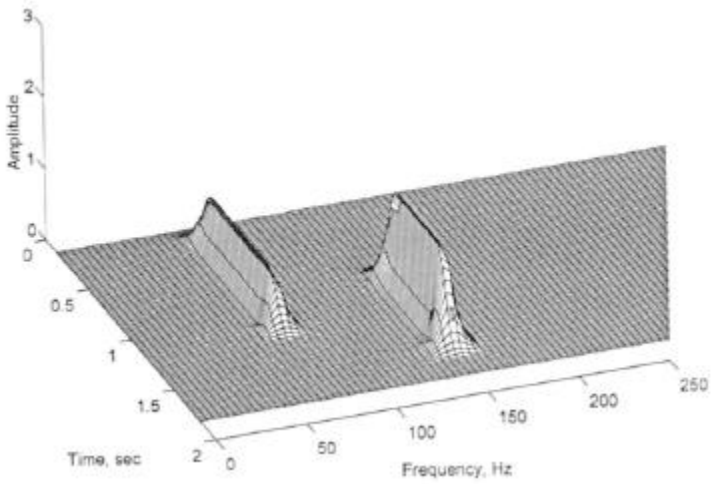


Figure (2.2) : STFT de signal de test.

On peut remarquer sur la figure (2.2) la transformée de Fourier à court terme qu'elle analyse bien le signal de test et elle ne présente pas des termes croisés (interférence) mais il se peut que la STFT élimine quelques amplitudes qui dépasse hauteur de la fenêtre de Hanning choisis.

2.3 La Distribution de Wigner-Ville :

La distribution de Wigner-Ville (WVD), introduite par Wigner en 1932 comme représentation de phase en mécanique statistique quantique et séparément par Ville en 1948 abordant la question d'une fonction de distribution conjointe donne simultanément la représentation d'un signal dans les variables de temps et de fréquence. Le WVD a été noté comme l'une des techniques d'analyse de temps fréquence bilinéaire les plus utiles pour le traitement du signal, elle représente la concentration d'énergie dans le plan temps fréquence .

La WVD pour une fonction continue d'une seule dimension (un signal d'entrée) $x(t)$ est donnée par :

$$W_x(t,w) = \int_{-\infty}^{+\infty} x\left(t+\frac{\tau}{2}\right).x^*(t-\frac{\tau}{2})\,e^{(-jw\tau)}d\tau \quad \text{.....(2.2)}$$

Avec :

- $W_x(t,w)$: la fonction de Wigner
- t : variable de temps
- $t+\frac{\tau}{2}$: le temps moyen
- $\tau = t_1 - t_2$: le décalage dans le temps
- w : variable de fréquence $w = 2\text{pi}f$
- x^* : indique le conjugué de x(t)

L'inconvénient de la distribution WVD c'est qu'elle souffre de termes croisés forts (interférences), La transformée de pseudo Wigner-Ville lissée est une variante de la transformée de Wigner- Ville classique avec des termes croisés réduits, Elle utilise les propriétés du signal analytique x(t) associe au signal réel X R(t) et de deux fenêtres dites de lissage F(w) et g(t), la première opérante un lissage fréquentiel et la seconde agissant temporellement.

Ainsi, la transformée de Wigner-Ville lissée (SPWVD) s'écrit :

$$SPWD_x^g H(t,f) = \int_{-\infty}^{+\infty} g(t)\,H(f)\,x\left(t+\frac{\tau}{2}\right).x^*(t-\frac{\tau}{2})\,e^{(-jw\tau)}d\tau \text{.....(2.3)}$$

De la même façon pour démontrer la performance de la WVD on utilise le signal de test utilisé pour la STFT.

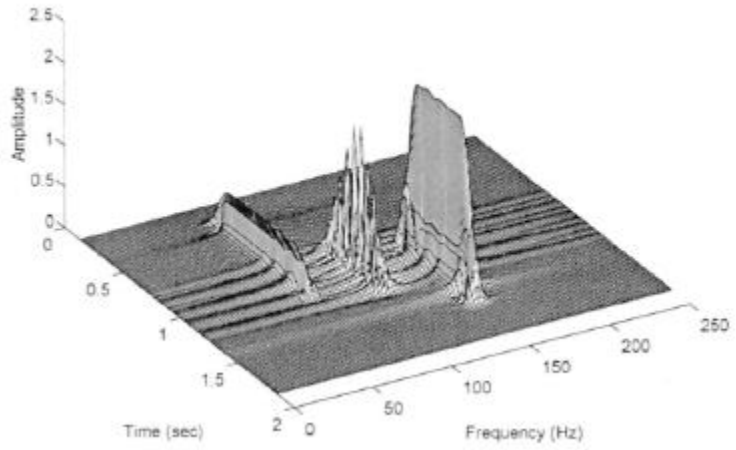


Figure (2.3) : WVD de signal de test.

Pour la WVD on peut tirer comme remarque qu'elle présente des termes croisés, sur la figure on voit les deux spectres de fréquence pour les deux sinus 75 Hz et 150 Hz mais plus une interférence au milieu.

2.4 La distribution de Chois-William :

La fonction de distribution Choi – Williams est l'un des membres de la fonction de distribution de classe de Cohen. Elle a été proposée pour la première fois par Hyung-Ill Choi et William J. Williams en 1989.

La forme générale de la fonction de distribution de classe de Cohen est donnée par l'équation (2.) :

$$C_f(\mathrm{t},\mathrm{w},\emptyset) = \frac{1}{2\pi}\iiint e^{j(\xi\mu-\tau w-\xi t)}\ \emptyset(\xi,\tau)\ A(\mu,\tau)\ d\mu\ d\tau\ d\xi \ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.4)$$

Avec : $\emptyset(\xi,\tau)$: est la fonction noyau en anglais Kernel function

$$A(\mu,\tau) = x\left(\mu+\frac{\tau}{2}\right).x^*(\mu+\frac{\tau}{2}) \qquad (2.5)$$

Pour $x(\mu)$ est le signal temporel, et $x^*(\mu)$ est son conjugué complexe.

Pour que le calcul peut minimiser les termes croisés tout en conservant les propriétés souhaitables de signal, Choi et Williams ont choisis soigneusement le noyau dans l'équation (2.4), La distribution (CWD) utilise un noyau de pondération exponentielle afin de réduire les composantes transversales de la distribution.

La fonction noyau qui donne la distribution Choi-Williams est :

$$\emptyset(\xi,\tau) = e^{-\xi^2\tau^2/\sigma} \text{.....(2.6)}$$

Où σ (σ> 0) est un facteur d'échelle.

En substituant ce noyau dans (2.4), la CWD du signal d'entrée $x(t)$ est donnée comme :

$$CWD_x(\mathrm{t,w}) = \int_{\tau=-\infty}^{+\infty} e^{jw\tau} \left[\int_{\mu=-\infty}^{+\infty} \sqrt{\frac{\sigma}{4\pi\tau^2}}\ G(\mu,\tau)\ A(\mu,\tau)\, d\mu \right] d\tau \ \text{... ... ... (2.7)}$$

Où $G(\mu,\tau) = e^{(\mu-t)^2/(4\tau^2)}$

Avec t est la variable de temps, $w = 2\pi f$ est la variable de fréquence angulaire, et σ est un facteur d'échelle à valeur positive.

La figure ci-dessous présente la CWD pour le signal de test utilisé pour les deux précédentes transformées.

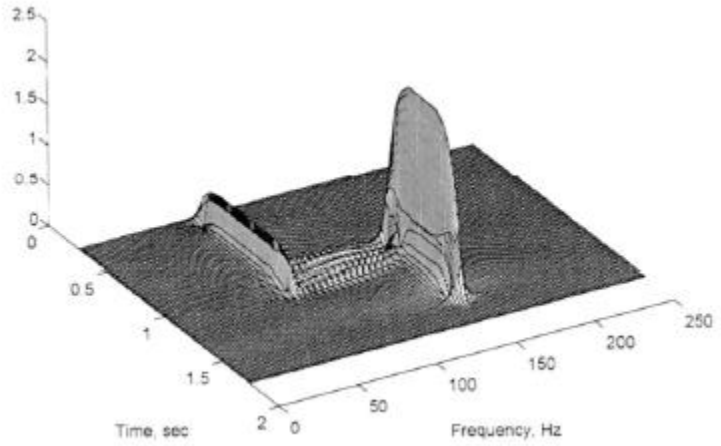


Figure (2.4) : CWD de signal de test avec $\sigma = 7$.

La fonction de distribution Choi – Williams montré sur la figure (2.4) présente elle aussi bien le signal de test et elle n'élimine pas d'amplitude, mais elle aussi présente des termes croisés mais qui n'affecte pas vraiment la représentation temps-fréquence.

2.5 Conclusion :

L'absence de termes croisés forts dans le plan temps-fréquence est essentiels, il permet de déterminer plus facilement le type de modulation, pour cela il faut bien choisir la transformée temps fréquence.

Comme la STFT qui présente bien la distribution temps-fréquence mais qu'elle élimine les amplitudes supérieures à la fenêtre, on peut dire alors que le choix de la fenêtre est critique dans le cas de STFT.

Pour la WVD et la CWD elle souffre des termes croisés mais elles gardent la vraie amplitude de signal, dans le cas de WVD Plus le signal est composé, plus il y a des termes croisés par contre pour la CWD qui peut éliminer la majorité des interférences grâce à ce noyau exponentiel.

Chapitre 03 :

Réseau Neuronal Convolutionnel

3 CHAPITRE 03 : RESEAU DE NEURONES

3.1. Historique :

L'Intelligence Artificielle, branche de l'Informatique fondamentale s'est développée avec pour objectif la simulation des comportements du cerveau humain. Les premières tentatives de modélisation du cerveau sont anciennes et précèdent même l'ère informatique. C'est en 1943 que Mc Culloch (neurophysiologiste) et Pitts (logicien) ont proposé les premières notions de neurone formel. Ce concept fut ensuite mis en réseau avec une couche d'entrée et une sortie par Rosenblatt en 1959 pour simuler le fonctionnement rétinien et tacher de reconnaître des formes. C'est l'origine du perceptron. Cette approche dite connexionniste a atteint ses limites technologiques, compte tenu de la puissance de calcul de l'époque, mais aussi théoriques au début des années 70 [7]

L'approche connexionniste à connaissance répartie a alors été supplantée par une approche symbolique qui promouvait les systèmes experts à connaissance localisée dont L'objectif était d'automatiser le principe de l'expertise humaine en associant trois concepts :

- une base de connaissance dans laquelle sont regroupées les connaissances d'experts humains sous forme de propositions logiques élémentaires ou plus élaborées en utilisant des quantificateurs (logique du premier ordre).
- une base de faits contenant les observations du cas à traiter comme, par exemple, des résultats d'examens, d'analyses de sang, de salive pour des applications biomédicales de choix d'un antibiotique,
- un moteur d'inférence chargé d'appliquer les règles expertes sur la base de faits afin d'en déduire de nouveaux faits jusqu'à la réalisation d'un objectif comme le choix du traitement d'une infection bactérienne. Face aux difficultés rencontrées lors de la modélisation des connaissances d'un expert humain, au volume considérable des bases qui en découlaient et au caractère exponentiel de la complexité des algorithmes d'inférence mis en jeu, cette approche s'est éteinte avec les années 80. Il a été montré que les systèmes basés sur le calcul des prédicats du premier ordre conduisaient à des problèmes non complets.

L'essor technologique et quelques avancées théoriques :

Estimation du gradient par rétro-propagation de l'erreur .

Analogie de la phase d'apprentissage avec les modèles markoviens de systèmes de particules de la mécanique statistique .

Au début des années 80 ont permis de relancer l'approche connexionniste. Celle-ci a connu au début des années 90 un développement considérable si l'on considère le nombre de publications et de congrès qui lui ont été consacrés mais aussi les domaines d'applications très divers où elle apparaît. La motivation initiale de simulation du cortex cérébral a été rapidement abandonnée alors que les méthodes qui en découlaient ont trouvé leur propre intérêt de développement méthodologique et leurs champs d'applications. Remis en veilleuse depuis le milieu des années 90 au profit d'autres algorithmes d'apprentissage machine ou plutôt statistique : boosting, support vector machine..., les réseaux de neurones connaissent un regain d'intérêt et même un énorme battage médiatique sous l'appellation d'apprentissage profond (deep learning). La taille des bases de données, notamment celles d'images issues d'internet, associée à la puissance de calcul disponible, permettent d'estimer les millions de paramètres du percetron accumulant des dizaines voire centaines de couches de neurones aux propriétés très spécifiques. Ce succès médiatique est la conséquence des résultats spectaculaires obtenus par ces réseaux en reconnaissance d'image, jeux de go, traitement du langage naturel...

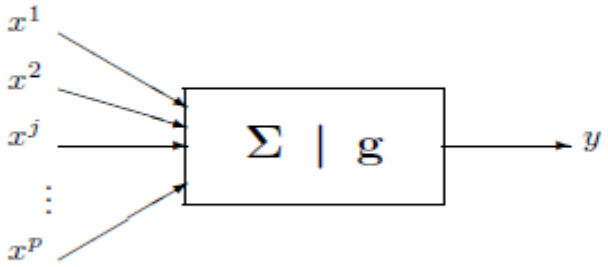


Figure (3.1) - Représentation d'un neurone formel

3.2. Réseaux de neurones :

Un réseau neuronal est l'association en un graphe plus ou moins complexe d'objets élémentaires ou bien des neurones formels. Les principaux réseaux se distinguent par l'organisation du graphe (en couches) c'est-à-dire leur architecture, son niveau de complexité (le nombre de neurones, présence ou non de boucles de rétroaction dans le réseau), par le type des neurones (leurs fonctions de transition ou d'activation) et enfin par l'objectif visé : apprentissage supervisé ou non, optimisation, systèmes dynamiques...

3.3. Neurone formel :

De façon très réductrice, un neurone biologique est une cellule qui se caractérise par

- des synapses, les points de connexion avec les autres neurones, fibres nerveuses ou musculaires ;
- des dendrites ou entrées des neurones
- les axones, ou sorties du neurone vers d'autres neurones ou fibres musculaires
- le noyau qui active les sorties en fonction des stimulations en entrée. Par analogie, le neurone formel est un modèle qui se caractérise par un état interne S, des signaux d'entrée $x_1, \dots, x_p$ et une fonction d'activation g

$$s = h(\alpha_1 \dots, \alpha_p) = g(\alpha_0 + \sum_{j=1}^{p} \alpha_j x_j) = g(\alpha_0 + \alpha'_x). \qquad (3.1)$$

La fonction d'activation opère une transformation d'une combinaison affine des signaux d'entrée, α_0 terme constant étant appelé le biais du neurone. Cette combinaison est déterminée par un vecteur de poids $[\alpha_0 \dots\dots \alpha_p]$ associé à chaque neurone et dont les valeurs sont estimées dans la phase d'apprentissage.

Ils constituent la mémoire ou connaissance répartie du réseau. Les différents types de neurones se distinguent par la nature g de leur fonction d'activation. Les principaux types sont(08) :

- linéaire g est la fonction identité
- seuil g(x) $= 1_{[0,+\infty[}$ (x)
- sigmoïde g(x) = $1/(1 + e^{x})$

- ReLU g(x) = max(0,x) (rectified linear unit)
- softmax $g(x)_j = \frac{e^{xj}}{\sum_{k-1}^{k} e^{xk}}$ pour tout $k \in \{1.....k\}$
- radiale $g(x) = \sqrt{1/2\pi} e^{-x^2/2}$
- stochastique g(x) = 1 avec la probabilité $1/(1 + e^{-\frac{x}{H}})$, sinon 0 (H intervient comme une température dans un algorithme de recuit simulé)

3.4. Perceptron multicouche :

Nous ne nous intéresserons dans cette section qu'à une structure élémentaire de réseau, celle dite statique ne présentent pas de boucle de rétroaction et dans un but d'apprentissage supervisé. Les systèmes dynamiques, avec boucle de rétroaction, les réseaux récurrents (LSTM) ainsi que les cartes de Kohonen ou cartes auto-organisatrices pour la classification non supervisée ne sont pas abordés.

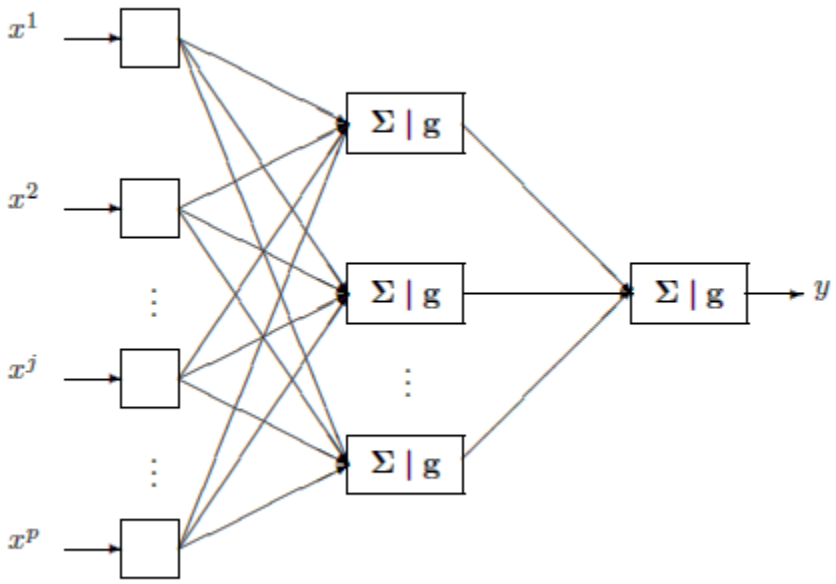


FIGURE (3.2) – Exemple de perceptron multicouche élémentaire avec une couche cachée et une couche de sortie.

3.4.1. Architecture :

Le perceptron multicouche (PMC) est un réseau composé de couches successives. Une couche est un ensemble de neurones n'ayant pas de connexion entre eux, Une couche d'entrée lit les signaux entrant, un neurone par entrée, x_i une couche en sortie fournit la réponse du système. Selon les auteurs, la couche d'entrée qui n'introduit aucune modification n'est pas comptabilisée. (09)

Une ou plusieurs couches cachées participent au transfert. Dans un perceptron, un neurone d'une couche cachée est connecté en entrée à chacun des neurones de la couche précédente et en sortie à chaque neurone de la couche suivante.

3.4.2. Fonction de transfert :

Par souci de cohérence, les mêmes notations ont été conservées à travers les différents chapitres. Ainsi, les entrées d'un réseau sont encore notées $x_1, \dots x_p$comme les variables explicatives d'un modèle tandis que les poids des entrées sont des paramètres α; β à estimer lors de la procédure d'apprentissage et que la sortie est la variable Y à expliquer ou cible du modèle.

Un perceptron multicouche réalise donc une transformation des variables d'entrée :

$Y = f(x_1, \dots x_p; \alpha)$

Où α est le vecteur contenant chacun des paramètres α_{jkl}de la $j^{ème}$ entrée du $k^{ème}$ neurone de la $l^{ème}$ couche ; la couche d'entrée ($l = 0$) n'est pas paramétrée, elle ne fait que distribuer les entrées sur tous les neurones de la couche suivante.

Un théorème dit d'approximation universelle montre que cette structure élémentaire à une seule couche cachée est suffisante pour prendre en compte les problèmes classiques de modélisation ou apprentissage statistique. En effet, toute fonction régulière peut être approchée uniformément avec une précision arbitraire et dans un domaine fini de l'espace de ses variables, par un réseau de neurones comportant une couche de neurones cachés en nombre fini possédant tous la même fonction d'activation et un neurone de sortie linéaire. Attention, ce résultat, qui semble contradictoire avec les structures d'apprentissage

profond, est théorique, il masque des difficultés d'apprentissage et de stabilité pour des problèmes complexes en très grande dimension.

De façon usuelle et en régression (Y quantitative), la dernière couche est constituée d'un seul neurone muni de la fonction d'activation identité tandis que les autres neurones (couche cachée) sont munis de la fonction sigmoïde.

En classification binaire, le neurone de sortie est muni également de la fonction sigmoïde tandis que dans le cas d'une discrimination à m classes (Y qualitative), le neurone de sortie intègre une fonction d'activation softmax à valeurs dans R_m et de somme unit. Ces m valeurs sont assimilables à des probabilités d'appartenance à une classe.

Ainsi, en régression avec un perceptron à une couche cachée de q neurones et un neurone de sortie, cette fonction s'écrit : $Y = f(x, \alpha, \beta) = \beta_0 + \beta'_z$ (3.2)

Avec $z_k = \mathrm{g}(\alpha_0 + x\alpha_\mathrm{k}); k = 1, \dots, q;$

3.4.3. Apprentissage :

Supposons que l'on dispose d'une base d'apprentissage de taille n d'observations $(x_i^1, \dots., x_i^p; y_i)$ des variables explicative $x^1, \dots, x^p$ et de la variable à prévoir Y. Considérons le cas le plus simple de la régression avec un réseau constitué d'un neurone de sortie linéaire et d'une couche à q neurones dont les paramètres sont optimisés par moindres carrés. Ceci se généralise à toute fonction perte dérivable et donc à la discrimination à m classes.

L'apprentissage est l'estimation des paramètres $\alpha_j = 0, p\,; k = 1, q \;\; et \;\; \beta_k = 0, q$ par minimisation de la fonction perte quadratique ou de celle d'une fonction d'entropie en classification :

$$Q(\alpha; \beta) = \sum_{i=0}^{n} Qi = \sum_{i=1}^{n} [yi - f(x; \alpha, \beta]^2 \qquad (3.3)$$

Différents algorithmes d'optimisation sont proposés, ils sont généralement basés sur une évaluation du gradient par rétro-propagation.

3.4.3.1. Rétro-propagation de l'erreur :

Il s'agit donc d'évaluer la dérivée de la fonction coût en une observation et par rapport aux différents paramètres.

Soit z_{ki} = g(α_{k0} +α_k,x_i) et z_i = {z_{1i}, … . , z_{qi}}. (3.4)

Les dérivées partielles de la fonction perte quadratique s'écrivent :

$$\frac{\partial Qi}{\partial \beta k} = -2(y_i - \Phi(x_i))(\beta' z_i) z_{ki} = \delta_i z_{ki} \qquad (3.5)$$

$$\frac{\partial Q_i}{\partial \alpha_{kj}} = -2(y_i - \Phi(x_i))(\beta' z_i)\beta_{kg'}(\alpha'_k x_i) x_{ip} = S_{ki} x_{ip} \qquad (3.6)$$

Les termes δ_i et S_{ki} sont respectivement les termes d'erreur du modèle courant à la sortie et sur chaque neurone caché. Ces termes d'erreur vérifient les équations dites de rétro-propagation :

$$S_{ki} = g'(\alpha'_k x_i)\, \beta_k \delta_i \qquad (3.7)$$

Dont les termes sont évalués en deux passes. Une passe avant, avec les valeurs courantes des poids : l'application des différentes entrées xi au réseau permet de déterminer les valeurs ajustées $f(x_i)$ La passe retour permet ensuite de déterminer les δ_i qui sont rétro-propagés afin de calculer les s_{ki} et ainsi obtenir les évaluations des gradients.

3.4.3.2. Algorithmes d'optimisation :

Sachant évaluer les gradients, différents algorithmes, plus ou moins sophistiqués, sont implémentés. Le plus élémentaire est une utilisation itérative du gradient : en tout point de l'espace des paramètres, le vecteur gradient de Q pointe dans la direction de l'erreur croissante. Pour faire décroître Q il suffit donc de se déplacer en sens contraire. Il s'agit d'un algorithme itératif modifiant les poids de chaque neurone selon :

$$\beta_k^{r+1} = \beta_k^r - \boldsymbol{\tau} \sum_{i=1}^{n} \frac{\partial Q_i}{\partial \beta_k^r} \qquad (3.8)$$

$$\alpha_{kp}^{r+1} = \alpha_{kp}^r - \boldsymbol{\tau} \sum_{i=1}^{n} \frac{\partial Q_i}{\partial \alpha_{kp}^r} \qquad (3.9)$$

Le coefficient de proportionnalité $\boldsymbol{\tau}$ est appelé le taux d'apprentissage. Il peut être fixe, à déterminer par l'utilisateur, ou encore varier en cours d'exécution selon certaines heuristiques.

Il paraît en effet intuitivement raisonnable que grand au début pour aller plus vite, ce taux décroisse pour aboutir à un réglage plus fin au fur et à mesure que le système s'approche d'une solution.

Si l'espace mémoire est suffisant, une version accélérée de l'algorithme fait intervenir à chaque itération un ensemble (batch) d'observations pour moyenner les gradients et mises à jour des poids.

Bien d'autres méthodes d'optimisation ont été adaptées à l'apprentissage d'un réseau : méthodes du gradient avec second ordre utilisant une approximation itérative de la matrice hessienne (algorithme BFGS, de Levenberg- Marquardt) ou encore une évaluation implicite de cette matrice par la méthode dite du gradient conjugué. La littérature sur le sujet propose quantités de recettes destinées à améliorer la vitesse de convergence de l'algorithme ou bien lui éviter de rester collé à une solution locale défavorable. D'autres heuristiques proposent d'ajouter un terme d'inertie afin d'éviter des oscillations de l'algorithme.

D'autres algorithmes encore sont des versions adaptatives. Lorsque de nouvelles observations sont proposées une à une au réseau. Dans ce dernier type d'algorithme, des propriétés de dynamique markovienne (processus ergodique convergeant vers la mesure stationnaire) impliquent une convergence presque sûre : la probabilité d'atteindre une précision fixée a priori tend vers 1 lorsque la taille de l'échantillon d'apprentissage tend vers l'infini.

On pourra se reporter à l'abondante littérature sur le sujet pour obtenir des précisions sur les algorithmes d'apprentissage et leurs nombreuses variantes. Il est important de rappeler la liste des choix qui sont laissés à l'utilisateur.

En effet, même si les logiciels proposent des valeurs par défaut, il est fréquent que cet algorithme connaisse quelques soucis de convergence.

3.4.4. Contrôle de la complexité

Régularisation

Dans les réseaux élémentaires, une option simple pour éviter le sur-apprentissage consiste à introduire un terme de pénalisation ou régularisation, comme en régression ridge, dans le critère à optimiser.

Celui-ci devient alors : $Q(\gamma) + \gamma\|\Phi\|^2$ Plus la valeur du paramètre γ (decay) est importante et moins les poids des entrées des neurones peuvent prendre des valeurs chaotiques contribuant ainsi à limiter les risques de sur-apprentissage.

Choix des paramètres

L'utilisateur doit donc déterminer

1. les variables d'entrée et la variable de sortie ; leur faire subir comme pour toutes méthodes statistiques, d'éventuelles transformations, normalisations.

2. L'architecture du réseau : le nombre de couches cachées qui correspond à une aptitude à traiter des problèmes de non-linéarité, le nombre de neurones par couche cachée. Ces deux choix conditionnent directement le nombre de paramètres (de poids) à estimer et donc la complexité du modèle. Ils participent à la recherche d'un bon compromis biais/variance c'est-à-dire à l'équilibre entre qualité d'apprentissage et qualité de prévision.

3. Trois autres paramètres interviennent également sur ce compromis : le nombre maximum d'itérations, l'erreur maximum tolérée et un terme éventuel de régularisation ridge (decay).

4. Le taux d'apprentissage ainsi qu'une éventuelle stratégie d'évolution de celui-ci.

5. la taille des ensembles ou batchs d'observations considérés à chaque itération.

En pratique, tous ces paramètres ne peuvent être réglés simultanément par l'utilisateur. Celui-ci est confronté à des choix concernant principalement le contrôle du sur-

apprentissage : limiter le nombre de neurones ou la durée d'apprentissage ou encore augmenter le coefficient de pénalisation de la norme des paramètres. Ceci nécessite de déterminer un mode d'estimation de l'erreur : échantillon validation ou test, validation croisée ou bootstrap.

Une stratégie simple et sans doute efficace consiste à introduire un nombre plutôt grand de neurones puis à optimiser le seul paramètre de régularisation (decay) par validation croisée.

3.4.5. Remarques

Les champs d'application des PMC sont très nombreux : discrimination, prévision d'une série temporelle, reconnaissance de forme. . . Ils sont en général bien explicités dans les documentations des logiciels spécialisés. (11)

Les critiques principales énoncées à l'encontre du PMC concernent les difficultés liés à l'apprentissage (temps de calcul, taille de l'échantillon, localité de l'optimum obtenu) ainsi que son statut de boîte noir. En effet, contrairement à un modèle de discrimination ou un arbre, il est a priori impossible de connaître l'influence effective d'une entrée (une variable) sur le système dès qu'une couche cachée intervient. Néanmoins, des techniques de recherche de sensibilité du système à chacune des entrées permettent de préciser les idées et, éventuellement de simplifier le système en supprimant certaines des entrées.

En revanche, ils possèdent d'indéniables qualités lorsque l'absence de linéarité et/ou le nombre de variables explicatives (images) rendent les modèles statistiques traditionnelles inutilisables. Leur flexibilité par l'introduction de couches spécifiques en apprentissage profond, alliée à une procédure d'apprentissage intégrant la pondération (le choix) des variables comme de leurs interactions peuvent les rendre très efficaces.

3.5. Introduction à l'apprentissage profond :

Les techniques associées sont simplement introduites dans ce document, elles sont développées dans celui associé au cours de Statistique en grande dimension.

3.5.1. Préambule :

Pendant les années 90s et le début des années 2000, le développement de l'apprentissage machine s'est focalisé sur les algorithmes de machines à vecteurs supports et ceux d'agrégation de modèles. Pendant une relative mise en veilleuse du développement de la recherche sur les réseaux de neurones, leur utilisation est restée présente de même qu'une veille attendant le développement de la puissance de calcul et celle des grandes bases de données, notamment d'images.

Le renouveau de la recherche dans ce domaine est dû **à Geoffrey Hinton**, **Yoshua Bengio** et **Yan le Cun** qui a tenu à jour un célèbre site dédié à la reconnaissance des caractères manuscrits de la base **MNIST**. La liste des publications listées sur ce site témoigne de la lente progression de la qualité de reconnaissance, de **12%** avec un simple perceptron à 1 couche jusqu'à moins de 0,3% en 2012 par l'introduction et l'amélioration incrémentale d'une couche de neurones spécifique appelée **Convulational Neural Network (Conv-Net).** L'étude de ces données qui ont servi de benchmark pour la comparaison de très nombreuses méthodes sert maintenant de données jouet pour beaucoup de tutoriels des environnements dédiés (tensorFlow, Keras, pyTorch, caffe...)

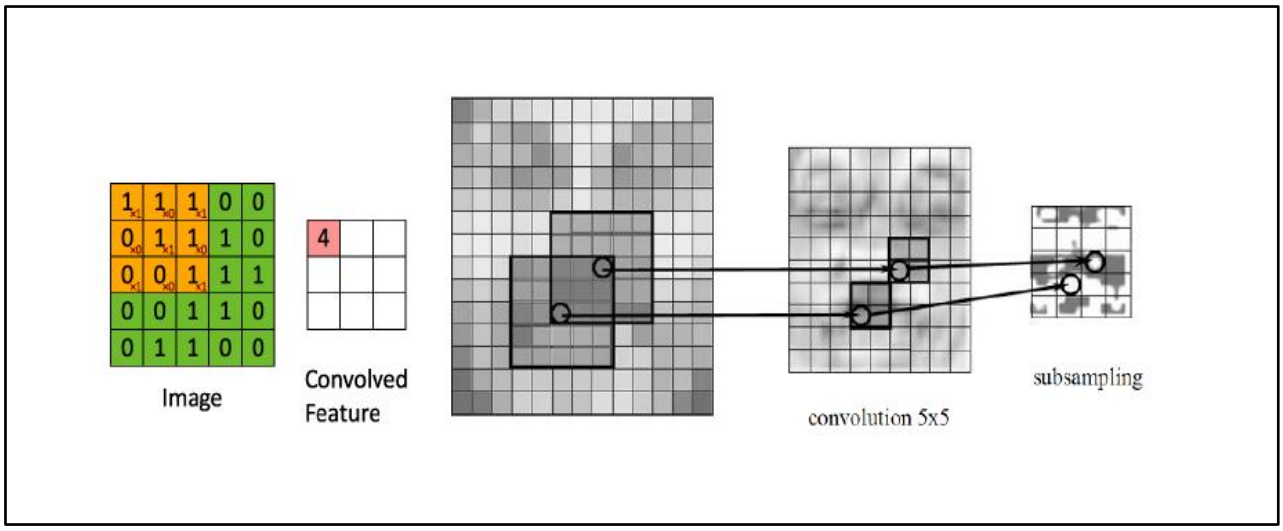
Schématiquement, trois grandes familles de réseaux d'apprentissage profond sont développées avec des ambitions industrielles en profitant du développement des cartes graphiques (GPU) pour paralléliser massivement les calculs au moment de l'apprentissage.

convolutional neural networks (ConvNet) pour l'analyse d'images. long-short term memory (LSTM) lorsqu'une dimension temporelle ou plus généralement des propriétés d'autocorrélation sont à prendre en compte pour le traitement du signal ou encore l'analyse du langage naturel.
Auto Encoder/Décoder ou réseau diabolo en apprentissage non supervisé pour, par exemple, le débruitage d'images ou signaux, la détection d'anomalies.

Seul le premier point est développé pour illustrer les principaux enjeux.

3.5.2. Reconnaissance d'images :

Cette couche de neurones (ConvNet) illustrée par la figure (3.3) ou plutôt un empilement de ces couches induit des propriétés locales d'invariance par translation. Ces propriétés sont indispensables à l'objectif de reconnaissance de caractères et plus généralement d'images qui peuvent être vues sous des angles différents. C'est dans ce domaine que les résultats les plus spectaculaires ont été obtenus tandis que l'appellation **deep learning** était avancée afin d'accompagner le succès grandissant et le battage médiatique associé.



Figure(3.3)- Principe élémentaire d'une couche de convolution et application à une image.

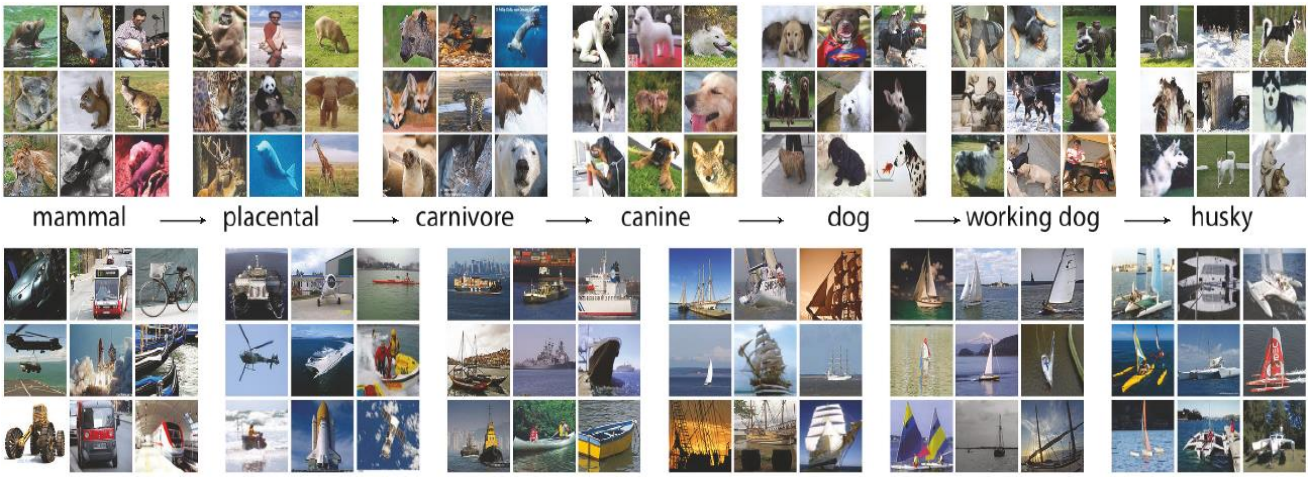


FIGURE (3.4)– Échantillon de la base ImageNet.

La communauté de reconnaissance d'images figure (3.4) se confronte chaque année depuis 2010 sur un jeu de données issues d'une base d'images labellisées : 15 millions d'images, 22000 catégories hiérarchisées. De cette base sont extraites 1,2 millions d'images pour l'apprentissage avec 1000 catégories.
Les participants au concours doivent prévoir la catégorie de 15000 images de l'échantillon test. Ce projet à l'initiative de l'Université Stanford est largement soutenu par Google.

Comme pour les données de reconnaissance de caractères, une progression largement empirique à conduit à l'introduction et au succès d'un réseau empilant des couches de neurones aux propriétés particulières.

Cette progression est retracée dans la figure (3.5). C'est en 2012 qu'une équipe utilise pour la première fois un réseau de neurones profond contrairement à des traitements spécifiques de l'analyse d'images utilisées jusque-là. L'amélioration était telle que toutes les équipes ont ensuite adopté cette technologie pour une succession d'améliorations empiriques. En 2016 une équipe propose un réseau à 152 couches et atteint un taux d'erreur de 3%, mieux que les 5% d'un expert humain.

Ce concours est depuis lors abandonné au profit de problèmes plus complexes de reconnaissance de scènes associant plusieurs objets ou thèmes.

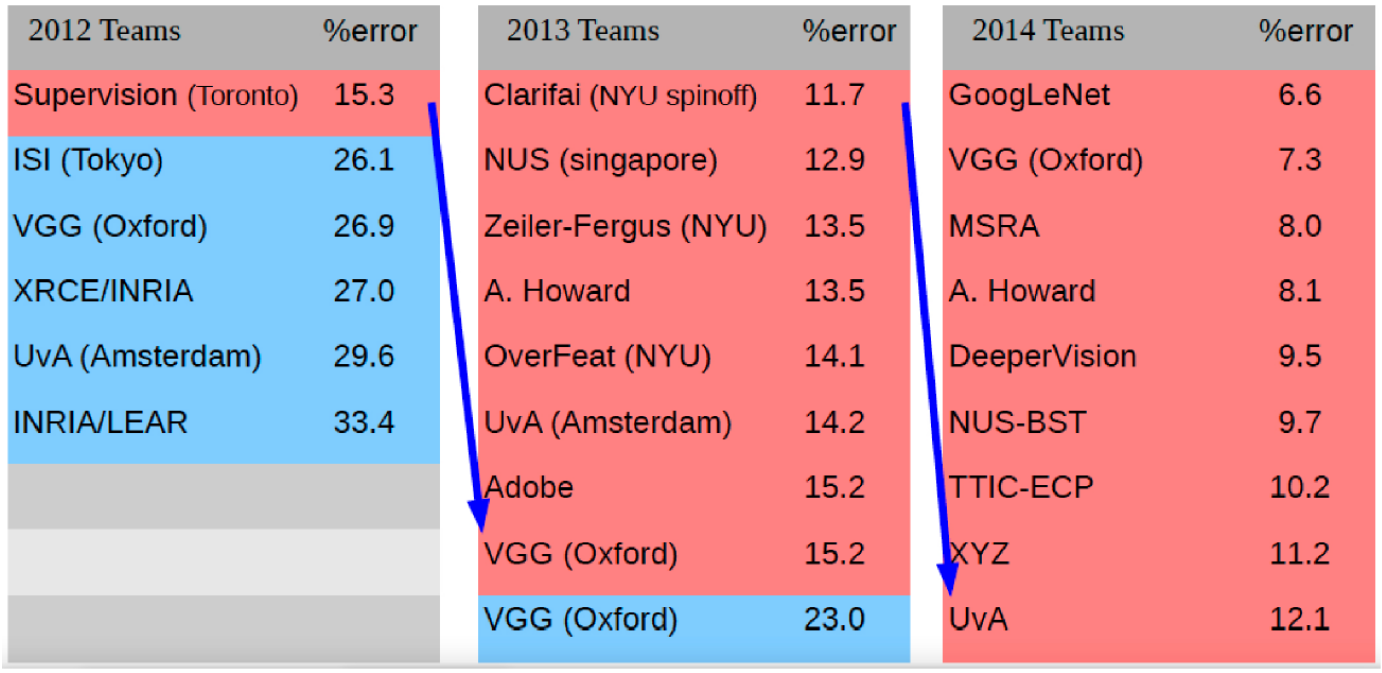


FIGURE (3.5) – Classements successifs (Le Cun 2016) des équipes participant au concours ImageNet. En rouge, celles utilisant des neurones profonds.

3.5.3. Couches pour l'apprentissage profond :

Construire un réseau d'apprentissage profond consiste à empiler des couches de neurones aux propriétés spécifiques rapidement résumées ci-dessous. Le choix du type, de l'ordre, de la complexité de chacune de ces couches ainsi que du nombre est complètement empirique et l'aboutissement de très nombreuses expérimentations nécessitant des moyens de calculs et bases de données considérables

Fully connected : Couche classique de perceptron et dernière couche d'un réseau profond qui opère la discrimination finale entre par exemple des images à reconnaître. Les couches précédentes construisant, extrayant, des caractéristiques (features) de celles-ci.

Convolution : opère une convolution sur le signal d'entrée en associant une réduction de dimension (figure 6).

Pooling : réduction de dimension en remplaçant un sous-ensemble des entrées (sous-image) par une valeur, généralement le max. normalisation identique au précédent avec une opération de centrage et / ou de normalisation des valeurs.

Drop out les paramètres estimés sont les possibilités de supprimer des neurones d'une couche afin de réduire la dimension.

3.5.4. Utilisation rudimentaire :

Sans bases de données très volumineuses et moyens de calcul substantiels il est illusoire de vouloir apprendre un réseau profond impliquant l'estimation de millions de paramètres.
Une mise en œuvre simple sur des données spécifiques consiste à :

- Identifier un réseau ou modèle existant appris sur des données similaires.

Pour les images, considérer par exemple les versions des réseaux inception de tensorFlow ou AlexNet de Caffe.

- Supprimer la dernière couche du modèle dédiée à la classification,
- Apprendre les poids de cette seule dernière couche sur les données spécifiques.

3.6. Conclusion

Dans ce chapitre on a parlé sur les réseaux de neurones à partir de neurone biologique jusqu'aux réseaux de neurones artificiels, avec ce réseau on va construire un environnement artificiel intelligent pour effecteur l'opération de la reconnaissance des images.

Les CNN sont souvent utilisés dans les systèmes de reconnaissance d'image . En 2018, un taux d'erreur de 0,18% sur la base de données MNIST a été signalé. [10]

Chapitre 04 :

Classification des Modulations Radar

4. Chapitre 4 : Classification des Modulations Radar

4.1. Introduction :

Comme annoncé, ce mémoire se divise en deux parties, l'une théorique et l'autre pratique. Cette dernière, présentée dans les pages qui suivent, illustre et propose différents simulations réaliser, qui commence par la génération des signaux à différentes modulation qui se transforme au domaine temps-fréquence et par la suite en images JPG et cela pour déterminer le type de la modulation et l'extraction des paramètres de cette modulation à travers le réseau de neurone convolutionnel et aussi dans le but de savoir quelle type de transformée temps fréquence est plus adapté à ce type de signaux et pour découvrir la performance de réseau de neurone convolutionnel.

Enfin, avant d'aborder les contenus de différentes simulations, nous avons tenu, dans ce qui suit, un aperçu global de cette partie pratique sous la forme d'un organigramme permettant ainsi, à la fois une lecture aisée et une synthèse de ce présent module.

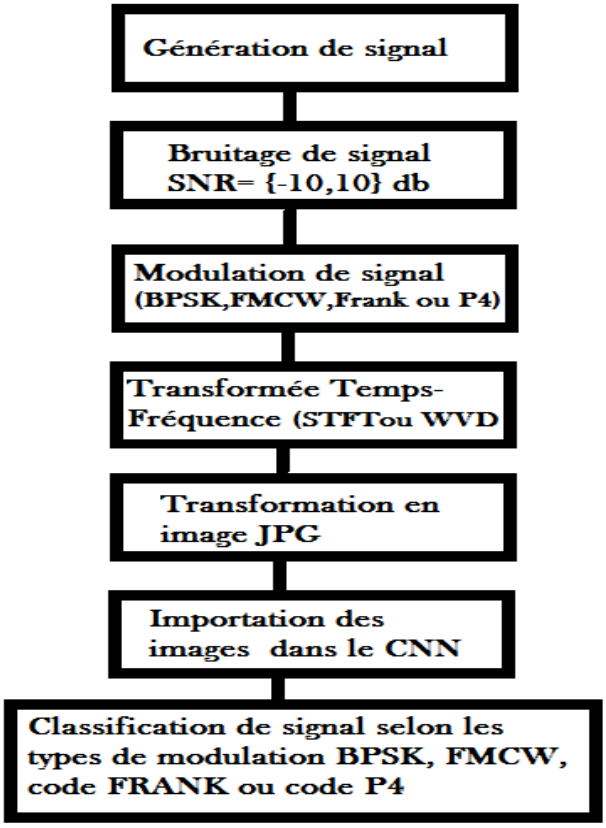


Figure 4.1 Organigramme illustrant la partie pratique

4.2. Présentation de logiciel Matlab :

Matlab (MATrix LABoratory) est un logiciel commercial pour le calcul scientifique, orienté vers les vecteurs et les listes de données, il permet de réaliser des simulations numériques basées sur des algorithmes d'analyse numérique.

Matlab est un langage interprété, chaque ligne d'un programme Matlab est lue, interprétée et exécutée, il peut donc être utilisé pour la résolution approchée d´équations différentielles, d´équations aux d´dérivées partielles, il permet aussi de manipuler des matrices, d'afficher des courbes et des données, de traiter les signaux, de mettre en œuvre des algorithmes, de créer des interfaces utilisateurs…etc.

Dans notre cas le logiciel MATLAB est utilisé pour le traitement des signaux et des images.

Le travail dans cette partie expérimentale se partage essentiellement en 3 taches :

- La première tâche est la génération des signaux avec 4 types de modulation (BPSK-CODE DE FRANK-CODE P4-FMCW)) avec la variation de plusieurs paramètres de signal (SNR, f_C, Δf, ….).

La figure (4.2) montre un exemple d'un signal généré pour une modulation FMCW avec de bruit gaussien (un rapport signal/bruit égal à -2 décibel).

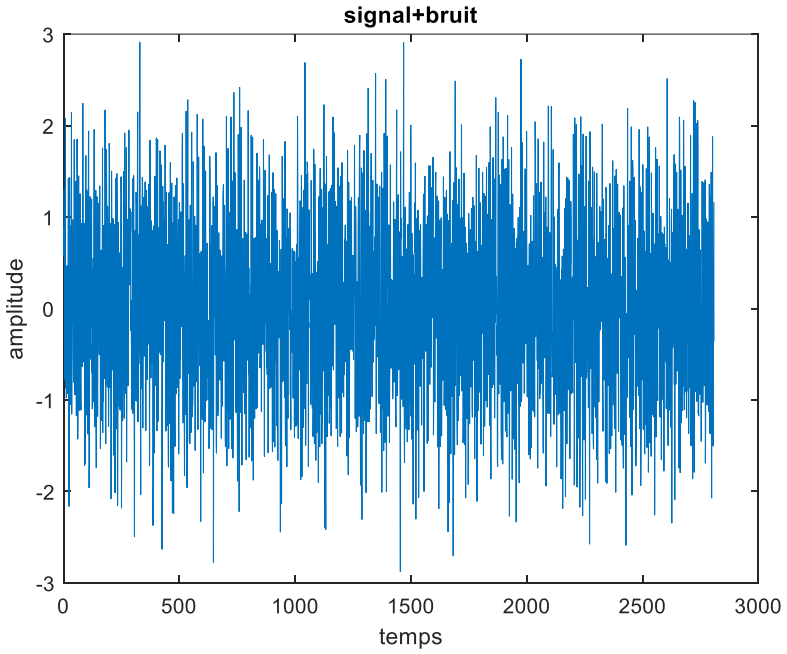


Figure (4.2) : un signal modulé en FMCW + le bruit gaussien SNR=-2 dB

Les signaux générés sont transformés en domaine temps-fréquence à l'aide de deux transformées déférentes, la distribution de Wigner Ville et la transformée de Fourier à court terme pour qu'elles soient comparées par la suite.

Les figures (4.3) (4.4) présentent les deux transformées temps-fréquence pour le signal précédent.

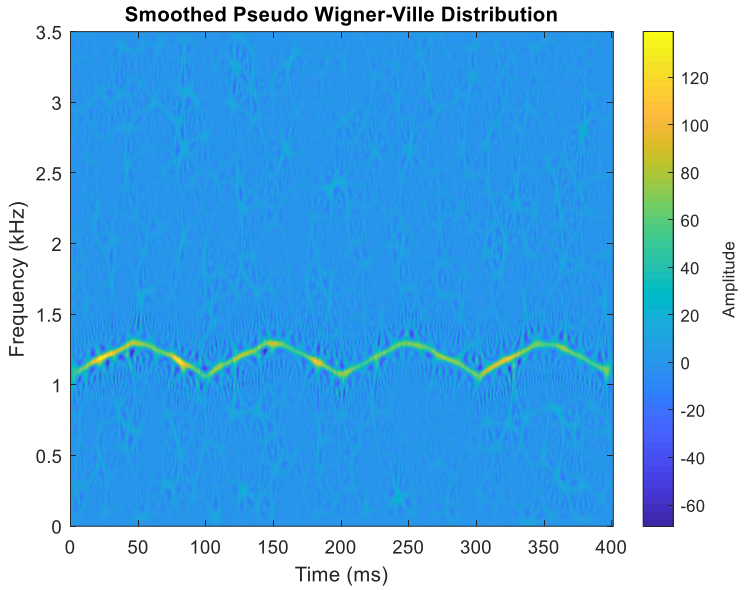


Figure (4.3) : la transformée de signal modulé en FMCW+ le bruit gaussien SNR=-2 dB avec WVD.

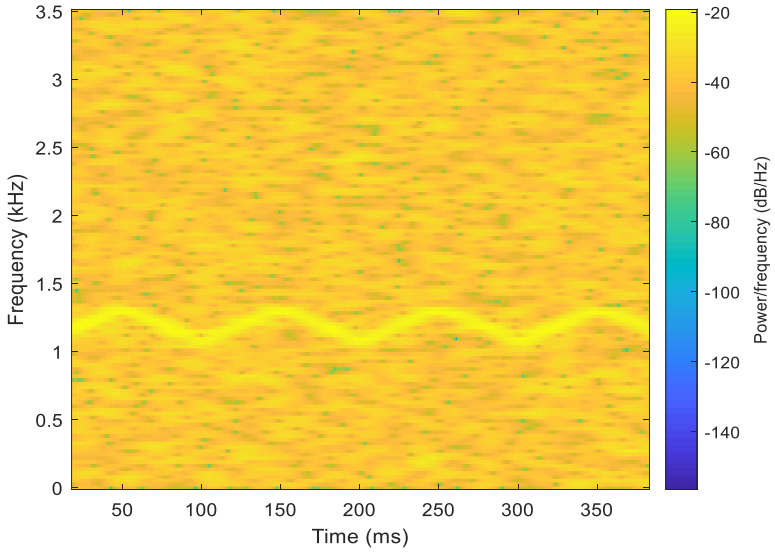


Figure (4.4) : la transformée de signal modulé en FMCW+ le bruit gaussien SNR=-2 dB avec STFT

Les transformées temps-fréquence générées doit être enregistré en format image(JPG) avec le code MATLAB qui transforme une matrice en une image.

Nom de variable=['image',num2str(snr), num2str(f_c), num2str(Deltaf),'.jpg'];
saveas(gca,Nom de variable);

- La deuxième tâche est l'importation des images depuis leur répertoire dans le PC
  En général les images ont des paramètres (longueur, largeur, profondeur) alors il faudra les redimensionner et de diminuer leurs tailles avant de les télécharger sur MATLAB car le réseau de neurone convolutionnel exige des tailles spécifique pour son traitement d'images.

  Pour le redimensionnement des images on utilise : un toolbox qui se trouve au niveau des applications Matlab '' **APPS**'' s'appelle **''Image Batch Processor''**

  La fonction MATLAB de redimensionnement des images est la suivante :

```
results= imresize(im,[227,227]);
```

  Après on passe à l'exportation des images, sur « output » en sélectionnant ''JPG'' le format d'image.

  Apres le redimensionnement et le sauvegarde des images dans un dossier qui contient des sous dossier pour chaque type de modulations.
  En total il existe 960 images pour les deux types de transformées utilisés (480 pour WVD et 480 pour le STFT)

- La troisième tâche est l'implémentation de réseau de neurones convolutionnel

  Pour notre projet on a préféré d'utiliser un toolbox qui existe déjà sur MATLAB s'appelle « **squeezenet** »

  La figure (4.5) montre l'architecture de réseaux et le nombre des couches et les propriétés de chaque couche

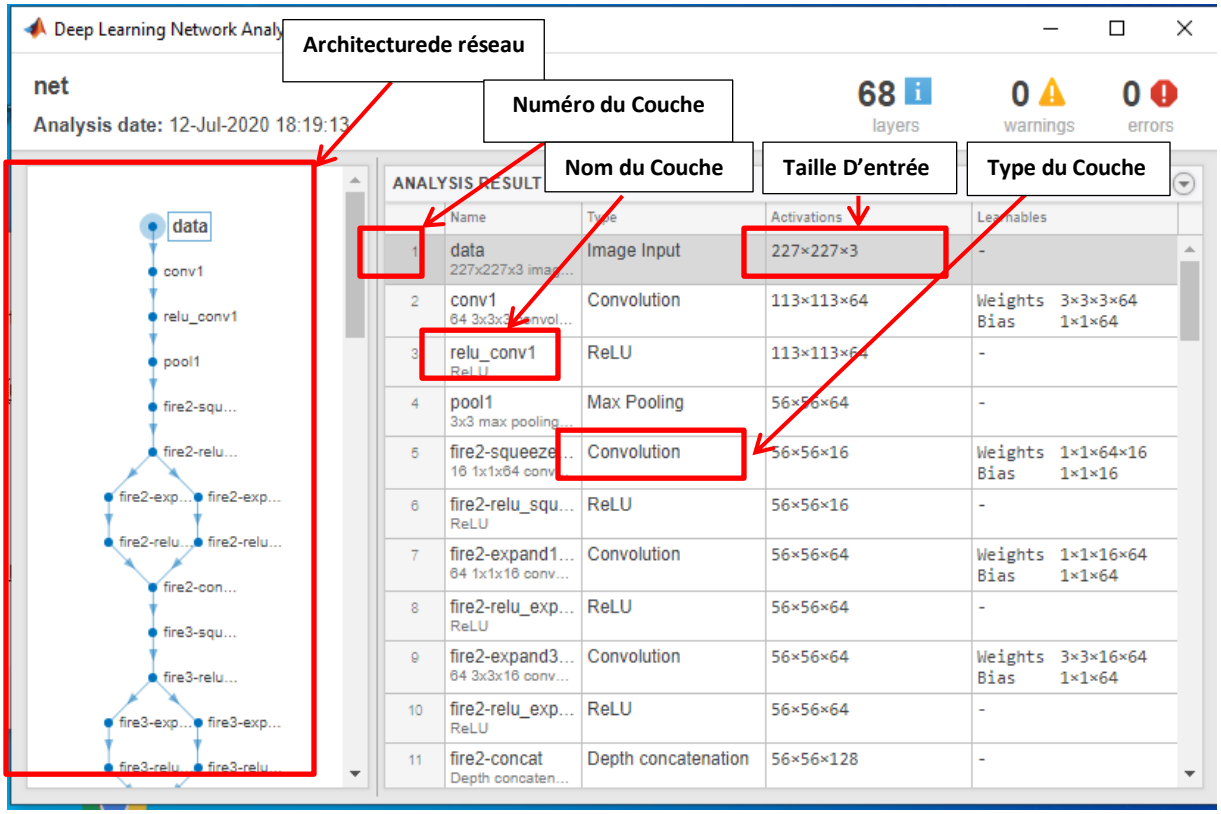


Figure (4.5) : l'architecture de réseaux squeezenet.

Ses réseaux de classification d'images préformés ont été conçu pour plus d'un million d'images et peuvent classer des images en 1000 catégories d'objets, Les réseaux ont appris de riches représentations de fonctionnalités pour un large éventail d'images. Le réseau prend une image en entrée, puis sort une étiquette pour l'objet dans l'image ainsi que les probabilités pour chacune des catégories d'objets.

On peut prendre un réseau préformé et l'utiliser comme point de départ pour apprendre une nouvelle tâche en utilisant la fonction : net = squeezenet.

Le réglage d'un réseau avec l'apprentissage par transfert est généralement beaucoup plus rapide et plus facile que la formation d'un réseau à partir de zéro avec des poids initialisés de manière aléatoire, vous pouvez rapidement transférer des fonctionnalités apprises vers une nouvelle tâche à l'aide d'un plus petit nombre d'images d'entraînement à importer

Pour décompresser les images on utilise la fonction `unzip ( 'MerchData.zip' );`
Et pour les charger dans le réseau on utilise la fonction

```
imds = imageDatastore ( 'MerchData' ,'InclureSous-dossiers' , true, 'LabelSource' ,
'noms de dossiers' );
```

Une parties des images importées sont utilisées pour l'apprentissage de réseau et le reste pour le test.

La fonction `analyserNetwork (net)` permet une visualisation interactive de l'architecture du réseau et des informations détaillées sur les couches du réseau.

Le premier élément dans un réseau est la couche d'entrée d'image, pour un réseau **squeezenet**, cette couche nécessite des images d'entrée de taille 227 x 227 x 3, où 3 est le nombre de canaux de couleur

```
inputSize = net.Layers (1) .InputSize
inputSize = 1 × 3

   227 227 3
```

D'autres réseaux peuvent nécessiter des images d'entrée de tailles différentes, par exemple pour le réseau Xception nécessite des images de taille 299 x 299 x 3.

Dans la plupart des réseaux, la dernière couche avec des poids apprenants est une couche entièrement connectée. On doit remplacez cette couche entièrement connectée par une nouvelle couche entièrement connectée avec un nombre de sorties égal au nombre de classes dans le nouvel ensemble de données (4 dans notre cas).
Dans notre cas on remplace la couche Convolutionnelle et la couche de classification finale
On doit trouver leurs noms d'abord en utilisant :

```
[learnableLayer, classLayer] = findLayersToReplace (lgraph);
ans =
Tableau 1 × 2 couches avec couches:

1 'conv10' Convolution 1000 1 × 1 × 512 convolutions avec foulée [1 1] et rembourrage
[0 0 0 0]
2 'ClassificationLayer_predictions' Classification Crossentropyex en sortie avec
'tench' et 999 autres classes
```

Dans certains réseaux, tels que SqueezeNet, la dernière couche apprenable est une couche convolutive 1 par 1 à la place. Dans ce cas, remplacez la couche convolutive par une nouvelle couche convolutive avec le nombre de filtres égal au nombre de classes en utilisant la fonction :

```
numClasses = numel (catégories (imdsTrain.Labels))
numClasses = 5
```

Pour apprendre plus rapidement dans les nouvelles couches que dans les couches transférées, augmentez les valeurs **WeightLearnRateFactor** et **BiasLearnRateFactor** de la couche convolutive.

```
newConvLayer = convolution2dLayer ([1, 1], numClasses, 'WeightLearnRateFactor' , 10,
'BiasLearnRateFactor' , 10, "Name" , 'new_conv' );
lgraph = replaceLayer (lgraph, 'conv10' , newConvLayer);
```

Après le réglage d'architecture on passe à l'apprentissage, validation et affichage des résultats.

La figure (4.6) montre un exemple d'évolution d'apprentissage pour le **squeezenet** avec 20 % des images, Sur la figure de l'évolution on peut visualiser la courbe bleu qui montre l'apprentissage avec succès et la rouge c'est les images perdus (il n'arrive pas à les classer), le pourcentage de validation des résultats qui est égal à 86.31% de cet exemple, le nombre d'itérations, la durée…etc.

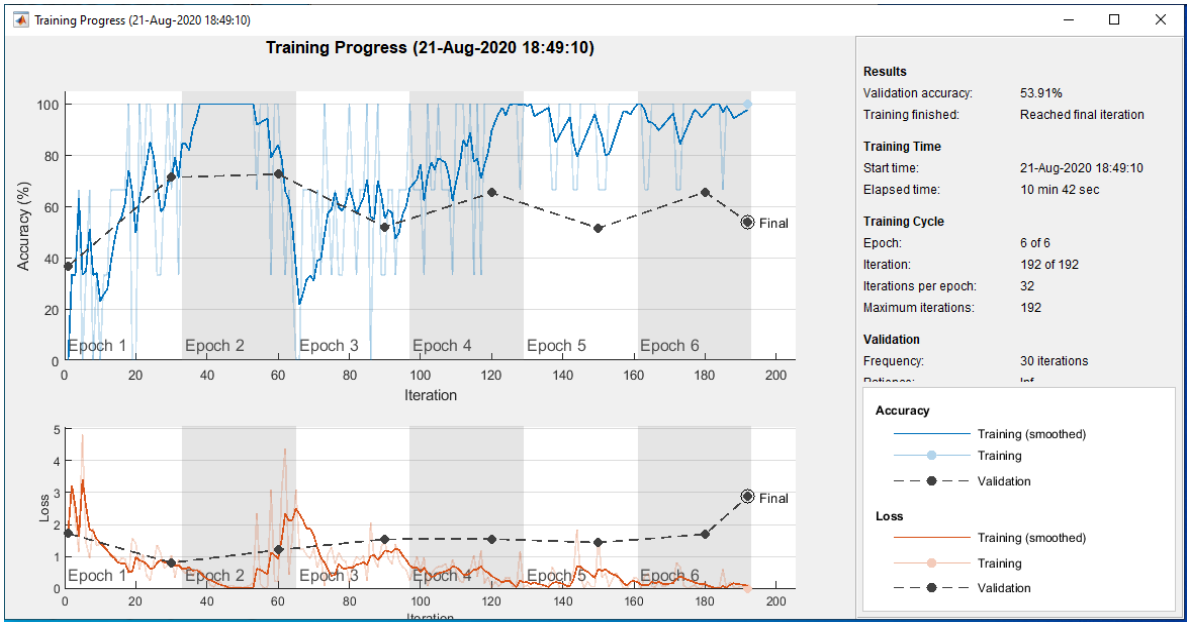


Figure (4.6) : courbes d'évolution d'apprentissage de réseau.

La figure suivante qui s'appelle matrice de confusion présente les résultats de reconnaissance des classifications. Tel que les cellules sur la diagonal indiquent le taux de la bonne classification et les autres cellules donnent le taux de mauvaise classification par classe.

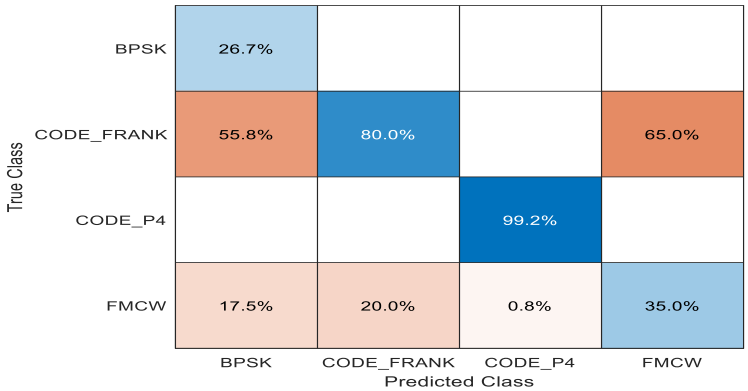


Figure (4.7) : matrice de confusion pour la prédiction des résultats de test

Pour résumer les résultats de différentes valeurs utilisées dans la simulation et pour comparer les deux transformées temps-fréquence, les tableaux (4.1) et (4.2) récapitulent les résultats obtenus.

- Pour WVD :

	Nombre d'image	Pourcentage de la reconnaissance	Durée de traitement
Distribution de Wigner-Ville	0.2 % (96 image)	73.96 %	10 min 58 sec
	0.3 % (144 image)	81.56 %	14 min 5 sec
	0.5 % (240 image)	97.92 %	18 min 56 sec

Tableau (4.1) : résultats d'apprentissage avec le réseau squeezenet pour le WVD

- Pour STFT :

	Nombre d'image	Pourcentage de la reconnaissance	Durée de traitement
Transformée de Fourier à court Terme	0.2 % (96 image)	76.82 %	10 min 57 sec
	0.3 % (144 image)	86.31 %	14 min 3 sec
	0.5 % (240 image)	100 %	19 min 17 sec

Tableau (4.2) : résultats d'apprentissage avec le réseau squeezenet pour le STFT.

Les Tableaux (4.1) et (4.2) donnent les résultats des classifications des modulations radar avec le réseau de neurone convolutionnel en utilisant comme donnée la transformé WVD et STFT respectivement. On remarque que le temps d'exécution pour effectuer l'opération de classification augmente en fonction de nombres des images utilisé dans le processus d'apprentissage. Dans le cas de 96 images en apprentissage le temps d'exécution est autour de 10 minutes alors que pour 240 images le temps est autour de 19 minutes.
On conclue aussi que la STFT donne de meilleur résultat de classification que le WVD.
on donne aussi la matrice de confusion de chaque transformée temps fréquence.

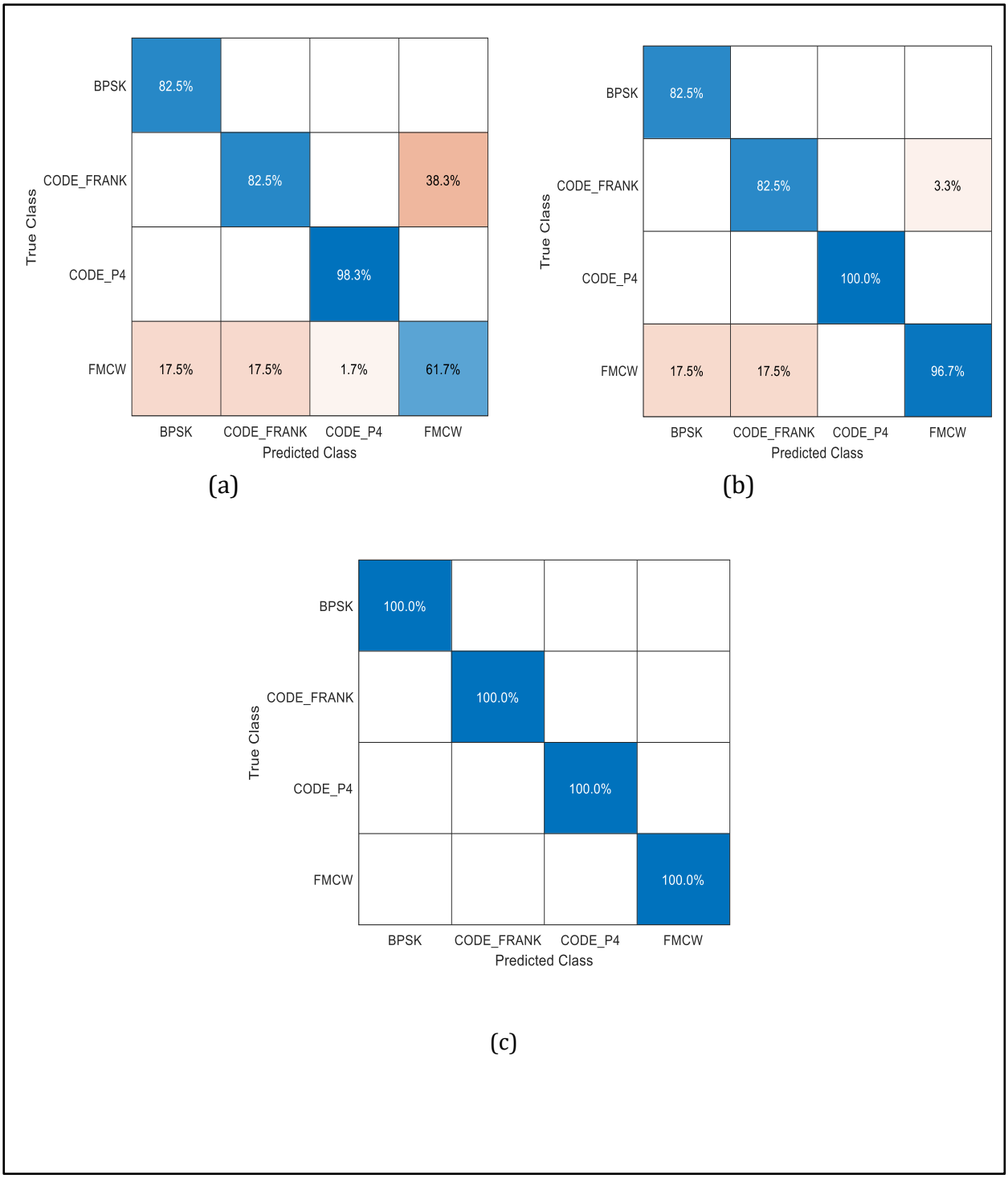
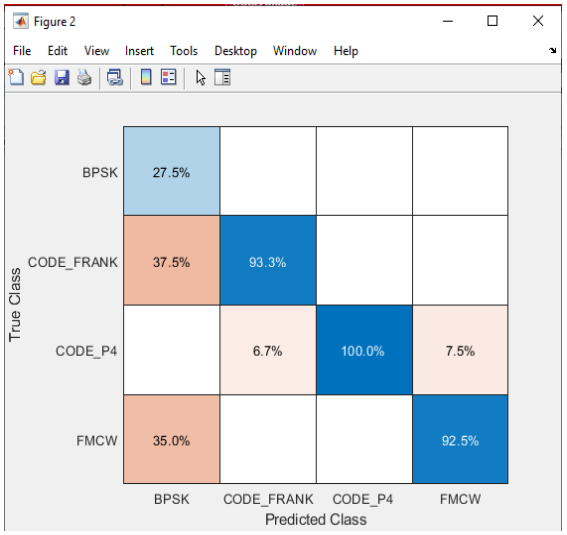
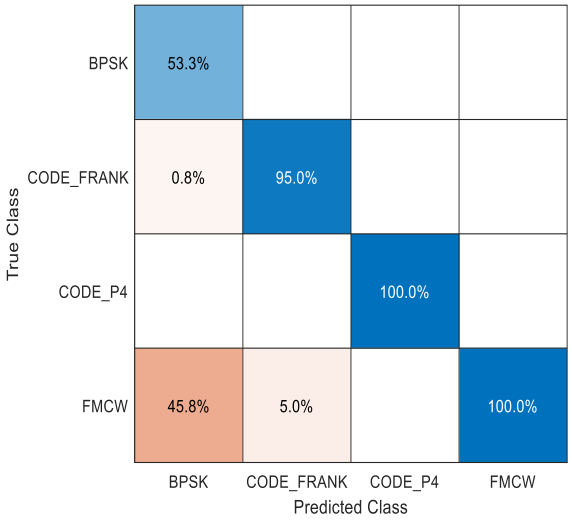


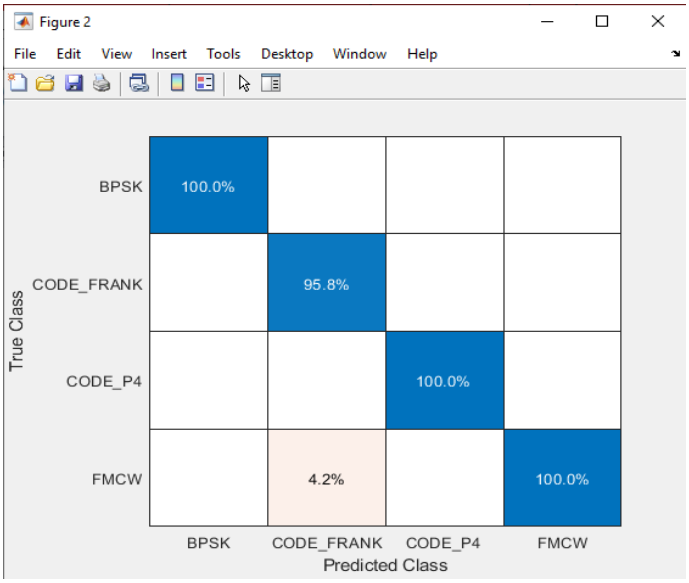
Figure (4.8) : (a)-matrice de confusion pour 0.2 % des images, (b)-matrice de confusion pour 0.3 % des images, (c) matrice de confusion pour 0.5 % des images pour STFT



(a)



(b)



(c)

Figure (4.9) : (a)-matrice de confusion pour 0.2 % des images, (b)-matrice de confusion pour 0.3 % des images, (c) matrice de confusion pour 0.5 % des images pour WVD.

La figure (4.8) donne les matrices de confusion sur les résultats des classifications des images générer avec la STFT. A partir de la matrice (a) avec un apprentissage de 20% des images, le taux de reconnaissance des modulations BSPK et Frank est le même (82%), la modulation la plus reconnu est P4 (98.3%). Après augmentation de nombre des images à 30% les résultats indiquer dans la matrice (b) indique qu'il n'y a pas de changement de taux de reconnaissance des modulations BPSK et Frank, par contre toutes les modulations P4 sont reconnues à 100% et FMCW a connu une grande amélioration, de 61.7% à 96.7%.
Après l'utilisation de 50% des images dans l'apprentissage des classificateurs, la matrice de confusions (c) de la figure (4.16) indique que le taux de reconnaissance des modulations est de 100% pour les quatre types des modulations.

La figure (4.9) donne les matrices de confusions des classifications des images générer avec la WVD. La matrice de confusion (a), pour une classification pour un apprentissage avec 20% des images, indique un taux très de reconnaissance très bas pour la modulation BPSK avec 27.5%, par contre les autres types des modulations présentent des taux de reconnaissance très élevé, 93.3% pour Frank, 92.5 % pour FMCW et 100% pour P4. Après l'utilisation de 50% des images dans l'apprentissage on est arrivé à une reconnaissance de 100% pour toutes les modulations.

A partir des analyses précédentes, on déduit que globalement, si on prend en compte les quatre modulations la STFT donne de meilleurs résultats de classification que la WVD. Mais si on prend analyse les résultats selon le type de modulations, on remarque que la WVD donne de meilleurs résultats que la STFT pour les modulations de Frank, P4 et FMCW. Sauf dans le cas de la modulation BPSK où les images générer avec la WVD présente de mauvais résultats ce qui a permis de réduit le taux de reconnaissance globale pour la WVD.

4.3. Conclusion :

D'après le tableau (4.1), le tableau (4.2) et les déférentes matrice de confusion on peut remarquer que la transformée de Fourier à court terme est plus performante que la distribution de Wigner-Ville dans le traitement des signaux modulés pour le radar et cela parce que la WVD présente plus de termes croisés, par contre la STFT malgré son inconvénient de fenêtrage qui élimine les amplitudes supérieures à l'amplitude de la fenêtre choisie, elle ne présente pas des termes croisées fortes.
Nous déduisons que la caractéristique essentielle dans les transformées temps-fréquence est le nombre de termes croisés quelle présente et cela parce que l'absence de ces termes dans le plan temps-fréquence permet de déterminer plus facilement le type de modulation et facilite également l'extraction des paramètres de modulation et ce qui concerne le réseau de neurone convolutionnel on peut conclure que ce réseau est très performant et puissant, car c'est seulement avec 50 % des images on est arrivé à 100 % des prédictions vraies.

5. Conclusion générale :

Ce mémoire de fin d'étude a eu pour objectif de répondre à la question de la recherche : « peut-on concevoir un système qui permet d'identifier et d'extraire les paramètres de la modulation intra-impulsion pour le signal radar d'une façon rapide et automatique ? ». Pour cela nous avons proposé une approche de reconnaissance de cette modulation basée sur un réseau neurone convolutionnel. L'approche peut identifier les quatre déférents types de modulation telle que la modulation en phase (BPSK, FRANK, P4) et en fréquence (FMCW).

Apres les deux premiers chapitres qui présentent l'étude théorique de traitement de signal radar suivi par un troisième consacré au réseau de neurone convolutionnel, on terminera par un quatrième qui montre Les résultats de la simulation réalisé par le logiciel MATLAB.

L'approche atteint 100% de réussite pour les quatre types de signaux radar avec seulement 50% d'images utilisées pour l'apprentissage sachant que le rapport signal sur bruit SNR varient entre -10 dB et 10 dB

Cela montre que l'approche est utile et fiable et la proposition de cette dernière a une bonne adaptabilité et elle peut identifier efficacement les signaux radar avec des paramètres de plage de variation importante tel que le SNR, et les paramètres des modulations.

La simulation montre aussi que le type de transformée temps-fréquence jouent un rôle très important lors d'identification de type de modulation, pour cela le choix de cette transformée doit s'effectuer soigneusement, et il peut être aussi comme futur travail de recherche.

Finalement, nous nous sommes rendu compte que le but d'un travail de recherche n'est pas forcément de donner des réponses complétement nouvelles mais d'essayer de contribuer aux résultats d'autre chercheurs.

Comme une future perspective et amélioration pour notre projet de fin d'étude, est l'exploration d'autres transformés temps-fréquence, et autres types d'algorithmes de classification que ceux utilisé dans notre projet, avec l'utilisation de plus de type des modulations.

Bibliographie

[1] Phillip E. Pace, *Detecting and Classifying Low Probability of Intercept Radar*, Artech house, 2nd Edition, 2009.

[2] Taboada, Fernando L, *Detection and classification of low probability of intercept radar signals using parallel filter arrays and higher order statistics*. ,Monterey, California Naval Postgraduate School , September 2002

[3] Dave Fahlman, B.Eng. *Automatic Radar Modulation Classification* ,Ottawa-Carleton Institute for Electrical and Computer Engineering Department of Electronics Carleton University, Ottawa, Ontario , june 2018

[4] Bensalem Merouane, Dekhli Nassim, étude des propriétés d'un signal radar LFM Université SAAD DAHLEB Blida 1, institut d'aéronautique et des études spatiales,2015

[5] Yong June Shin, B.S.; M.S. Theory and Application of Time-Frequency Analysis to Transient Phenomena in Electric Power and Other Physical Systems, Faculty of the Graduate School of The University of Texas at Austin, 2004

[6] Young S. Shin, Jae-Jin Jeon, Pseudo Wigner-Ville Time-Frequency Distribution and Its Application to Machinery Condition Monitoring, Department of Mechanical, Engineering Naval Postgraduate School Monterey, CA 93943,1993

[7] Zhang, Wei (1988). "Shift-invariant pattern recognition neural network and its optical architecture". Proceedings of Annual Conference of the Japan Society of Applied Physics

[8] Convolutional Neural Networks (LeNet) – DeepLearning 0.1 documentation". DeepLearning 0.1. LISA Lab. Retrieved 31 August 2013.

[9] Krizhevsky, *Alex.* "ImageNet Classification with Deep Convolutional Neural Networks" (PDF). Retrieved 17 November 2013.

[10] A Survey of FPGA-based Accelerators for Convolutional Neural Networks, NCAA 2018

[11] Karman Kowsari, Mojtaba Heidarysafa, Donald E.Brown, kiana jafari Meimandi et Laura E.Barnes, « RMDL : Random Multimodel Deep Learning for Classification, sur arXiv.org e-Print archive, consulté le 10 mai 2018.

Annexe : Code MATLAB

Code MATLAB pour la génération des signaux, la transformée temps-fréquence et la conversion en images JPG avec le code Frank :

```
clear all;
clc;
A=1; % l'amplitude de l'onde continue
f =1.2e3; % la fréquence porteuse
fs =7e3; % fréquence d'échantillonnage
SNR_dB =10; %rapport signal sur bruit
scale=30; % Mise à l'échelle pour la trace des graphes
j=sqrt(-1);
m=8; % nombre de code de phases
cpp = 4; %nombre de cycles par phase
SAR=ceil(fs/f); % rapport d'échantillonnage
tb=1/(fs); %la période d'échantillonage
N=m;
for i=1:m
 for j=1:m
 phi(i,j)=2*pi/N*(i-1)*(j-1);
 end
 end
index=0;
for p=1:5
 for i=1:m
 for j=1:m
 for n=1:SAR*cpp
 I(index+1)=A*cos(2*pi*f*(n-1)*tb+phi(i,j));
IWO(index+1)=A*cos(2*pi*f*(n-1)*tb);
 Q(index+1)=A*sin(2*pi*f*(n-1)*tb+phi(i,j));
QWO(index+1)=A*sin(2*pi*f*(n-1)*tb);
 time(index+1)=index*tb; le vecteur temps.
 index = index +1;
 end
 end
 end
end
 [a,b]=size(I);
SNR=10^(SNR_dB/10);
power=10*log10(A^2/(2*SNR));
 noise=wgn(a,b,power);
IN=I+noise;
IPWON=I;
QN=Q+noise;
QPWON=Q;
 ff=floor(f/1e3);
ffs=floor(fs/1e3);
nn=0;
 for ii=1:N
 for jj=1:N
```

```
 nn=nn+1;
 phi2(nn)=phi(ii,jj);
 end
 end
 xx=0:length(phi2)-1;
INP=IN';
QNP=QN' ;
IPWONT=IPWON' ;
QPWONT=QPWON' ;
I= INP(:,1);
Q=QNP(:,1);
II= IPWONT(:,1);
QQ=QPWONT(:,1);

d=wvd(IN,fs,'smoothedPseudo',[]);
s =imagesc(d);
xlabel('temps');ylabel('fréquence'); axis xy;  title('Frank');
 variable=['Frank_', num2str(ff), '_', num2str(ffs), '_', num2str(m),
'_',num2str(cpp),'_',num2str(SNR_dB),'.jpg'];
  saveas(gca,variable);
```

Code MATLAB pour la génération des signaux, la transformée temps-fréquence et la conversion en images JPG avec le code BPSK :

```
clc;
clear all;
A = 1; %Amplitud of the carrier signal
f = 1.2e3; %fréquence de signal porteuse
fs = 7e3; % fréquence d'échantillonage
SNRdb = 1; % SNR désiré  in dB
barker = 7; %nombre de bits de code Barker
np = 175; % nombre des périodes de  code
NPBB = 5; % Nombre de cycles per bit de Barker
NPV = 55; % nombre de périodes pour afficher le graphe
   for SNRdb =-10:10
       for NPBB = 1:7
        for   barker = [7 11]

ti = 1/(f*100);
t = 0:ti:np/f;
xt = A*sin(2*pi*f.*t);
SAR=floor(fs/f);%
n = 0:1:SAR*np;
xnT = A*cos(2*pi.*n*f/fs);
xnT2 = A*sin(2*pi.*n*f/fs);
fm = f/NPBB;
pw = floor(fs/fm);
if barker==13
 brk = [ones(1,pw*5),-(ones(1,pw*2)),ones(1,pw*2),-
ones(1,pw),ones(1,pw),-ones(1,pw),ones(1,pw)];
 elseif barker==11
 brk = [ones(1,pw*3),-(ones(1,pw*3)),ones(1,pw),-ones(1,pw),-
ones(1,pw),ones(1,pw),-ones(1,pw)];
else
 brk = [ones(1,pw*3),-(ones(1,pw*2)),ones(1,pw),-ones(1,pw
end
brkseq = [];
ns = floor(length(n)/(pw*barker));
for step = 1:1:ns; brkseq = [brkseq,brk];
end
I = xnT(1:barker*ns*pw).*brkseq;
Q = xnT2(1:barker*ns*pw).*brkseq;
rl = length(I);
 [a,b]=size(xnT);
SNR=10^(SNRdb/10);
power=10*log10(A^2./(2*SNR));
noise=wgn(a,b,power);
IwN =I+noise(1:length(I));
QwN=Q+noise(1:length(Q));
 xnTwN = xnT + noise;
format short e;
IwN=IwN';
QwN=QwN';
```

```
 I2=I; Q2=Q;
 I=IwN; Q=QwN;
 ff=floor(f/1e3);
 ffs=floor(fs/1e3);
I=I2';
 Q=Q2';
spectrogram(IwN,256,250,256,fs,'yaxis');
 spectrogram(QwN,256,250,256,fs,'yaxis');
xlabel('temps');ylabel('fréquence'); axis xy;  title('barker');
xlabel('temps');ylabel('fréquence'); axis xy;  title('barker');
 variable=['barker_', num2str(ff), '_', num2str(ffs), '_',
num2str(barker), '_',num2str(NPBB),'_',num2str(SNRdb),'.jpg'];
 saveas(gcf,variable);
        end
       end
   end
```

Code MATLAB pour la génération des signaux, la transformée temps-fréquence et la conversion en images JPG avec le code P4:

```
clear all;
clc;
close all;
A=1; % Amplitude de l'onde continue
f =1e3; % fréquence porteuse
fs =7e3; % taux d'échantillonnage
SNR_dB = 0; %rapport signal sur bruit
scale=30; % échelle d'affichage de graphe
j=sqrt(-1);
m=16; % nombre de code de phases
cpp = 1; % nombre de  cycles par phase
SAR=ceil(fs/f); % rapport  d'échantillonnage
tb=1/(fs); % période d'échantillonage
for SNR_dB = -10:10
    for cpp = 1:1:7

for k = 1:m
 phase(k)=((pi/m)*(k-1)^2)-(pi*(k-1)); end
index=0;
for p=1:5
for l=1:m
 for n=1:SAR*cpp
 I(index+1)=A*cos(2*pi*f*(n-1)*tb+phase(l));
IWO(index+1)=A*cos(2*pi*f*(n-1)*tb);
 Q(index+1)=A*sin(2*pi*f*(n-1)*tb+phase(l));
QWO(index+1)=A*sin(2*pi*f*(n-1)*tb);
 time(index+1)=index*tb; index = index +1;
 end
end
end
 [a,b]=size(I);
SNR=10^(SNR_dB/10);
power=10*log10(A^2/(2*SNR));
 noise=wgn(a,b,power);
IN=I+noise;
IPWON=I;
QN=Q+noise;
QPWON=Q;
 ff=floor(f/1e3);
 ffs=floor(fs/1e3);
```

Code MATLAB pour la génération des signaux, la transformée temps-fréquence et la conversion en images JPG avec FMCW:

```
clc;
clear all;
A = 1; % amplitude de signal porteur
f0 =1e3; fréquence de la porteuse
fs = 7e3; %fréquence d'échantillonage
SNR_dB = 0; %SNR désiré en db
deltaF = 250; %bande passante de la Modulation
tm = 0.05; %période de Modulation
triangles=5; %Nombre de triangles à générer
sigma=1;
if SNR_dB ~= -inf
 SNR = 10^(SNR_dB/20);
 sigma = A/(sqrt(2)*SNR);
else
 SNR = 0;
 sigma = 1; end
ts = 1/fs;
time = (0:ts:tm)';
I_carrier = cos(2*pi*f0.*time);
Q_carrier = sin(2*pi*f0.*time);

f1 = f0 - deltaF/2 + deltaF/tm.*time;
f2 = f0 + deltaF/2 - deltaF/tm.*time;
if triangles==1
 f = 1e-6*[f1;f2];
 elapsed_time = 1e6.*(linspace(0,2*tm,length(f)))';
elseif triangles ==4
 f = 1e-6*[f1;f2;f1;f2;f1;f2;f1;f2];
 elapsed_time = 1e6.*(linspace(0,8*tm,length(f)))';
elseif triangles ==5
 f = 1e-6*[f1;f2;f1;f2;f1;f2;f1;f2;f1;f2]; elapsed_time =
1e6.*(linspace(0,10*tm,length(f)))';
elseif triangles ==6  f = 1e-6*[f1;f2;f1;f2;f1;f2;f1;f2;f1;f2;f1;f2];
 elapsed_time = 1e6.*(linspace(0,12*tm,length(f)))';
end
 if SNR ~= 0
 sI1 = A*cos(2*pi*((f0-deltaF/2).*time + deltaF/(2*tm).*time.^2));
 sI2 = A*cos(2*pi*((f0+deltaF/2).*time - deltaF/(2*tm).*time.^2));
else
 sI1 = zeros(length(time),1); %If noise only, Signal = 0
 sI2 = zeros(length(time),1);
end
sI1_noisy = sI1 + sigma*randn(length(sI1),1);
sI2_noisy = sI2 + sigma*randn(length(sI2),1);
I = [sI1;sI2;sI1;sI2;sI1;sI2;sI1;sI2];
 [a,b]=size(I);
SNR=10^(SNR_dB/10);
power=10*log10(A^2/(2*SNR
noise=wgn(a,b,power);
IwN=I+noise;
```

```
 if SNR ~= 0
 sQ1 = A*sin(2*pi*((f0-deltaF/2).*time + deltaF/(2*tm).*time.^2));
 sQ2 = A*sin(2*pi*((f0+deltaF/2).*time - deltaF/(2*tm).*time.^2));
else
 sQ1 = zeros(length(time),1);
 sQ2 = zeros(length(time),1);
end
sQ1_noisy = sQ1 + sigma*randn(length(sQ1),1);
sQ2_noisy = sQ2 + sigma*randn(length(sQ2),1);
Q = [sQ1;sQ2;sQ1;sQ2;sQ1;sQ2;sQ1;sQ2];
QwN=Q+noise;
figure;
 plot(time(1:50),I_carrier(1:50)); grid;
 title('signal porteuse en phase');
 xlabel('temps (s)'); ylabel('Cos(2*pi*f_0*t)');
 figure;
 plot(time(1:50),Q_carrier(1:50)); grid;
 title('signal porteuse en quadrature');
 xlabel('temps (s)'); ylabel('-Sin(2*pi*f_0*t)');
 figure;
 plot(elapsed_time,f); grid;
 title(' Signa triangulaire')
 xlabel('Time (us)'); ylabel('Frequence (MHz)');
 figure;
 subplot(2,1,1);
 plot(time, sI1_noisy); grid;
 title(['In-Phase, Up-Ramp Transmitted Signal- SNR = ',
 num2str(SNR_dB), ' dB'])
 xlabel('Time (s)'); ylabel('Signal');
figure;
 plot(time, sI2_noisy); grid
 title(['  Signal transmis- SNR = ', num2str(SNR_dB), ' dB'])
 xlabel('temps (s)'); ylabel('Signal');
  figure;
 plot(time, sQ1_noisy);grid;
 title([' Signal transmis - SNR = ', num2str(SNR_dB), ' dB'])
 xlabel('Time (s)'); ylabel('Signal');
 title(['Quadrature,Up-Ramp Transmitted Signal(Near End)-SNR
 =',num2str(SNR_dB),'dB'])
 xlabel('Time (s)'); ylabel('Signal');
 figure;
 plot(time, sQ2_noisy); grid;
 title(['Quadrature, Down-Ramp Transmitted Signal - SNR = ',
num2str(SNR_dB), 'dB'])
 xlabel('Temps (s)'); ylabel('Signal');
 xlabel('temps');ylabel('fréquence'); axis xy;  title('P4');
variable=['F_', num2str(ff), '_', num2str(ffs), '_', num2str(deltaF),
'_', num2str(tm/1000),'_',num2str(SNR_dB),'.jpg'];
  saveas(gcf,variable);
    end
    end
```

Code MATLAB de classification d'images avec le réseau de neurone convolutionel :

```
clear all;
clc;
close all;
disp('Convolutional
Neural Network (CNN) Image Classification in Matlab')
%% la première étape est d'importer les images depuis le dossier
imagededoosier=fullfile('Nouveau');
Categories={'BPSK','CODE_FRANK','FMCW','CODE_P4'};
imds=imageDatastore(fullfile(imagededoosier,Categories),'LabelSource','
foldernames');
% nombre d'image dans chaque catégories
tbl=countEachLabel(imds);
minSetCount = min(tbl{:,2})
 %Utilisez la méthode splitEachLabel pour découper l'ensemble.
 imds = splitEachLabel(imds, minSetCount, 'randomize');
 [imdsTrain,imdsValidation] = splitEachLabel(imds,0.2);
 %.........résau de neuronne choisi
net = squeezenet;
 %......affichage de réseau selectioné
 figure();
plot(net);
title('archetecture de toolbox squeezenzt');
set(gca,'Ylim', [0 20]);
%......... pour savoir les dimensions des images qu'il accepte
net.Layers(1)  ;
 %.......pour savoir les propriétés de la derniére couche
net.Layers(end);

if isa(net,'SeriesNetwork')
  lgraphSqz = layerGraph(net.Layers);
else
  lgraphSqz = layerGraph(net);
end
% on doit remplacer la couche «drop9», la dernière couche de décrochage
du réseau, par une couche
%de décrochage de probabilité 0,6 au lieu de 0.5
tmpLayer = lgraphSqz.Layers(end-5);
newDropoutLayer = dropoutLayer(0.6,'Name','new_dropout');
lgraphSqz = replaceLayer(lgraphSqz,tmpLayer.Name,newDropoutLayer);
%% on doit aussi remplacer La dernière couche apprenable dans
SqueezeNet qui est une couche convolutionnelle
%1 par 1, «conv10».  par une nouvelle couche convolutionnelle avec un
nombre de filtres égal au nombre de types de modulation.
numClasses = 4;
tmpLayer = lgraphSqz.Layers(end-4);
newLearnableLayer = convolution2dLayer(1,numClasses, ...
        'Name','new_conv', ...
        'WeightLearnRateFactor',20, ...
        'BiasLearnRateFactor',15);
```

```
lgraphSqz = replaceLayer(lgraphSqz,tmpLayer.Name,newLearnableLayer);
%Remplacez la couche de classification par une nouvelle de 3 classe
tmpLayer = lgraphSqz.Layers(end);
newClassLayer = classificationLayer('Name','new_classoutput');
lgraphSqz = replaceLayer(lgraphSqz,tmpLayer.Name,newClassLayer);
%.....options d'apprentissage
options = trainingOptions('sgdm', ...
    'MiniBatchSize',3, ...
    'MaxEpochs',6, ...
    'InitialLearnRate',3e-4, ...
    'Shuffle','every-epoch', ...
    'Verbose',false, ...
    'ValidationFrequency',30, ...
    'Plots','training-progress',...
    'ExecutionEnvironment','auto',...
    'ValidationData',imdsValidation);
trainedNet = trainNetwork(imdsTrain,lgraphSqz,options);
%.......................
predicted = classify(trainedNet,imds);
figure
confusionchart(predicted,imds.Labels,'Normalization',
'column-normalized')
```