الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي والبحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

Mémoire de Master

Filière Électronique
Spécialité Instrumentation

Présenté par

Bennanni Zakaria

&

Khour Khadidja

Classification des événements audio environnementaux à l'aide de réseaux de neurones convolutifs CNN

Proposé par : Ykhlef Farid

Année Universitaire 2019-2020

Nous remercions d'abord ALLAH le tout puissant qui nous a guidés et donnés la force et la volonté de réaliser cette mémoire. Nôtres pensés vont vers nos parents, qui ont toujours cru en nous. C'est grâce à leurs soutient et prières que nous avons accomplis ce travail.

Nous remercions notre promoteur Mr Ykhlef Farid d'avoir pris en charge et aidé tout au long du projet. Et de nous avoir orientés avec ces précieux conseils et son soutient moral vue ces dernières conséquences du COVID_19.

Remerciements les plus sincères à toutes les personnes qui ont de près ou de loin à l'évaluation de cette mémoire. Enfin, nous tenons aussi à remercier les jurés pour avoir accepté et de juger notre travail.

ملخص: مع انتشار استخدام النماذج القائمة على التعلم العميق في مشاكل التصنيف المختلفة وقوتها المثبتة مقارنة بالطرق التقليدية. يستخدم أداء النماذج الحالية ميزات سمعية مثل معاملات تردد سيبسترال ميل ومخطط الطيف لوغاريتم ميل لتدريب الشبكات العصبية العميقة لتصنيف الصوت البيئي. في هذه الأطروحة، نقترح شبكة عصبية تلافيفيه مكونة من 5 طبقات للتدريب على مجموعة بيانات صوتية. سنثبت أن الشبكات العصبية التلافيفية يمكن تطبيقها بنجاح في تصنيف الصوت البيئي حتى مع مجموعات البيانات المحدودة.

كلمات المفاتيح: التعلم العميق؛معاملات تردد سيبسترال ميل؛ مخطط الطيف لوغاريتم ميل؛الشبكات العصبية .التلافيفية؛تصنيف الصوت البيئي.

Résumé : Avec la popularité de l'utilisation de modèles basés sur l'apprentissage profond (Deep learning) dans divers problèmes de classification et leur robustesse prouvée par rapport aux méthodes conventionnelles. Les performances des modèles existants utilisent des fonctionnalités auditives telles que les coefficients cepstrales de fréquence mel (MFCC) et le spectrogramme Log-Mel (LM) pour entraîner les réseaux de neurones profonds pour la classification sonore de l'environnement (ESC). Dans ce mémoire, nous proposons un réseau neuronal convolutif à 5 couches (CNN) pour l'entraîner sur un ensemble de données audios. On va prouver que les réseaux de neurones convolutifs CNN peuvent être appliqués avec succès dans la classification environnementale sonore (ESC) même avec des ensembles de données limités.

Mots clés: Deep learning; MFCC; LM; ESC; CNN.

Abstract: With the popularity of the use of models based on deep learning (Deep learning) in various classification problems and their proven robustness compared to conventional methods. The performance of existing models uses auditory features such as cepstral mel frequency coefficients (MFCC) and Log-Mel spectrogram (LM) to train deep neural networks for environmental sound classification (ESC). In this thesis, we propose a 5-layer convolutional neural network (CNN) to train on an audio dataset. We will prove that CNN convolutional neural networks can be successfully applied in environmental sound classification (ESC) even with limited data sets.

Keywords: Deep learning; MFCC; LM; ESC; CNN.

Table des matières

Liste des figures

Liste des tableaux

Acronyme et abréviation

ESC : Environmental Sound classification.

CNN : Convolutional neural network.

ZCR : Taux de passage par zéro.

MFCC : Les coefficients cepstraux Mel Frequency.

FFT : Fast Fourier transform.

DCT : La transformée en cosinus discrète.

KNN : K nearest neighbors.

SVM : Machines à support vectoriel.

RVB : Rouge, Vert, Bleu.

Relue : Unité linéaire rectifiée.

ANN : Les réseaux de neurones artificiels.

VM : Machines virtuelles.

WOLA : Windowed Over-Lapping Analysis.

Introduction générale

L'apprentissage automatique (ou Machine Learning « ML » en anglais) est une technologie de l'intelligence artificielle permettant aux ordinateurs d'apprendre sans avoir été programmés précisément à cet effet. Il est considéré comme étant une liaison entre le monde humain et le monde artificielle. Toutefois, les ordinateurs ont besoin de données audio à analyser et à entraîner pour apprendre et se développer.

L'apprentissage automatique est un outil important dans le domaine de la classification des évènements audio. De nombreuses tâches de classification peuvent être résolues en entraînant un classifieur sur un ensemble de données audios. Pour les tâches de classification se basant sur l'audio, il est possible d'extraire les caractéristiques de ce dernier à l'aide de méthodes du traitement de signal. Ces caractéristiques apprises permettent souvent d'améliorer la performance sur une tâche de classification donnée. Pour apprendre des représentations des classes, il est important de considérer les aspects particuliers à l'audio dans la conception des modèles d'apprentissage. Vu la structure temporelle et spectrale de l'audio, les représentations profondes et multi échelles sont particulièrement bien conçues pour représenter la classification audio.

Un réseau neuronal convolutif ou CNN est un réseau neuronal d'apprentissage en profondeur former pour traiter des tableaux structurés de données telles que des images. Les réseaux de neurones convolutifs sont largement utilisés dans la vision par ordinateur et sont devenus l'état de l'art pour de nombreuses applications visuelles telles que la classification d'images, et ont également trouvé du succès dans le traitement du langage naturel pour la classification de texte. Dans notre travail nous nous intéressons à l'application des réseaux de neurones convolutifs pour la classification des sons environnementaux.

Cette étude a deux étapes principales : premièrement, nous convertissons le signal audio à des MEL-spectrogrammes pour une taille accessible à l'entré de notre classifieur. Deuxièmement, nous choisissons une architecture de réseau de neurones à convolution profonde adapté pour la classification des sons environnementaux **[CSDDV] [WDKR]**.

Chapitre 1 Généralités sur le traitement du signal audio

1.1 Introduction

Historiquement, l'intérêt pour les sons a été porté par la volonté de comprendre et maitriser les phénomènes sonores dans le contexte de la musique. Les pythagoriciens ainsi que Aristote et entre autres, se sont intéressés à la vibration des cordes et au comportement ondulatoire des ondes. Très tôt les sons ont été associés à une onde, malgré qu'il ait fallu attendre le XIIe siècle et les travaux de Mersenne et Galilée pour que l'acoustique devienne une véritable science.

Physiquement, le son est alors décrit comme une onde mécanique de pression se propageant dans un milieu. Le Britannique Boyle, le Hollandais Huygens, les Suisses Euler et Bernoulli et le Turinois Lagrange ont collaboré aux XIIe et XIIIe siècles à apporter les bases de l'étude des phénomènes sonores. Au XIXe, l'apparition de techniques expérimentales a permis à Regnault puis Helmotz d'améliorer la compréhension de l'acoustique, la perception des sons et l'analyse des sons complexes. Les théories les plus récentes reposent encore sur les résultats de cette époque. En 1877, le futur prix Nobel lord Rayleigh a fixé l'étendue des connaissances acquises jusque-là dans son ouvrage Theory of Sound. Rééditée plusieurs fois, cette œuvre en deux volumes reste encore aujourd'hui une référence.

Michel Chion montre dans son essai que le monde des sons, souvent méconnu, reflète par bien des aspects le monde qui nous entoure. Au-delà de cette compréhension, la maitrise technique permet également d'analyser l'environnement à partir des sons, naturels ou provoqués. Ainsi, l'étude des sons permet de traiter nombre de sujets, de la voix humaine au bang sonique en passant par la cartographie acoustique ou la thérapie ultra-sonore. L'analyse sonore est donc une discipline variée **[ADESA] [EDAOSED].**

Le traitement automatique du son consiste à réaliser des opérations sur un signal audio pour cela dans ce chapitre, nous rappelons dans un premier temps la manière ou le son est numérisé. Puis, nous nous intéressons à décrire la représentation du signal sonore par des paramètres acoustiques, ces paramètres sont nommés les descripteurs acoustiques aussi on a défini plusieurs domaines de descripteurs. Nous abordons ensuite le problème de la sélection automatique de ces paramètres.

1.2 L'analyse du son dans le domaine numérique

1.2.1 Numérisation

Les signaux analogiques peuvent prendre une infinie de valeur et sont continus dans le temps. Par contre, les signaux numériques sont :

- **Échantillonnés :** les échantillons sont discontinus dans le temps, parce qu'ils n'existent qu'aux instants d'échantillonnage.
- **Quantifiés :** ils ne peuvent prendre qu'un nombre fini de valeurs quantifiées.

La **Figure 1.1** suivante montre la différence entre un signal analogique (continue dans le temps et en valeurs) et un signal numérique (discontinu dans le temps et ne prenant que 8 valeurs quantifiées).

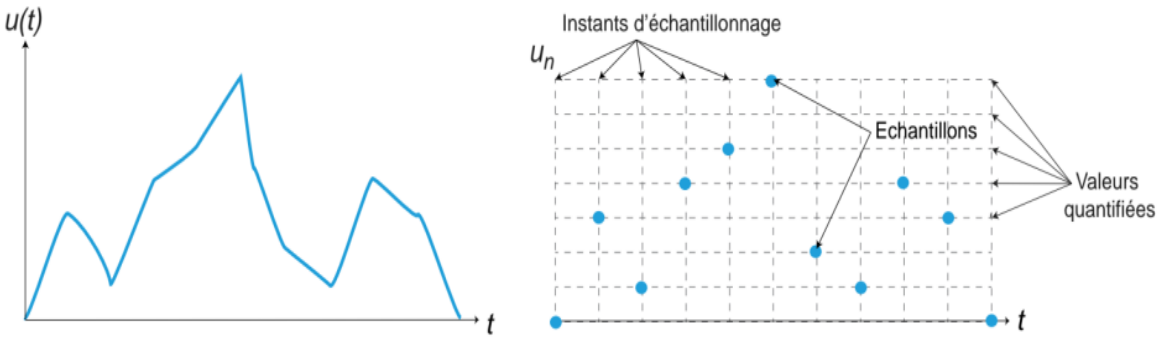


Figure 1.1 : signal analogique et signal numérique.

Il faut passer par deux étapes pour convertir un signal analogique en signal numérique qui sont l'échantillonnage et la quantification (**Figure 1.2**).

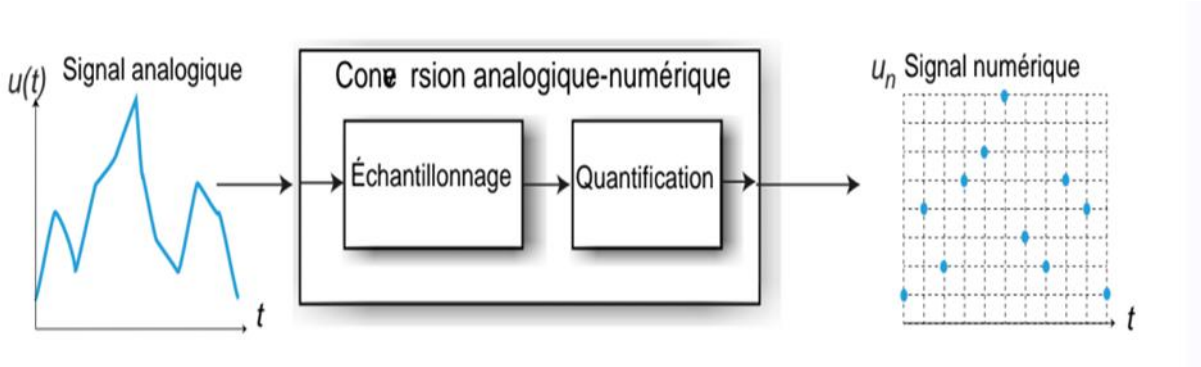


Figure 1.2 : étape de la conversion analogique numérique.

a L'échantillonnages

L'échantillonnage se compose de prélèvement des valeurs du signal à intervalles de temps réguliers (**Figure 1.3**). On obtient une suite d'échantillons espacés de Te notée U*(t) et appelée signaux échantillonnés.

Période d'échantillonnage Te : Te représente l'intervalle de temps en second entre chaque échantillon.

Fréquence d'échantillonnages ($f_e = 1/Te$) : c'est le nombre d'échantillons par seconde (en Hz).

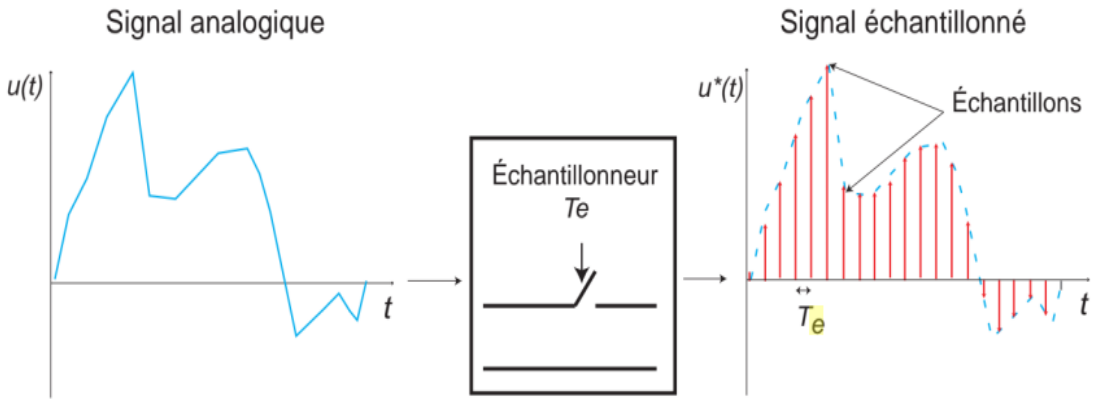


Figure 1.3 : principe de l'échantillonnage.

b La quantification

Un signal analogique échantillonné, malgré discontinu dans le temps, peut toujours prendre une infinité de valeurs au contraire du signal numérique qui ne peut prendre

qu'un nombre fini. La quantification diminue le nombre de valeurs possibles pour un signal et c'est une opération irréversible.

Quantifier un échantillon consiste à approcher sa valeur par une autre valeur quantifiée et codée par un mot de N bites le plus souvent. La quantification est :

- **Centrée** : on arrondit à la valeur quantifiée la plus proche.
- **Uniforme** : l'intervalle entre deux valeurs quantifiées est constant.

Toutes les étapes précédentes sont mentionnées dans **[TSNOPC].**

1.2.2 Fenêtrage en trames

En vue d'analyser le signal audio numérisé, on détermine des fenêtres temporelles de quelques dizaines à quelques centaines d'échantillons (soit de quelques millisecondes à quelques secondes).

Ces fenêtres d'analyse, appelées trames (frames en anglais), se recouvrent généralement les unes avec les autres et sont pondérées de façon à garder l'amplitude d'origine aux points de recouvrement. Il s'agit alors d'analyse fenêtrée avec recouvrement ou WOLA (Windowed Over-Lapping Analysis). La pondération est souvent une fenêtre de Hanning ou de Hamming.

Sachons que ce découpage peut être échantillonné, autrement dit qu'on peut garder qu'une partie des trames. Le choix des trames est réalisé soit par expertise humaine, soit par l'application d'algorithmes spécifiques **[ADESA].**

1.2.3 Extraction des descripteurs acoustiques

La représentation de chaque trame par un ensemble de grandeurs physiques, appelées paramètres acoustiques ou descripteurs, calculées à partir des valeurs de ses échantillons et donnent l'information qu'elle contient. Le choix d'une fenêtre temporelle adaptée permettant généralement de vérifier que le signal est stationnaire à l'horizon d'une trame

Par expansion, on appelle aussi trame un vecteur de descripteurs acoustiques qui constitue la représentation numérique élémentaire du signal sonore.

Le choix des descripteurs dépend en grande partie des capacités d'un système. On peut aussi ne conserver qu'une partie des descripteurs extraits **[ADESA].**

1.3 Descripteurs pour la classification audio

Un signal audio peut être observé grâce à sa forme d'onde, décrivant les variations d'amplitude de l'onde acoustique au cours du temps **(Figure 1.4**). Néanmoins, cette représentation brute ne permet pas une caractérisation suffisante pour pouvoir être utilisée par la suite en classification. Par conséquent, pour tout système de classification audio la première étape consiste à extraire des descripteurs qui font ressortir certaines propriétés du signal jugées appropriées afin de séparer les différentes classes audios. Pour cela, le signal est découpé en segments élémentaires, nommées trames, changeant selon les applications d'une dizaine à une centaine de millisecondes. Durant ce découpage, le pas d'avancement peut correspondre à la taille de la trame **[CASCFL] [EDAOSED].**

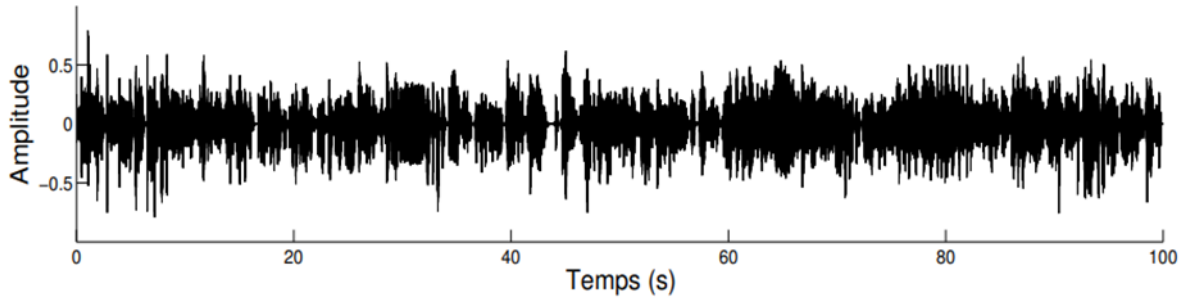


Figure 1.4 : Forme d'onde d'un extrait audio contenant de la musique et de la parole **[CASCFL].**

Généralement la classification audio utilise des descripteurs audios. Ces derniers sont extraits des différentes représentations des signaux :

- **Temporelle** : le son brut dépendant du temps est utilisé pour calculer les descripteurs temporels. Ils contiennent surtout l'énergie, les coefficients d'autocorrélation et le taux de passage par zéro.

- **Spectrale** : une représentation fréquentielle ou locale de type Transformée de Fourier à court terme (TFCT) est utilisée. Ces descripteurs comportent principalement les moments spectraux (centroïde, spread, skewness, kurtosis).
- **Cepstrale et Perceptuelle et harmonique** : les descripteurs sont basés sur le cepstre du son, c'est-à-dire la TF de la log-amplitude de la TF du son. Le son est décomposé selon une autre échelle de fréquence, supposée représenter loyalement l'audition humaine, comme l'échelle Mel, Bark ou ERB (Equivalent Rectangular Bandwidth).

Les descripteurs les plus appliqués dans le cas de la classification sont l'énergie du signal, les MFCC et la fréquence fondamentale **[CSSST].**

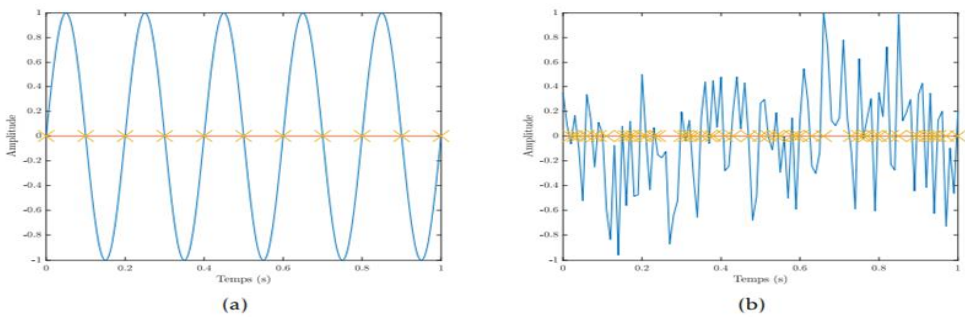
1.3.1 Descripteurs temporels

Les descripteurs temporels sont retirés de la forme d'onde du signal seulement. Étant donné qu'aucune transformation n'est nécessaire, généralement ces descripteurs présentent l'avantage d'être faiblement complexes **[CASCFL].**

a Taux de passage par zéro

Le (ZCR) est le taux de passage par zéro qui représente le nombre de fois que le signal passe par zéro. Il est aussi considéré comme descripteur temporel.

Le taux de passage par zéro permet de séparer les signaux bruités (qui présentent des valeurs ZCR élevées - fort ZCR) des signaux non bruités (qui présentent des valeurs ZCR faibles - faible ZCR) (voir **Figure 1.5**). Par exemple le ZCR permet de discriminer les sons voises des sons non-voises, il est connu dans la séparation parole/musique **[CSSST] [SOVAZCR].**



(a) (b)

Figure 1.5 : Illustration du ZCR sur deux signaux différents. (a) sinusoïdal le ZCR est plus tôt faible. (b) Pour un signal bruité le ZCR est plutôt fort **[CSSST].**

b L'Energie

L'énergie dépend du calcul de La racine des carrés moyens (Root Mean Square, RMS) de l'amplitude et parfois elle est appliquée pour mesurer l'intensité sonore dans une trame. Par exemple dans le cas de la séparation parole/musique, on peut montrer que la parole présente une plus grande variation d'énergie que la musique

Le calcul de L'énergie du signal, ou son volume sur une trame est un indicateur de détection du silence et de précision de la frontière d'un segment. C'est également un critère pour discerner par exemple un signal composé de plusieurs pics d'énergie d'un signal plus stable. Elle est calculée pour une trame t comme la puissance quadratique moyenne de l'amplitude du signal **[SDACSE]**.

$$\boldsymbol{e(t)} = \sqrt{\frac{\mathbf{1}}{\boldsymbol{N_t}} \sum_{\boldsymbol{i=0}}^{\boldsymbol{N_{T-1}}} \boldsymbol{s_t^2(i)}} \qquad \textbf{(1)}$$

c Autocorrélation

L'autocorrélation permet d'estimer la distribution fréquentielle du signal audio depuis le domaine temporel. Cette analyse consiste à multiplier les échantillons d'une trame avec une copie d'eux-mêmes, après avoir appliqué un délai à cette copie. On recherche alors des pics parmi les premiers coefficients ainsi obtenus pour identifier les périodicités du signal. Chaque délai diffèrent permet l'analyse d'une fréquence particulière **[GPEE]**.

Ce descripteur a été utilisé avec succès par Brown pour la reconnaissance automatique des instruments de musique [Brown, 1998].

$$(\boldsymbol{k}) = \frac{1}{x(0)^2} \sum_{n=0}^{N-K-1} x(n).x(n+k) \qquad \textbf{(2)}$$

1.3.2 Descripteurs cepstraux et perceptifs

a MFCC (Les coefficients cepstraux Mel-fréquence)

Les coefficients cepstraux Mel Frequency MFCC sont une représentation compacte du spectre audio originalement formulé pour saisir les caractéristiques importantes de la parole, les MFCC ont été utilisées par un bon nombre d'auteurs pour paramétriser le flux de la classification automatique. Ils ont été introduits par Davis et Mermelstein dans les années 1980 et sont depuis lors à la pointe de la technologie **[MFCCT] [ZCODA]**.

Les étapes d'extraction des coefficients MFCC de son :

1. Cadrez le signal en trames courtes.
2. Calculez l'estimation du périodogramme du spectrogramme pour chaque trame.
3. Appliquez le banc de filtres mel du spectrogramme.
4. Additionnez l'énergie de chaque filtre et prenez le logarithme de toutes les énergies du banc de filtres.
5. Prenez le DCT des énergies du banc de filtres log.

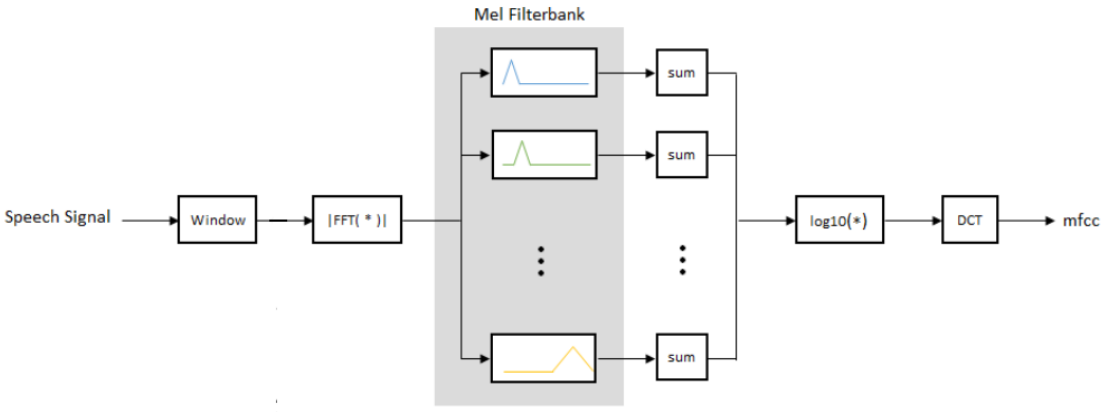


Figure 1.6 : les principales étapes pour le calcul des MFCC **[SIUPMFCC].**

On aborde la description détaillée du calcul des MFCC :

1. La première étape est de cadrer le signal en trames courtes. Un signal audio change constamment donc pour simplifier les choses, on suppose que sur des échelles de temps courtes le signal audio ne change pas beaucoup (c'est-à-dire statistiquement stationnaire). En effectuant une transformée de Fourier sur cette trame de temps court, on peut obtenir une bonne approximation des contours de fréquence du signal en enchaînant des trames adjacentes, c'est pourquoi on encadre le signal dans des trames de 20 à 40 ms. Si la trame est

beaucoup plus courte, on n'a pas assez d'échantillons pour obtenir une estimation spectrale fiable, si elle est plus longue, le signal change trop tôt au long de la trame **[MFCCT]**.

Après avoir découpé le signal en trames, on applique une fonction de fenêtre telle que la fenêtre De Hamming à chaque trame. Une fenêtre de Hamming à la forme suivante **[SPFML]**

$$w[n] = 0.54 - 0.46cos(\frac{2\pi n}{N-1}) \quad \textbf{(3)}$$

Où, 0≤n≤N − 1, N est la longueur de la fenêtre. Le tracé de l'équation précédente donne le tracé suivant **[SPFML] :**

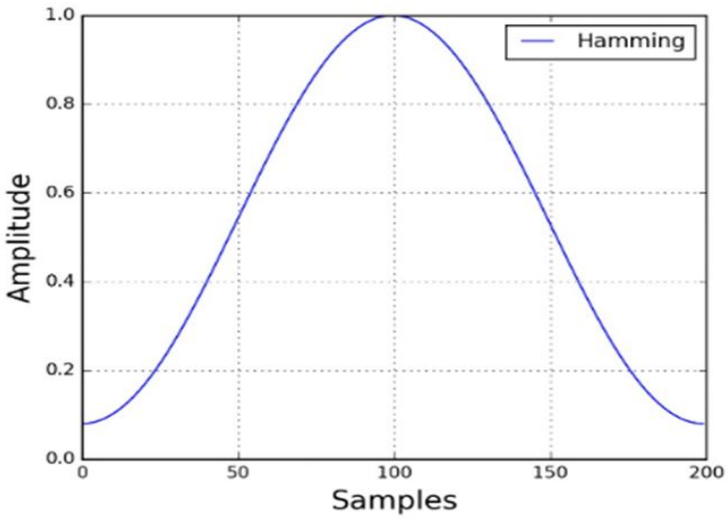


Figure 1.7*:* Hamming Window **[SPFML].**

Exemple pour calculer le nombre d'échantillons (Samples) dans une trame : cadrez le signal dans des trames de 20 à 40 ms, 25 ms standard. Cela signifie que la longueur de trame pour un signal 16 kHz est de 0,025 * 16000 = 400 échantillons (Samples).

2. L'étape suivante consiste à calculer le spectrogramme de chaque trame. Ceci est motivé par la cochlée humaine (un organe dans l'oreille) qui vibre à différents endroits en fonction de la fréquence des sons entrants. En fonction de l'emplacement de la cochlée qui vibre (qui fait trembler les petits poils), différents nerfs se déclenchent pour informer le cerveau que certaines

fréquences sont présentes. L'estimation de spectrogramme effectue un travail similaire identifiant les fréquences présentes dans les trames [MFCCT].

La transformée de Fourier rapide (FFT : fast Fourier transform) est une fonction qui reçoit un signal dans le domaine temporel en entrée, et sort sa décomposition en fréquences.

On Prend par exemple un clip audio, le séparant en fenêtres temporelles et appliquant la transformation de Fourier à chaque fenêtre temporelle **(Figure 1.8)**, puis ces fenêtres se recouvre les unes avec les autres et sont pondérées de façon à garder l'amplitude d'origine aux points de recouvrement, et de ce fait on aura ce que nous appelons un spectrogramme **[GTKTMS].**

On ne voit pas grand-chose parce que la plupart des sons que les humains entendent sont concentrés dans de très petites plages de fréquences et d'amplitudes.

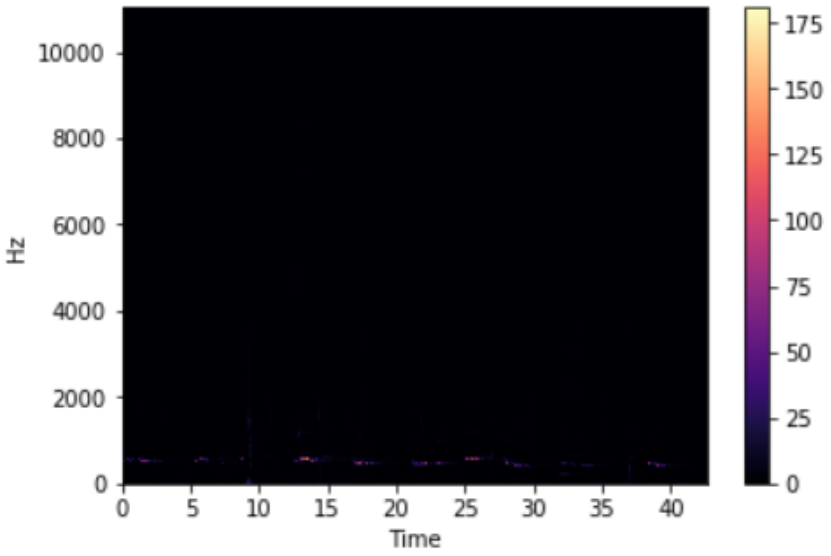


Figure 1.8 : Un spectrogramme d'un clip audio **[GTKTMS].**

3. La cochlée ne peut pas discerner la différence entre deux fréquences étroitement espacées. Donc, on prend des tas de cases de spectrogramme qu'on résume pour avoir une idée de la quantité d'énergie qui existe dans diverses régions de fréquence.

Ceci est effectué par une banque de k filtres (**Figure 1.9**) positionnés selon une échelle Mel. Cette échelle est construite à partir d'une série de filtres passe-bandes de formes triangulaires positionnés d'une façon linéaire pour les basses fréquences (<1000 Hz) et logarithmique pour les hautes fréquences. Le premier filtre (donne une indication sur la quantité d'énergie existante près de 0 Hertz) est très étroit, en passant vers les hautes fréquences cette largeur augmente, ceci imite le fonctionnement de l'oreille humaine qui est insensible dans ces fréquences et qui ne peuvent pas distinguer entre deux fréquences très proches **[MFCCT].**

Le premier banc de filtres débutera au premier point, atteindra son pic au deuxième point, puis retournera à zéro au troisième point. Le deuxième banc de filtres commencera au 2ème point, atteindra son maximum au 3ème, puis sera nul au 4ème etc.

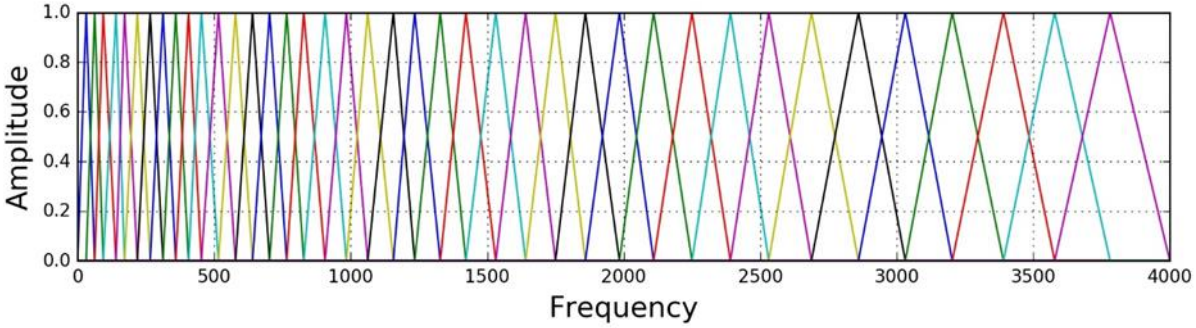


Figure 1.9 : La répartition des filtres triangulaires sur les échelles Mel **[SPFML].**

La formule de conversion de l'échelle de fréquence en échelle de Mel est **(FigureI.10) [MFCCT] :**

$$\boldsymbol{M(f) = 1125 ln\left(1 + {f}/{700}\right)} \quad \textbf{(4)}$$

Pour passer du retour de Mel à la fréquence : **[MFCCT]**

$$\boldsymbol{M^{-1}(m) = 700\left(e^{\frac{m}{1125}} - 1\right)} \quad \textbf{(5)}$$

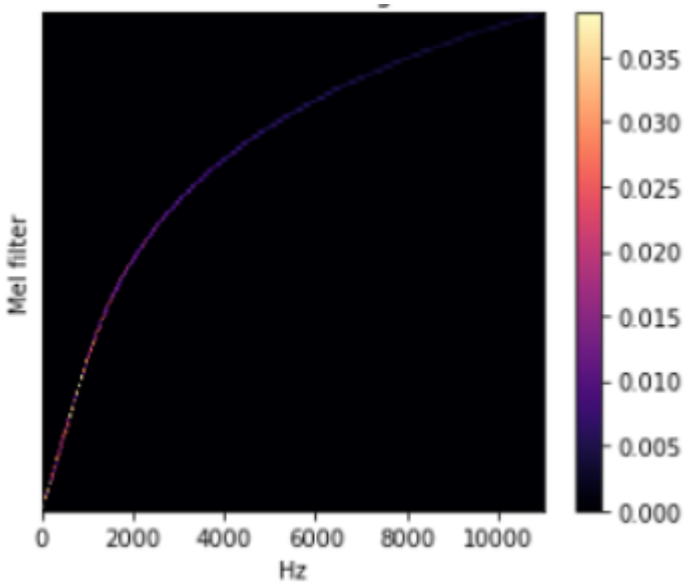


Figure 1.10 : la conversion de Hz en mels **[GTKTMS].**

4. L'étape suivante, on additionne l'énergie de chaque filtre puis on prend le logarithme de toutes les énergies du banc de filtres. Pour calculer les énergies du banc de filtres, on multiplie chaque banc de filtres par le spectre de puissance (spectrogramme), puis on additionne les coefficients **[MFCCT].**

   Une fois qu'on a les énergies du banc de filtres et le logarithme de toutes les énergies, on obtient alors une visualisation du son dans cette nouvelle échelle de fréquence ce que nous appelons un mel spectrogramme (**Figure 1.11)** [**MFCCT].**

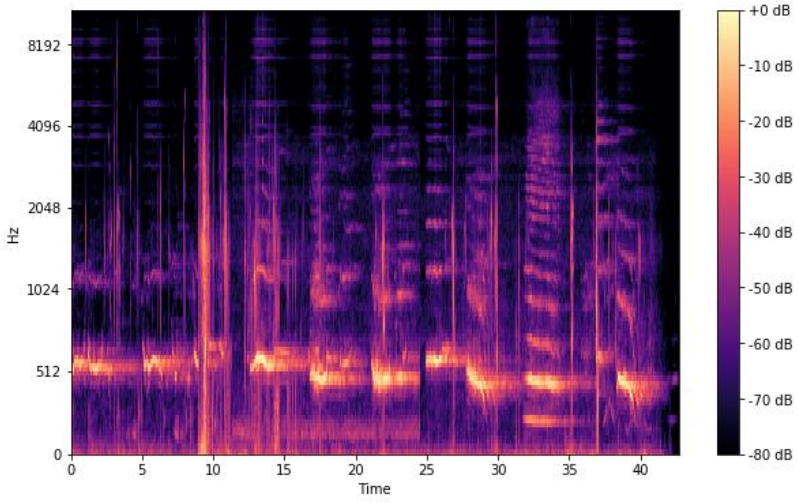


Figure 1.11 : Mel spectrogramme **[GTKTMS].**

5. Cette étape de ce processus consiste à calculer la transformée en cosinus discrète (DCT) des énergies logarithmiques E_i du banc de filtre pour une compression possible de l'information. L'ensemble de ces coefficients cepstraux produits est appelé " vecteur acoustique ". L'équation de la DCT est la suivante **[MFCCT] :**

$$\boldsymbol{c_k = \sum_{i=0}^{M} logE_i \, cos\left[\frac{\pi k}{M}\left(i - \frac{1}{2}\right)\right]} \qquad \boldsymbol{1 \leq k \leq d} \qquad \textbf{(6)}$$

Il y a 2 raisons principales pour lesquelles DCT est effectué. Comme nos bancs de filtres se chevauchent tous, les énergies du banc de filtres sont assez corrélées les unes aux autres. Le DCT décorrèle les énergies. Les coefficients DCT plus élevés montrent des changements rapides dans les énergies du banc de filtres **[MFCCT].**

Le vecteur de caractéristiques MFCC décrit seulement l'enveloppe spectrale de puissance d'une seule trame, mais quelles sont les trajectoires des coefficients MFCC au fil du temps ? Il s'avère que le calcul des trajectoires MFCC et leur ajout au vecteur de caractéristiques d'origine augmente considérablement les performances de la reconnaissance automatique des paroles (si nous avons 12 coefficients MFCC, nous obtiendrons également 12 coefficients delta, qui se combineront pour donner un vecteur de caractéristiques de longueur 24). Ces coefficients représentent la vitesse et l'accélération. **[MFCCT].**

La formule suivante est utilisée Pour calculer les coefficients delta :

$$\boldsymbol{dt = \frac{\sum_{n=1}^{N} n(c_{t+N} - c_{t-N})}{2\sum_{n=1}^{N} n^2}} \qquad \textbf{(7)}$$

Où $\mathbf{d_t}$ est un coefficient delta, de la trame *t* calculée en termes de coefficients statiques C_{t+N} à C_{t-N}. Une valeur typique pour *N* est 2. Les coefficients Delta-Delta (Accélération) sont calculés de la même manière mais à partir des deltas pas des coefficients statiques **[MFCCT]**.

1.3.3 Descripteurs harmoniques

a Fréquence fondamentale, pitch

Pour un signal harmonique, la fréquence fondamentale est la fréquence ou le multiple explique le mieux possible le contenu du spectre. La fréquence fondamentale, notée f0, correspond à une notion de hauteur des sons harmoniques. Chaque son peut se décomposer en une somme de sinusoïdes ou l'amplitude varie plus ou moins lentement dans le temps. Dans le cas où le son est harmonique, la décomposition du son donne des sinusoïdes prépondérantes dont les fréquences seront toutes multiples, plus ou moins exactement entiers, d'une fréquence que l'on qualifie alors de fréquence fondamentale. Par abus de langage, on confond souvent pitch et hauteur avec la fréquence fondamentale **(Figure 1.12). [GPEE]**

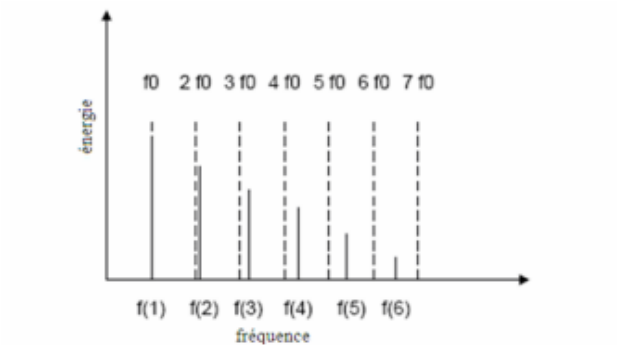


Figure 1.12 : Représentation des pics d'un signal (en lignes continue) et de ses harmoniques (en pointillé).

b Non-Harmonicité

Elle représente la divergence en fréquences du spectre du signal total par rapport à celui de sa partie harmonique, et elle vérifie donc si le signal total possède les mêmes fréquences que sa partie harmonique.

$$Inharmo = \frac{2}{f_0} \frac{\sum_n |f(n) - n.f_0| . M_t^2(n)}{\sum_n M_t^2(n)} \quad \textbf{(8)}$$

Ses valeurs sont comprises entre **0** et **1**. Elle prend **0** pour un signal purement harmonique et **1** pour un signal complètement non harmonique (bruit) **[ZCODA]**.

c Spectre-harmoniques de déviation

La déviation du spectre en amplitude du signal brut par rapport à celui de sa partie harmonique est nommée spectre-harmoniques de déviation.

$$\boldsymbol{HDEV} = \frac{\mathbf{1}}{\boldsymbol{N}}\sum_{\boldsymbol{n}}(\boldsymbol{a}(\boldsymbol{n}) - \boldsymbol{M}(\boldsymbol{n})). \quad \textbf{(9)}$$

N est le nombre d'harmoniques, *a(n)* est amplitude de la $n^{iéme}$ harmonique et *M(n)* est amplitude du spectre évalué à la fréquence associée à l'indice n **[ZCODA]**.

1.3.4 Descripteurs spectraux

a Les moments statistiques spectraux

Sont les quatre premiers moments statistiques calculés sur le spectre :

- ***Centroïde spectral***

  Le Centroïde spectrale est une mesure utilisée dans le traitement de signal numérique pour caractériser le spectre. Il estime le point d'équilibre entre les hautes et les basses fréquences du spectre (point du centre de gravité du spectre). Le Centroïde spectral est calculé comme une moyenne équilibrée des fréquences présentes dans le signal, avec leurs amplitudes prises comme poids **[CASCFL] :**

  Ou $\boldsymbol{M_t(n)}$ est la transformée de Fourier du signal à l'instant « t » et à la fréquence associé à l'indice n.

$$\boldsymbol{C_t} = \frac{\sum_{n=1}^{N} \boldsymbol{M_t(n).n}}{\sum_{n=1}^{N} \boldsymbol{M_t(n)}} \quad \textbf{(10)}$$

- ***Largeur spectrale***

Décrivant l'étendue du spectre autour de sa moyenne **[ZCODA] :**

$$\boldsymbol{ES} = \frac{\sum_{n=1}^{N}(\boldsymbol{n}-\boldsymbol{c_t})^2.\boldsymbol{M_t(n)}}{\sum_{n=1}^{N} \boldsymbol{M_t(n)}} = \boldsymbol{m_2} \quad \textbf{(11)}$$

- ***L'aplatissement spectrale***

Il évalue l'asymétrie de la distribution du spectre autour de sa moyenne. Il est calculé depuis le moment d'ordre 3. Un aplatissement spectral nul désigne une distribution symétrique, un aplatissement spectral négatif indique qu'il y a plus d'énergie à droite alors qu'un aplatissement spectral positif indique qu'il y a plus d'énergie à gauche **[GPEE].**

Figure 1.13 : Aplatissement spectral négatif (gauche) et aplatissement spectral positif (droite).

L'aplatissement spectral est défini comme suit :

$$\boldsymbol{AS} = \frac{\boldsymbol{m_3}}{\boldsymbol{\sigma_3}} \qquad \textbf{(12)}$$

$$\textit{Avec :}\ \ \boldsymbol{m_3} = \frac{\sum_n^N (n-c_t)^3 . M_t(n)}{\sum_n^N M_t(n)} \quad \textit{et} \quad \boldsymbol{\sigma_3} = (\boldsymbol{m_2})^{3}/_{2} \qquad \textbf{(13)}$$

- ***La platitude spectrale (kurtosis)***

La platitude spectrale (SK pour spectral kurtosis) est une mesure de l'aplatissement du spectre autour de sa moyenne, elle est considérée comme le $4^{éme}$ moment de la variable centrée-réduite $\boldsymbol{X} \mapsto \frac{\boldsymbol{X-u}}{\boldsymbol{\sigma}}$:

$$\boldsymbol{SK} = \frac{-3\mu_1^4 + 6\mu_1\mu_2 - 4\mu_1\mu_3 + 4}{SW^4} \qquad \textbf{(14)}$$

b La pente spectrale

Elle représente le taux de décroissance de l'amplitude spectrale, elle est donnée par **[GPEE] :**

$$\boldsymbol{DS} = \frac{1}{\sum_{n=2}^N M_t[n]} \sum_n^N \frac{M_t[n] - M_t[1]}{n-1} \qquad \textbf{(15)}$$

c Fréquence de coupure (FC)

C'est la fréquence du spectre en dessous de laquelle généralement 95% de l'énergie spectrale est prise en compte. Elle permet de donner par exemple des informations de registre, un son plus grave présentant une fréquence de coupure plus basse **[CASAF].**

d Le flux spectral

Le flux spectral est la mesure de la vitesse de changement du spectre de puissance de signal, calculé en comparant le spectre de puissance entre deux trames successives, il est calculé par :

$$\boldsymbol{Flux} = 1 - \frac{\sum_n \boldsymbol{M_{t-1}(n).M_t(n)}}{\sqrt{\sum_n \boldsymbol{M_{t-1}(n)^2}}.\sqrt{\sum_n \boldsymbol{M_t(n)^2}}} \quad \textbf{(16)}$$

où $M_{t\,(n)}$ et $M_{t-1\,(n)}$ sont les amplitudes du spectre aux instants t et t-1 du signal.

Lorsque le flux est égal à 0, les deux trames sont similaires et dans le cas où le flux égal à 1 elles sont complétement différentes **[ZCODA].**

1.4 Conclusion

Dans ce chapitre nous avons parlé de l'analyse du son ou on a différencié toutes les étapes pour l'extraction des paramètres acoustiques qui décrivent le signal audio, nommé les descripteurs. Puis nous avons abordé les différents descripteurs qui sont les descripteur temporels (aux de passage par zéro, l'Energie, Autocorrélation), les descripteurs cepstraux et perceptif (MFCC), les descripteurs harmoniques (Fréquence fondamental, Non-harmonicité, Spectre-harmonique de déviation) et les descripteurs spectraux (Les moments statique spectraux, La pente spectrale, Fréquence de coupure FC, Le flux spectrale).

L'extraction des caractéristiques est une partie très importante dans l'analyse et la recherche de relations entre différentes choses. Les données fournies de l'audio ne peuvent pas être comprises par les classifieurs directement. Pour ce fait, nous convertirons les données audios en un format cohérent par l'extraction des caractéristiques qui est un processus expliquant la plupart des données de façon compréhensible.

Ainsi, pour chaque besoin donné, une description « experte » du signal est proposée (descripteurs de timbre pour la reconnaissance des instruments de musique, MFCC pour l'analyse de la parole, etc.…). Dans notre travail, nous utiliserons spécialement le Mel spectrogramme.

Chapitre 2 Apprentissage automatique (Machine Learning)

2.1 Introduction

Dans le domaine du Machine Learning (apprentissage automatique), il existe deux principaux types d'apprentissages : supervisées et non supervisées. La différence entre ces deux types réside dans le fait que l'apprentissage supervisé est basé sur une vérité. En d'autres termes, nous avons une connaissance préalable qui devraient être les valeurs de sortie de nos échantillons. Par conséquent, l'objectif de l'apprentissage supervisé est d'avoir une fonction à partir d'un échantillon de données et des résultats souhaités se rapprochant dans les relations entre les entrées et les sorties observables dans les données. Par contre, l'apprentissage non supervisé n'a pas de résultats étiquetés. Le but principal de ce dernier est donc de déduire la structure naturelle présente dans un ensemble de points de données**.**

Les méthodes d'apprentissage que nous avons retenues reposent sur la représentation vectorielle des signales audio, aussi sur un prétraitement d'une matrice qui sera divisée aléatoirement en sous-ensembles d'apprentissage ainsi traitée par les méthodes d'apprentissage **[CTCAS] [ASVNS] [EDAOSED].**

Dans la classification, chaque vecteur Γ^{μ} appartient à une classe $\tau^{\mu} = g(\Gamma^{\mu})$, g étant une fonction inconnue. L'ensemble d'exemples dont on dispose, appeler l'ensemble d'apprentissage, consiste en *P* couples d'entrée-sortie $(\overrightarrow{\Gamma^{\mu}}, \tau^{\mu})$; μ=1, ..., *P.* où les sorties τ^{μ} (les classes) sont connues. Nous décrivons cet ensembles $\Lambda = g\{(\overrightarrow{\Gamma^{\mu}}, \tau^{\mu})$; μ=1, ..., *P*}. Si nous utilisons un apprentissage supervisé, un classifieur nécessite la connaissance des

classes τ^{μ} afin d'avoir la fonction g. Par conséquent, l'apprentissage non supervisé peut ignorer τ^{μ} **[CTCAS].**

2.2 Apprentissage automatique supervisé

Apprentissage supervisé (Supervised Learning) est le paradigme d'apprentissage le plus populaire en Machine Learning. Comme son nom l'indique, cela consiste à superviser l'apprentissage de la machine montrant des exemples (des données) de la tâche dans laquelle est réalisée. Les applications sont plusieurs : classifications, reconnaissance vocale, vision par ordinateur …etc. La grande majorité des problèmes de Machine Learning utilisent l'apprentissage supervisé. L'essentiel c'est de bien comprendre le fonctionnement de ce mécanisme **[MLIAA].**

2.2.1 Fonctionnement du ML supervisé

Avec l'apprentissage supervisé, la machine peut fonctionner certaine tâche pour étudier des exemples. Elle peut apprendre à reconnaître une photo de chien après qu'on lui est montré des millions de photos de chiens. Ou bien, elle peut reconnaître la traduction du français au chinois après avoir examiné des millions d'exemples sur la traduction **[AS]**.

2.2.2 Algorithmes d'apprentissage automatique supervisé

a Les k plus proches voisins

Les $\boldsymbol{k}$ plus proches voisins kPP ($\boldsymbol{k}$ *nearest neighbors – kNN*) appartient à la famille des algorithmes d'apprentissage supervisé automatique (*Machine Learning*). Nécessite d'avoir des données classifier pour qu'il pourrait classer une nouvelle donnée **[KPPV].**

- **Principe de l'Algorithme**

L'algorithme des k plus proches voisins n'exige pas de phase d'apprentissage à proprement dit, il faut stocker le jeu de données d'apprentissage. Soit un ensemble E contenant n données :

E= $\{(\vec{x}_i , y_i)\}$ avec i compris entre 1 et n, où y_i correspond à la classe de la donnée i, et le vecteur $\vec{x}_i$ de dimension P représente les variables prédicatrices de la donnée i **.** $\vec{x}_i$=

($\vec{x}_{1i}$, $\vec{x}_{2i}$, …, $\vec{x}_{pi}$). Soit une donnée u qui n'appartient pas à E et qui ne possède pas de classe (u est uniquement caractérisée par un vecteur $\vec{x}_u$ de dimension P). Une fonction d qui renvoie la distance entre la donnée u et une donnée quelconque appartenant à E**,** Soit un entier k inférieur ou égal à n **[KPPV].**

Voici le principe de l'algorithme des k plus proches voisins **[KPPV] :**

- À l'aide de la fonction d, On calcule les distances entre la donnée u et chaque donnée appartenant à **E**.
- On retient les $\boldsymbol{d}$ données du jeu de donnée **E** les plus proches de $\boldsymbol{u}$.
- On répartie à $\boldsymbol{u}$ la classe qui est la plus fréquente parmi les k données les plus proches.

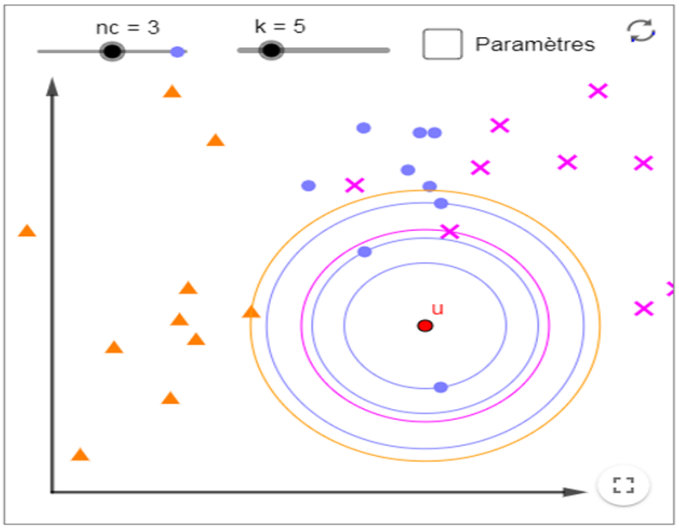


Figure 2.1 : Le principe de l'algorithme des k plus proches voisins.

- **Le paramètre *k***

Un objet est classé par un choix à la majorité de ses k voisins. k Est un entier positif pas très grand. Admettant que k = 1 l'objet est affecté à la classe de son voisin. Dans un contexte binaire où il n'y a que deux classes, il est admis de choisir k impair pour éviter de s'introduire à des situations de parité **[ACKPP].**

b Régression

- **Régression linéaire**

La régression linéaire est l'un des algorithmes d'apprentissage supervisé les plus populaires ainsi qu'un type d'analyse prédictive de base. Il est simple et mieux compris en statistique et en apprentissage automatique.

La forme la plus simple de l'équation de régression avec une variable dépendante et une variable indépendante est définie par la formule $\boldsymbol{y = c + b \times x}$, avec $\boldsymbol{y}$ = variable dépendante estimé, $\boldsymbol{c}$ = constante et $\boldsymbol{b}$ = coefficient de régression et enfin $\boldsymbol{x}$ = variable indépendante. On parle ici de régression linéaire simple. Pour la régression linéaire multiple on écrira $\boldsymbol{y = c + b \times x_1 + \ldots + n * x_n}$ avec $\boldsymbol{x_1}$ jusqu'à $\boldsymbol{x_n}$ les variables indépendantes et b jusqu'à n les coefficients de régression respectifs des variables **[LDAS].**

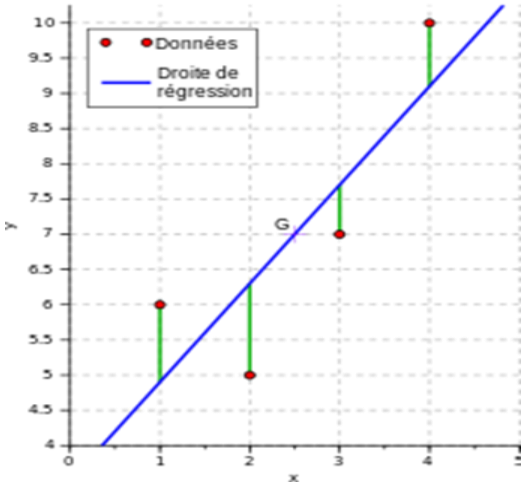


Figure 2.2 : Régression linéaire.

- **Régression logistique**

Les prévisions de régression linéaire sont des valeurs continues, par contre les prédictions de régression logistique sont des valeurs discrètes, à vrai dire un ensemble fini de valeurs (par exemple Vrai ou faux). La régression logistique convient à la classification binaire. Par exemple, on peut dire un ensemble de données où y = 0 ou 1 avec 1 représente la classe par défaut. Pour illustrer on peut imaginer que l'on veuille prédire s'il pleuvra ou non. On aura 1 pour s'il pleut et 0 le cas contraire **[LDAS].**

A l'opposé de la régression linéaire, la régression logistique propose le résultat sous forme de probabilités de la classe par défaut. Le résultat appartient donc à l'intervalle [0 :1], veut dire qu'il est compris entre 0 et 1 vu qu'il s'agit d'une probabilité. La valeur y de sortie est générée par la transformation de la valeur x, à l'aide de la fonction logistique :

$$\boldsymbol{h(x) = \frac{1}{1+e^{-x}}} \qquad \textbf{(17)}$$

Un seuil est appliqué pour forcer cette probabilité dans une classification binaire **[LDAS].**

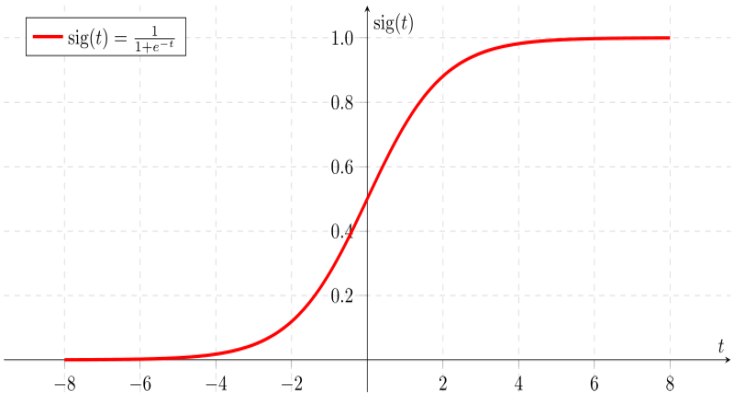


Figure 2.3 : Régression logistique.

c Arbres de décision

L'apprentissage se montre par partitionnement récursif suivant des règles sur les variables explicatives. Selon les données et les critères de partitionnement, on dispose de différentes méthodes, dont CART (Classification And Regression Tree), CHAID etc... Ces méthodes peuvent s'appliquer à une variable quantitative ou qualitative. Deux types d'arbres de décision sont définis :

- **Arbres de classification :** La variable expliquée est de type nominale (facteur).
- **Arbres de régression :** La variable expliquée est de type numérique

d Réseaux de neurone

- **Principe**

Le réseau de neurone repose sur la notion de neurone formel. Un neurone formel est un modèle caractérisé par des signaux d'entrée (les variables explicatives par exemple) et une fonction d'activation f (**Figure 2.4**). Possible que la fonction f est linéaire à seuil stochastique et le plus souvent c'est sigmoïde. Le calcul des paramètres se fait par apprentissage **[MUTRN].**

Les neurones sont ensuite unifiés en couche. Une couche d'entrée lit les signaux entrants x^j, c'est-à-dire un neurone par un signal entrée et une couche de sortie qui fournit la réponse du système.

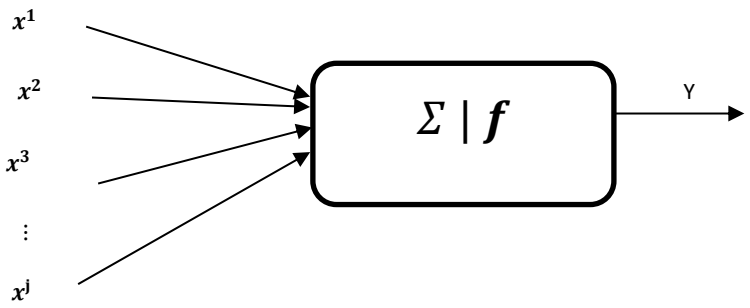


Figure 2.4 : Représentation d'un neurone formel.

Une ou plusieurs couches cachées collaborent au transfert. 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 (**Figure 2.5**). De façon habituelle et en régression (Y quantitative), la couche de sortie contient un seul neurone fourni de la fonction d'activation identité alors que les autres neurones (couche cachée) sont munis de la fonction d'activation sigmoïde. Par exemple En classification binaire, le neurone de sortie (couche sortie) est muni également de la fonction sigmoïde tandis que dans le cas d'une discrimination à m classes (Y qualitative), ce sont m neurones avec fonction sigmoïde, un par classe, qui sont considérés en sortie **[MUTRN].**

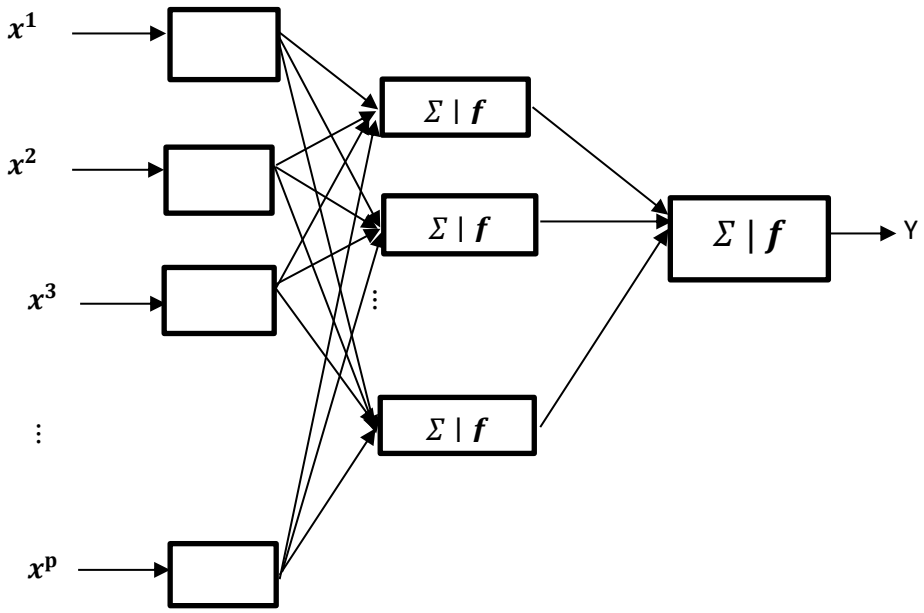


Figure 2.5 : Exemple de perceptron multicouche élémentaire avec une couche cachée et une couche de sortie.

- **Apprentissage (ajustement)**

On rétrograde une fonction objective $Q(\alpha)$ a partir des gradients de cette fonction, on utilise un algorithme d'optimisation **[MUTRN].**

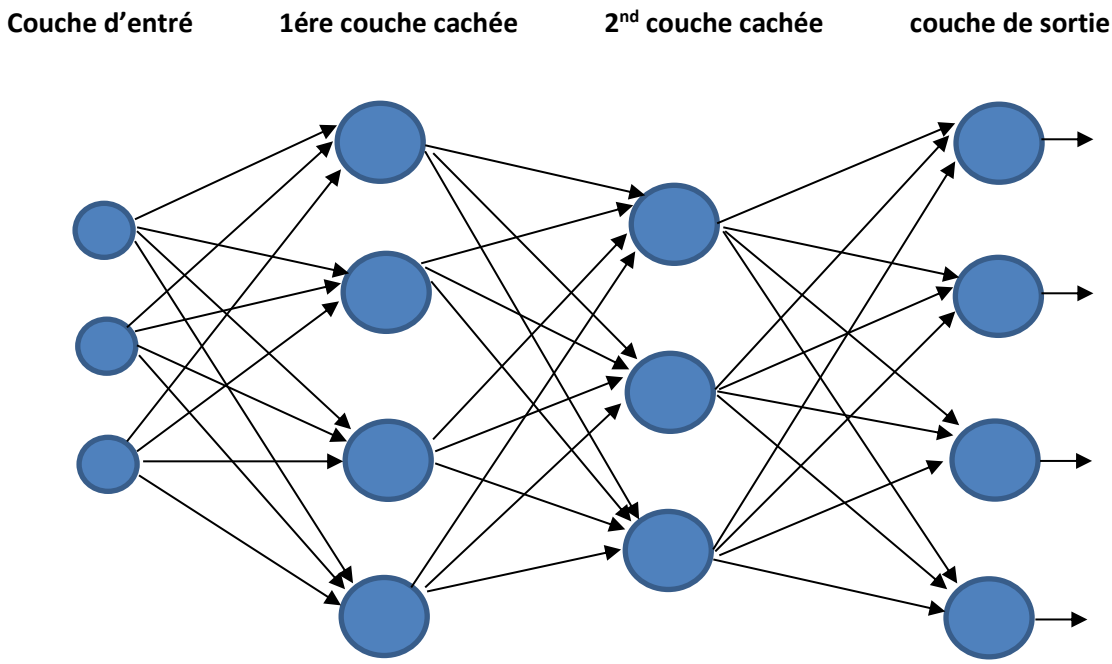


Figure 2.6 : Représentation les différents couches (entrés, sorties et cachées).

e *Machines à support vectoriel*

Ces machines, proposées par Vapnik ont été utilisées avec succès dans plusieurs tâches d'apprentissage et sont actuellement en plein essor. Elles donnent en particulier une bonne approximation du principe de minimisation du risque structurel. La méthode repose sur les idées suivantes **[CTCAS]** :

- Les données sont projetées dans un espace de grande dimension par une transformation basée sur un noyau linéaire, polynomial ou gaussien comme le démontre la **Figure 2.7**.
- Dans cet espace transformé, les classes sont séparées et distinguées par des classifieurs linéaires qui maximisent la marge (distance entre les classes).
- Les hyperplans peuvent être identifier au moyen d'un nombre de points limités qui seront nommés les « vecteurs supports ».

La complexité d'un classifieur SVM ne dépende pas de la dimension de l'espace des données, mais du nombre de vecteurs supports nécessaires pour réaliser la séparation **[CTCAS].**

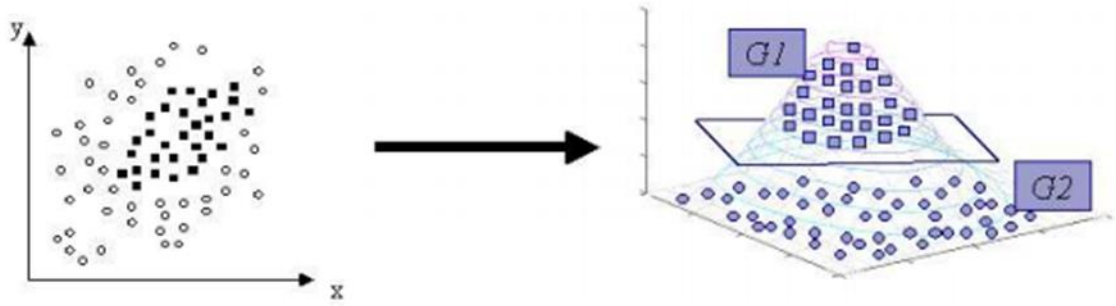


Figure 2.7 : plongeant les données dans un espace dimensionnel **[CTCAS].**

2.3 L'apprentissage automatique non-supervisé

Dans l'apprentissage non supervisé, le modèle n'est pas supervisé, il fonctionne seul pour prédire les résultats. Il utilise des algorithmes d'apprentissage automatique pour faire sortir des conclusions sur des données non étiquetées. En règle générale, les algorithmes d'apprentissage non supervisé sont plus difficiles que les algorithmes d'apprentissage supervisé où il y a peu d'informations. L'apprentissage non supervisé contient deux modèles d'algorithmes qui sont : regroupement et association **[DBSUS].**

2.3.1 Fonctionnement du ML non-supervisé

a Regroupement ou Clustering

La mise en cluster (Regroupement) consiste à séparer ou bien à diviser un ensemble de données en un certain nombre de groupes, le but est de séparer les groupes ayant des traits identiques et les assigner en grappes. Voyons cela avec un exemple, supposons que vous soyez le chef d'un magasin et que vous espérez de comprendre les préférences de vos clients pour développer votre activité. Vous pouvez regrouper tous les clients en 10 groupes tout dépend de leurs habitudes d'achat et d'utiliser une stratégie distincte pour les clients de chacun de ces 10 groupes. Ce que veut dire le Clustering **[ASVNS].**

b Association

L'association consiste à établir des relations intéressantes entre des variables dans une grande base de données. Par exemple, les personnes qui achètent une nouvelle demeure ont aussi une attirance à acheter de nouveaux meubles. Elle détecte la probabilité de co-occurrence d'éléments dans une collection **[ASVNS].**

c Résumé

Le clustering groupe des points de données en fonction de leurs similitudes, alors que l'association consiste à découvrir des relations entre les attributs de ces points de données **[ASVNS].**

2.3.2 Algorithmes d'apprentissage automatique non supervisés

- Algorithme k-means.
- Méthode hiérarchique.

2.4 Conclusion

Nous avons introduit dans ce chapitre la définition de l'apprentissage automatique (Machine Learning) et ces différents types : supervisé et non-supervisé avec leurs fonctionnalités et leurs principaux algorithmes. En résumé, on peut conclure que :

- Le ML supervisé est la tâche d'apprentissage automatique qui définit une fonction associe une entrée à une sortie en fonction d'exemples de couple entrée-sortie, tandis que le ML non-supervisé est la tâche d'apprentissage automatique qui consiste à raisonner une fonction décrivant la structure masquée à partir de données non étiquetées.
- Le ML supervisé utilise les algorithmes de régression et de classification, alors que le ML non-supervisé fonctionne avec des algorithmes de la mise en cluster.
- Le ML supervisé utilise des données étiquetées et les résultats générés par ces méthodes sont plus précis et fiables, tandis que le ML non-supervisé utilise des données non étiquetées dont leurs résultats ne sont pas très précis et qualifiables.

Après avoir étudié les différents types d'apprentissage, nous nous somme intéressés à l'apprentissage automatique supervisé parce que notre travail consiste à utiliser un ensemble d'exemples d'apprentissage (les données) étiquetés. La section suivante sera concentrée à la méthode de classification par les réseaux neuronaux. Plus précisément, les réseaux de neurones convolutifs (CNN) qui sont développée par Yann LeCun (1989) en théorie d'apprentissage automatique supervisé.

Chapitre 3 Les réseaux de neurones convolutifs CNN

3.1 Introduction

L'apprentissage profond est devenu de plus en plus apprécier dans les domaines académiques et industriels au cours de ces dernières décennies. Plusieurs domaines, tels que la vision par ordinateur, reconnaissance de formes et le traitement du langage naturel ont été témoin de la grande puissance de réseaux profonds. Cependant, les études actuelles sur l'apprentissage profond concentrent principalement sur les ensembles de données avec les étiquettes de classe équilibrées.

La classification d'images est un projet d'apprentissage profond. Plus précisément, elle relève de la catégorie des projets de vision par ordinateur, elle est aussi le cas d'utilisation le plus apprécier dans l'analyse d'images numériques. Le problème de classification est de catégoriser tous les pixels d'une image numérique dans l'une des classes définies.

Dans la classification supervisée, nous sélectionnons des échantillons pour chaque classe ciblée. Nous formons notre réseau neuronal sur ces échantillons de classes ciblées, puis classifions de nouveaux échantillons **[ICDLPPK].**

3.2 Convolutional Neural Network (CNN)

Les réseaux convolutifs également connus sous le nom Convolutional Neural Network (LeCun, 1989) est une classe de réseaux neuronaux d'apprentissage en profondeur. Les CNN représentent un grand avancement dans la reconnaissance d'images. Ils sont le plus

souvent utilisés pour analyser les images visuelles et travaillent fréquemment dans les coulisses de la classification des images **[TCBGDL].**

3.2.1 Le fonctionnement d'un CNN

Les réseaux de neurones convolutifs sont les modèles les plus compétitifs à ce jour pour classer des images. Désignés par l'acronyme CNN, de l'anglais Convolutional Neural Network, ils comportent deux parties bien distinguées. En entrée, une image avec des valeurs numériques sous la forme d'une matrice de pixels, disposée spatialement : la largeur, la hauteur et la profondeur (dimensions) **Figure 3.1**. Contient 2 dimensions(2D) pour une image en niveaux de gris. Une couleur est représentée par une troisième dimension, profondeur (3D) pour représenter les couleurs fondamentales [Rouge, Vert, Bleu] RVB.

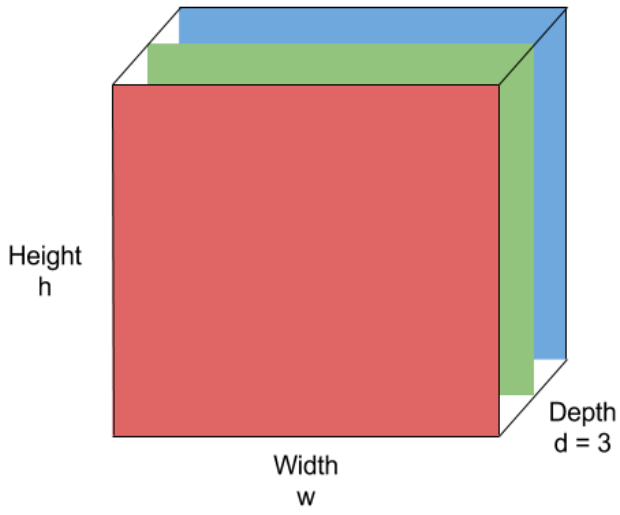


Figure 3.1 : Image disposé spatialement : la largeur, la hauteur et la profondeur **[CNNPP].**

La partie convolutive est La première partie de réseaux de neurone convolutive CNN, elle Fonctionne comme un extracteur de caractéristiques des images. Représenter comme une matrice qui va passer à travers un enchaînement de filtres : couche de convolution et couche de Pooling (regroupement), inventant de nouvelles matrices (images) appelées cartes de convolutions. A la fin, les cartes de convolutions sont concaténées en un vecteur de caractéristiques, nommé code CNN.

Ce code CNN est ensuite branché en entrée d'une deuxième partie qui nommée Fully-Connected layers (couches entièrement connectées). Le rôle principal de cette dernière est de combiner les caractéristiques du code CNN pour classer une image.

Un CNN **(Figure 3.2)** est composé de :

1.La première partie (convolutive)

- Un ou plusieurs blocs de Convolutional layers.
- Un ou plusieurs blocs de pooling layers.

2.Deuxième Partie (Fully-Connected layers)

- Une ou plusieurs Fully-Connected layers
- Une Output layer.

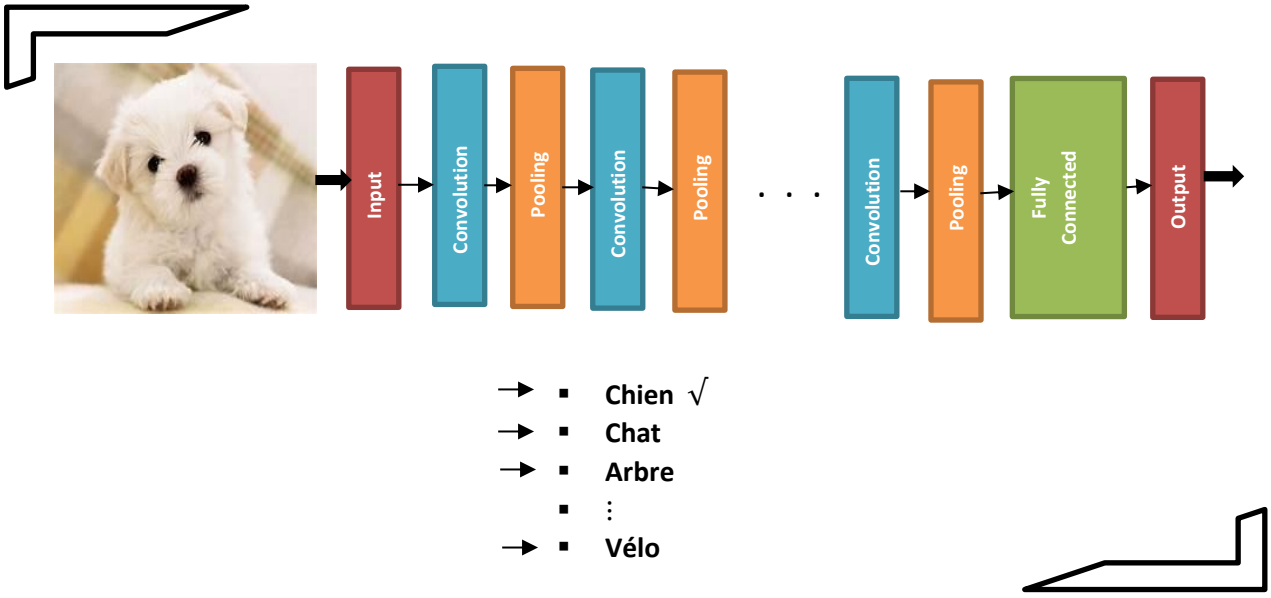


Figure 3.2 : principe de Fonctionnement de CNN.

3.2.2 Convolutional layer

Le but principal de l'étape de convolution est d'extraire les caractéristiques de l'image d'entrée. La couche convolutive est toujours la première étape d'un CNN. La convolution préserve la relation entre les pixels en apprenant les caractéristiques de l'image à l'aide

de petits carrés de données d'entrée. C'est une opération mathématique qui prend deux entrées **[TCBGDL].**

- Matrice d'image de dimension $(h \times w \times d)$.
- Un filtre de dimension $(f_h \times f_w \times d)$.

Disposant une image d'entrée, un détecteur de caractéristiques et d'une carte des caractéristiques. Prenons le filtre et appliquons le bloc de pixels par un autre bloc de pixels à l'image d'entrée. Cela fait par la multiplication des matrices **[CNNPP]**.

- Dans le cas ou d≠0 : $hauteur = h - f_h + 1 \quad , \quad largeur = w - f_w + 1$

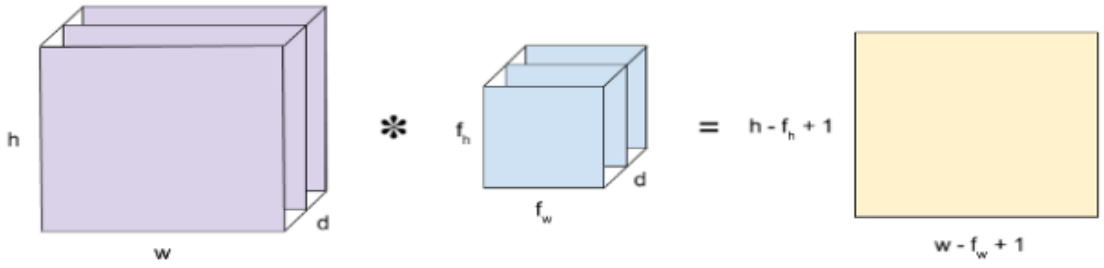


Figure 3.3 : Les dimensions de la matrice finale dans le cas ou $d \neq 0$ **[CNNPP].**

- Dans le cas ou d=0 : $hauteur = n - f + 1 \,, \quad largeur = n - f + 1$

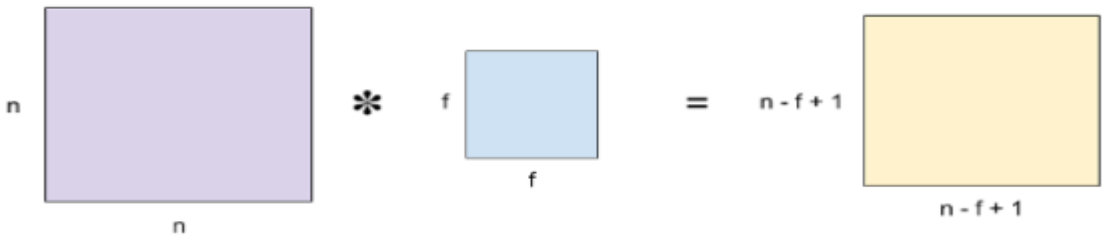


Figure 3.4 : Les dimensions de la matrice finale dans le cas ou $d = 0$ **[CNNPP].**

a Stride

On suppose que nous avons une lampe de poche et une feuille de papier bulle. La lampe de poche fait briller une zone de 3 bulles x 3 bulles. Pour observer la feuille entière, il faut glisser la lampe de poche sur chaque carré de (3 x 3) jusqu'à ce que nous

voyions toutes les bulles. La lumière de la lampe de poche ici c'est notre filtre et la région sur laquelle nous glissons est le champ récepteur. La lumière qui glisse à travers les champs réceptifs fait tourner la lampe de poche. Le filtre est un tableau de nombres (également appelés poids ou paramètres). La distance à laquelle la lumière de la lampe de poche glisse lorsqu'elle se déplace s'appelle la stride **[TCBGDL]**.

Par exemple, on imagine un volume d'entrée (7 x 7) et un filtre (3 x 3).

Quand on a une stride de 1 (stride = 1), cela veut dire que nous déplaçons notre filtre (3 x 3) sur 1 pixel à la fois **(Figure 3.5. A) [GUCNN]** :

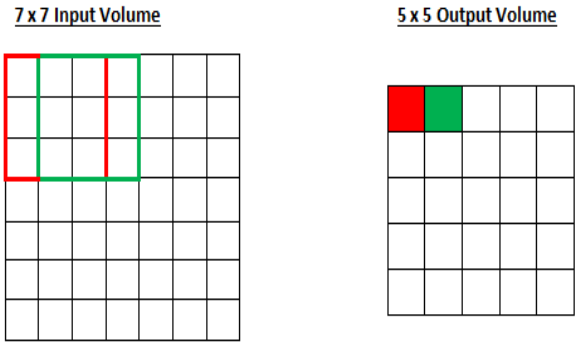


Figure 3.5. A : exemple de stride d'un pixel **[GUCNN].**

Et quand on a une stride de 2 (stride = 2) cela signifie que nous déplaçons notre filtre de (3 x 3) sur deux pixels à la fois (**Figure 3.5.B**), La convention est un pas de deux pixels.

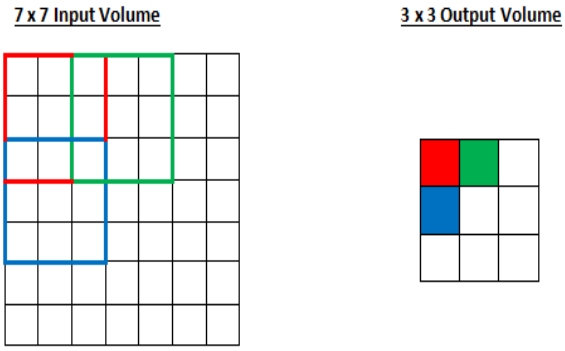


Figure 3.5.B : exemple de stride de 2 pixels **[GUCNN].**

b Padding

Observons maintenant le rembourrage (padding). Nous allons découvrir ce qui va arriver quand on applique un trois filtres (5 x 5 x 3) à un volume d'entrée (32 x 32 x 1). Le volume de sortie serait de (28 x 28 x 3). Notons que les dimensions spatiales (longueur, largeur) diminuent à chaque fois que nous appliquons des couches de convolution (**Figure 3.6.A**), la taille du volume diminuera plus rapidement que nous le souhaitons. Dans les premières couches de notre réseau, nous souhaitons conserver autant d'informations sur le volume d'entrée d'origine afin de pouvoir extraire ces fonctionnalités de bas niveau. Disons que nous voulons appliquer la même couche de convolution mais que le volume de sortie reste (32 x 32 x 3). Alors, nous appliquons un remplissage nul de taille deux pixels à ce calque. Zéro padding remplit le volume d'entrée avec des zéros autour de la bordure. Si nous pensons à un remplissage nul de deux, cela se traduit par un volume d'entrée de (36 x 36 x 3) comme montré sur la (**Figure 3.6. B**) **[IECNN] [TCBGDL]** :

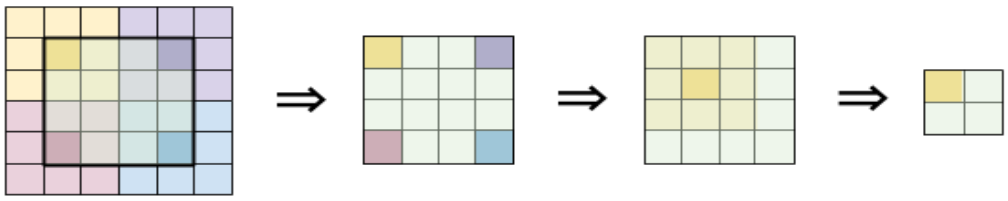


Figure 3.6.A : diminutions de volume du la matrice de sortie après chaque convolution **[IECNN].**

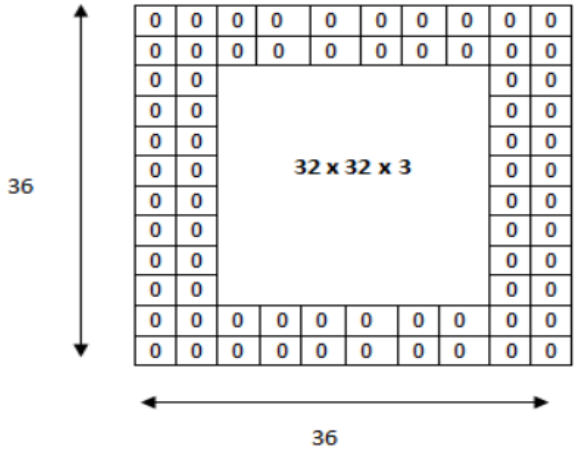


Figure 3.6.B : Exemple sur Zero padding avec taille 2(pixel) **[GUCNN].**

3.2.3 Pooling (regroupement)

La section des couches de pooling (Regroupement) réduit le nombre de paramètres lorsque les images sont trop larges ou grandes. Le pooling (Regroupement) spatial est également appelé sous-échantillonnage, il réduit la dimensionnalité de chaque carte de caractéristiques (Feature map) mais conserve les informations importantes. Il y a différents types de pooling spatiale : Pooling (Regroupement) maximum, Pooling (Regroupement) moyenne et Pooling (Regroupement) de somme.

Nous nous concentrons sur le pooling (Regroupement) maximale **(Figure 3.7)**, nous prenons une certaine partie de la carte des caractéristiques (future map) obtenue à l'étape précédente et prenons la valeur la plus élevée dans cette partie. Ce faisant, conserver la caractéristique de l'image mais en même temps nous nous débarrassons des informations qui non aucune importance. La carte que nous obtenons en conséquence s'appelle la carte des caractéristiques groupées **[IECNN] [CNNIQ].**

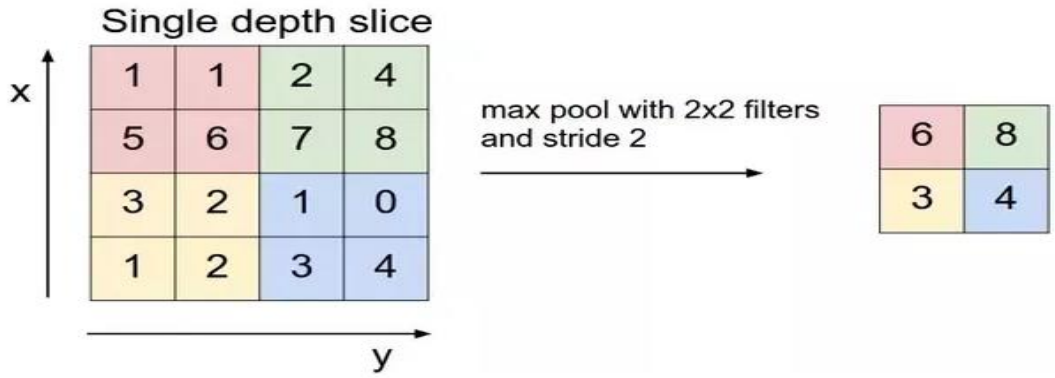


Figure 3.7: Max Pooling **[IECNN].**

3.2.4 Couches de correction (ReLU)

C'est une fonction qui met les valeurs des neurones qui sont négatives au 0 et garde les valeurs positives comme ils sont, elle est donnée par : F(x)=max (0, x).

Il existe encore plusieurs fonctions d'activation comme tanh, sigmoïde….

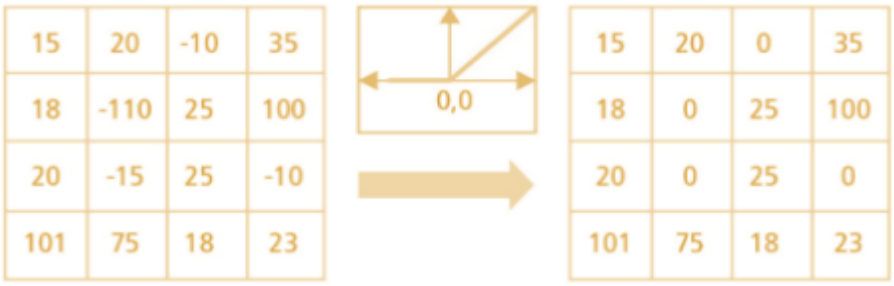


Figure 3.8 : La fonction de correction ReLU.

3.2.5 Flattening (Aplatissement)

Une fois que l'image est convolue et max regroupés, la procédure de Flattening (Aplatissement) consiste à tenir essentiellement les éléments dans une carte d'entités groupés et les transformés sous forme vectorielle (Code CNN). Cela devient la couche d'entrée pour la couche entièrement connectée **[CNNIQ].**

3.2.6 La couche entièrement connectée (Fully-Connected layers)

Après l'aplatissement, le résultat est un vecteur qui agit comme la couche d'entrée pour une deuxième partie de CNN nommée Fully-Connected layers. Elle fonctionne normalement pour détecter l'image. Elle dispose des poids aléatoires à chaque synapse, la couche d'entrée est ajustée en poids et placée dans une fonction d'activation. La sortie de ceci est comparée aux valeurs vraies et l'erreur générée est rétro-propagée. C'est-à-dire que les poids sont réajustés et tous les processus sont répétés, cela se fait jusqu'à ce que l'erreur ou la fonction de coût soit minimisée **[IECNN]**.

Notre Fully-Connected layers nommée ANN. On conclue qu'après la phase d'aplatissement, lorsque l'image est convertie en vecteur, toutes les étapes suivantes ressemblent à celles des ANN.

3.2.7 ANN (Artificiel Neural Networks)

a Introduction sur le ANN

Les réseaux de neurones artificiels (ANN) sont des algorithmes de calcul. Ils fonctionnent de la même façon que les réseaux de neurones biologiques, comme le cerveau humain qui traite les informations. Ils saisissent une large unité de traitement connectée qui travaillent ensembles pour traiter les informations. Ils génèrent des résultats significatifs **[ANN].**

Les réseaux de neurones artificiels visent à simuler le comportement du système biologique composé de « neurones ». Les ANN sont des modèles informatiques inspirés du système nerveux central d'un animal. Il s'agit d'un outil d'apprentissage automatique ainsi que de reconnaissance de formes. Ceux-ci sont présentés comme des systèmes de « neurones » interconnectés qui peuvent calculer des valeurs à partir d'entrées **[ANN]**.

b Applications des ANN

- Classification : le but de cette application est de prédire la classe d'un vecteur d'entrée.
- Pattern Matching : son objective est de produire un pattern mieux associé à un vecteur d'entrée donné.
- Complétion de motif : la cible de dernier est de compléter les parties manquantes d'un vecteur d'entrée.
- L'optimisation : elle se fait pour trouver les valeurs optimales des paramètres dans un problème d'optimisation.
- Contrôle : est une action appropriée est suggérée en fonction d'une entrée donnée vecteurs.
- Approximation de fonction / modélisation de séries temporelles : le but de cette application est d'apprendre la relation fonctionnelle entre les vecteurs d'entrée et de sortie souhaités.
- Exploration de données : elle dévoile des modèles cachés à partir des données (découverte de connaissances) **[ANNCK].**

c Architectures ANN

ANN est généralement organisé en couches. Ils sont constitués de nombreux nœuds interconnectés qui contiennent une fonction d'activation. Un réseau neuronal peut comporter les trois couches suivantes **[ANNML] [ANNCK] :**

- Couche d'entrée : nombre de neurones dans cette couche correspond au nombre d'entrées vers le réseau neuronal. Cette couche se compose de nœuds passifs, c'est-à-dire qui ne participent pas aux réelles modifications du signal, mais ne transmet que le signal à la couche suivante.
- Couche cachée : cette couche contient un ou plusieurs nombres de couches avec un nombre arbitraire des neurones. Les nœuds de cette couche prennent une partie de la modification du signal, par conséquent ils sont actifs.
- Couche de sortie : le nombre de neurones dans la couche de sortie correspond au nombre de valeurs de sortie du réseau neuronal. Les nœuds de cette couche sont actifs.

d Fonction d'activation

La fonction d'activation est une opération mathématique qui converti le résultat (les signaux des neurones) en une sortie normalisée. Son objectif en tant que composant d'un réseau neuronal est d'inséré la non-linéarité au sein du réseau. L'intégration d'une fonction d'activation permet au réseau neuronal d'avoir un grand pouvoir de représentation et de résoudre des fonctions complexes.il y'a plusieurs types de fonction d'activation **[RAIAN] :**

- **Fonction Softmax**

Elle est utilisée pour dériver la distribution de probabilité d'un ensemble de nombres dans un vecteur d'entrée. La sortie d'une fonction d'activation Softmax est un vecteur dans lequel ses ensembles de valeurs représente la probabilité d'occurrence d'une classe ou d'un événement. Les valeurs dans le vecteur s'additionnent toutes à 1 **[RAIAN].**

- **Fonction sigmoïde**

  L'équation de la fonction est :

$$f(x) = \frac{1}{1+e^{-z}} \quad \textbf{(18)}$$

C'est une fonction non linéaire, la valeur est entre 0 et 1(**Figure 3.9**).

La fonction sigmoïde est particulièrement utilisée pour les modèles où elle devait prédire la probabilité en tant que sortie, car la probabilité de qu'elle que soit n'existe qu'entre 0 et 1, le sigmoïde est le bon choix **[AFINN] [AFFNN].**

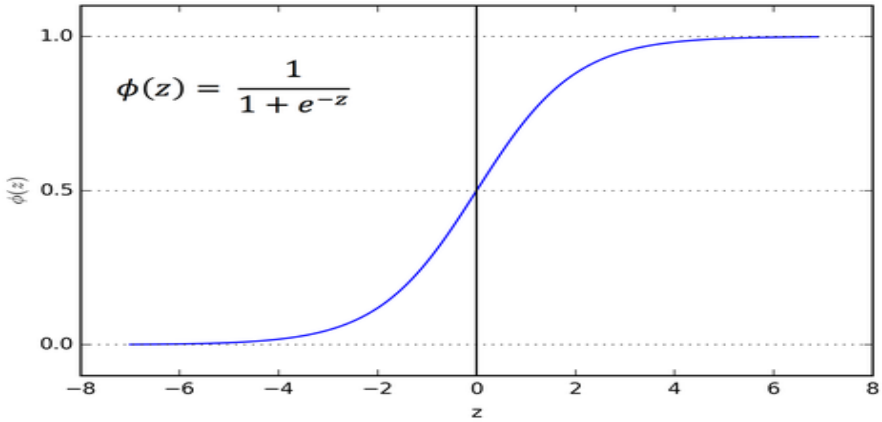


Figure 3.9 : fonction sigmoïde **[AFINN].**

- **Fonction Tanh**

  L'équation de la fonction est :

$$f(x) = \left(\frac{2e^{x}}{e^{x}+e^{-x}}\right) - 1 \quad \textbf{(19)}$$

La fonction tanh est pareil avec la fonction sigmoïde logistique mais mieux. La valeur est entre 1 et -1 (**Figure 3.10**). Après la différenciation, la valeur de cette fonction devient inférieure à 1.

La relation entre tanh $f(x)$et sigmoïde $g(x)$est :

$$f(x) = 2g(2x) - 1 \quad \textbf{(20)}$$

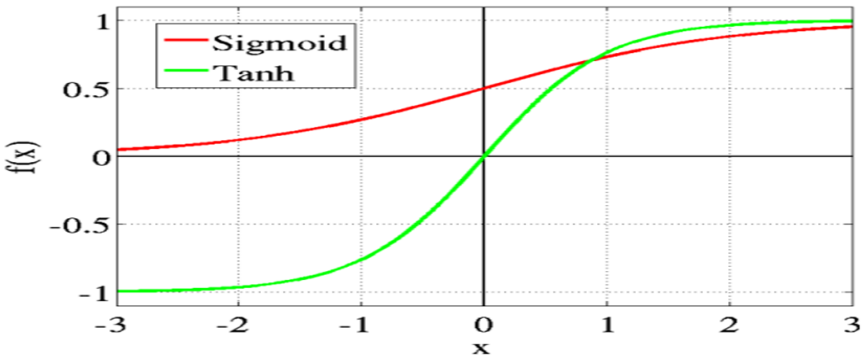


Figure 3.10: tanh v/s Logistic Sigmoid **[AFINN].**

L'avantage de cette fonction est que les entrées négatives seront mappées fortement négatives et les entrées nulles seront mappées près de zéro dans le graphique de tanh **[AFINN] [AFFNN].**

- **Fonction d'activation ReLU**

  Le fonction Relu (Rectified Linear Unit) est mondialisé dans l'utilisation à l'heure actuelle, elle est utilisée presque dans tous les réseaux de neurones convolutifs (couche caché), L'équation de la fonction est (**Figure 3.11**) :

$$\boldsymbol{f(x) = \max(0, x)} \qquad \textbf{(21)}$$

  La plage de valeur de cette fonction est : (0, inf) **[AFINN] [AFFNN].**

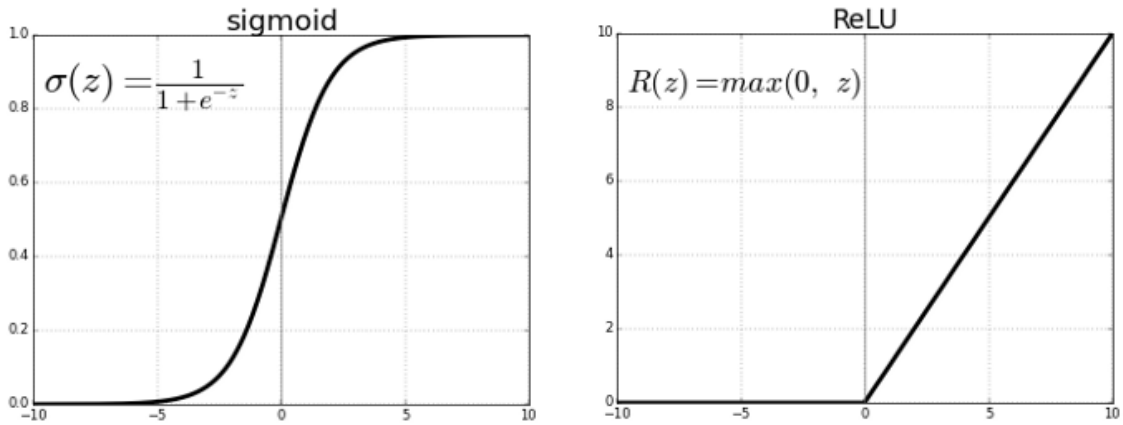


Figure 3.11: ReLU v/s Logistic Sigmoid **[AFINN].**

Comme on observe dans la **Figure 3.11** le ReLU est à moitié rectifié (par le bas).

- R (z) égale zéro : $(R(z) = 0)$ lorsque z est inférieur à zéro.

- R (z) est égal à z : $(R(z) = z)$ lorsque z est supérieur ou égal à zéro **[AFINN] [AFFNN].**

e Fonctionnement de ANN

ANN réalise une opération mathématique entre la sortie de la couche précédente et les poids de la couche actuelle, puis il transmet les données à la couche suivante en passant par la fonction d'activation. La **Figure 3.12** montre une unité d'un nœud **[WITDCNN].**

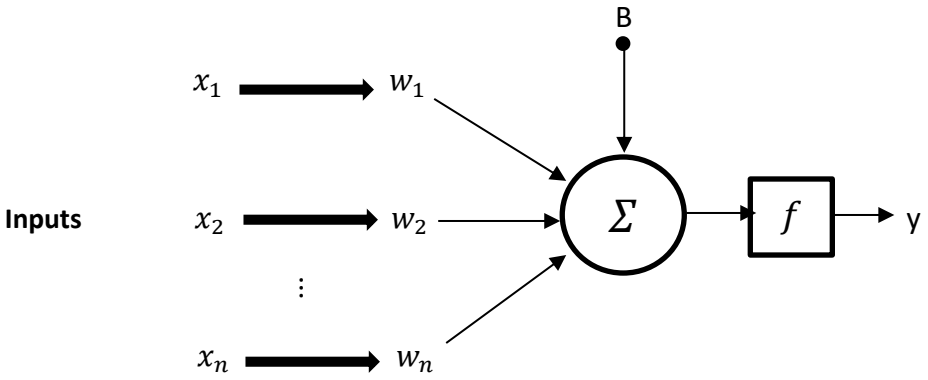


Figure 3.12 : opération mathématique entre la sortie de la couche précédente et les poids de la couche actuelle **[WITDCNN].**

f Les types de réseaux de neurones artificiels

Dans le réseau de neurones artificiels Il existe plusieurs types. Ils sont mis en œuvre sur la base des opérations mathématiques et d'un ensemble de paramètres essentielle pour déterminer la sortie. C'est la partie la plus critique de la mise en œuvre d'un réseau neuronal. Examinons certains des réseaux de neurones **[6TANN] :**

- **FeedforwardNetworks**

  Ce réseau de neurones est l'une des formes les plus simples d'ANN, Les données ou l'entrée circulent dans une direction. Les données passent par les nœuds d'entrée et sortent sur les nœuds de sortie. Ce réseau neuronal peut ou non avoir les couches cachées. Ça veut dire, il a une onde propagée par front et aucune rétropropagation (back propagation), en utilisant habituellement une fonction d'activation de classification **[6TANN].**

- **Feedback networks (rétroaction)**

  Dans ce type d'ANN, la sortie d'un neurone retourne dans le réseau pour décrocher les meilleurs résultats en interne. Le réseau de rétroaction (Feedback) renvoie des informations en lui-même (directement ou indirectement) et est bien concordé pour résoudre les problèmes d'optimisation, et aussi utilisés dans des tâches complexes de reconnaissance de formes, par exemple, la reconnaissance de parole etc…. **[ANNATT]**.

- **Lateral Networks (Réseaux latéraux)**

  Dans ce Réseau de neurone, Il existe des couplages de neurones au sein d'une même couche (voir **Figure 3.13**). Ainsi, il n'y a pas un chemin de rétroaction (Feedback) essentiellement bien exprimer entre les différentes couches. Cela peut être considéré comme un arrangement entre Feedforward et le Feedback **[ANNCK]**.

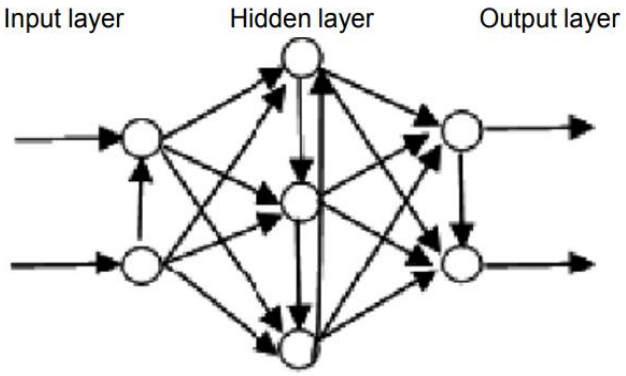


Figure 3.13 : Couplages de neurones dans les Réseau latéraux (Lateral Networks).

g Opération de Back-propagation(rétropropagation)

La rétropropagation (Back-propagation) est une forme abrégée de « propagation vers l'arrière des erreurs », C'est une méthode standard de formation des réseaux de neurones artificiels, elle permet de calculer la pente d'une fonction de perte par rapport à tous les poids du réseau. Le vecteur d'entrée et de sortie associé préparés à être entraînés par un algorithme de rétropropagation afin d'avoir la fonction qui donne la meilleure description de la relation entre les entrées et les sorties **[BPNN] [ANNBDL]**.

On explique le fonctionnement de la rétropropagation par quelques étapes, on considère le diagramme suivant (**Figure 3.14**) **[BPNN] :**

1.Entrées X, arrivent par le chemin pré connecté.

2.L'entrée est modélisée en utilisant des poids réels W.

3.Calcule de la sortie pour chaque neurone de la couche d'entrée, aux couches cachées, à la couche de sortie.

4.Calcule de l'erreur dans les sorties, avec :

$$Erreur_b = Actuel\ output - desired\ output \quad \textbf{(22)}$$

5. Revenir de la couche de sortie à la couche cachée pour combiner les pondérations pour réduire l'erreur.

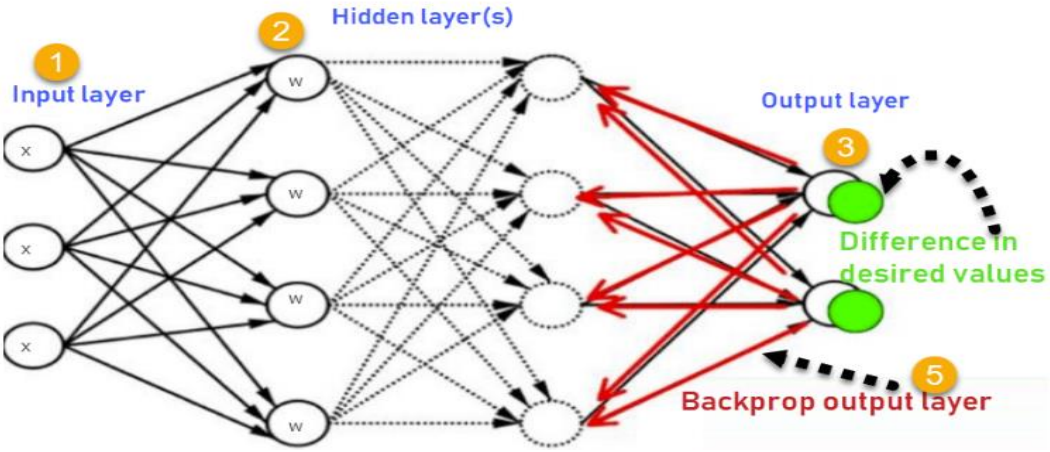


Figure 3.14 : fonctionnement d'algorithme du la rétropropagation **[BPNN].**

h Les Avantages et inconvénients

- Un réseau neuronal peut accomplir des tâches dans lesquelles un programme linéaire ne peut pas effectuer **[ANNATT].**
- Lorsqu'un élément du réseau neuronal tombe en panne, il peut continuer sans problème à cause de sa nature parallèle **[ANNATT].**
- Les réseaux de neurones fonctionnent même si une ou quelques unités ne réagit pas au réseau, mais pour mettre en œuvre des réseaux de neurones logiciels

volumineux et efficaces, une grande partie des ressources de traitement et de stockage doit être engagée. Alors que le cerveau dispose d'un matériel adapté à la tâche de traitement des signaux via un graphique de neurones, la simulation même d'une forme la plus simplifiée sur la technologie Von Neumann peut forcer un concepteur de réseau neuronal à insérer des millions de lignes de base de données pour ses connexions, ce qui peut consommer de grandes quantités de mémoire de l'ordinateur et espace disque dur **[ANN].**

3.3 La relation entre la partie convolutive et la partie entièrement connectée

On donne un exemple de visage humain, la couche convolutive pourrait être capable d'identifier des caractéristiques telles que le nez, les oreilles, les yeux etc… par contre, elle ne connait ni la position de ces caractéristiques ou ils devraient être. Avec les couches entièrement connectées (Fully-Connected layers) **(Figure 3.15)**, on a combiné ces fonctionnalités pour créer un modèle plus complexe qui pourrait donner au réseau plus de puissance de prédiction quant à l'emplacement de ces fonctionnalités afin de le classer comme humain. Ainsi, la différence fondamentale entre un CNN et un ANN n'est que l'étape de prétraitement **[TCBGDL].**

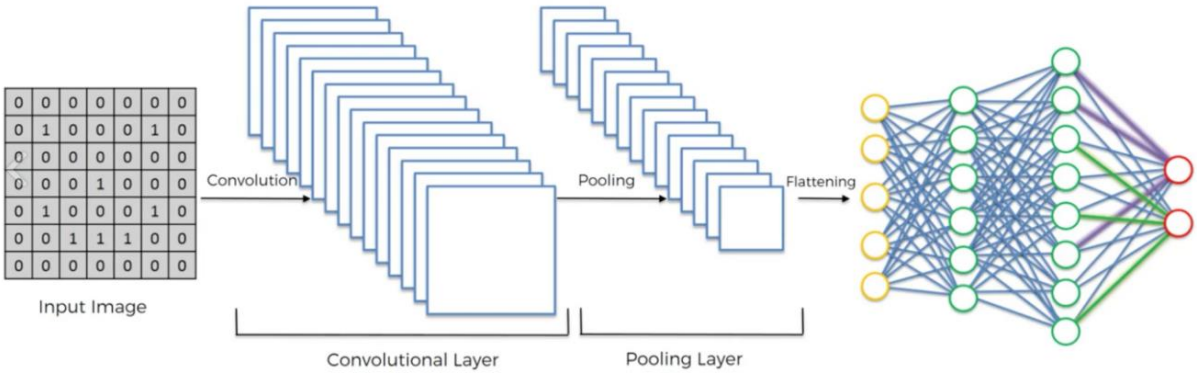


Figure 3.15 : La différence fondamentale entre la couche convolutive et ANN **[CNNIQ].**

3.4 Architecteur de CNN

Yann LeCun, Leon Bottou, Yosuha Bengio et Patrick Haffner ont suggéré une architecture de réseau neuronal pour la reconnaissance de caractères manuscrits et imprimés par machine dans les années 1990, qu'ils ont nommé LeNet-5. L'architecture

est facile à comprendre parce qu'elle est principalement utilisée comme première étape de l'enseignement **[MRCCNNA].**

On introduit ces différents types d'architecture CNN comme de suite :

3.4.1 LeNet-5 (1998)

LeNet-5 est le premier réseau neuronal convolutif (1998), il est considéré comme l'une des architectures les plus faciles. Lenet-5 est formé de **[RKICNNA] [SHAR]** :

- 32×32 inputs image
- Six 28× 28 feature maps convolutional layer (5×5 size)
- Average Pooling layers (2×2 size)
- Sixteen 10×10 feature maps convolutional layer (5×5 size)
- Average Pooling layers (2×2 size)
- Fully connected to 120 neurons
- Fully connected to 84 neurons
- Fully connected to 10 outputs

Autrement dit, il y a 2 couches convolutives et 3 couches entièrement connectées (Fully connected). La couche de Pooling moyenne s'appelle couche de sous-échantillonnage (subsampling) **Figure 3.16 [RKICNNA].**

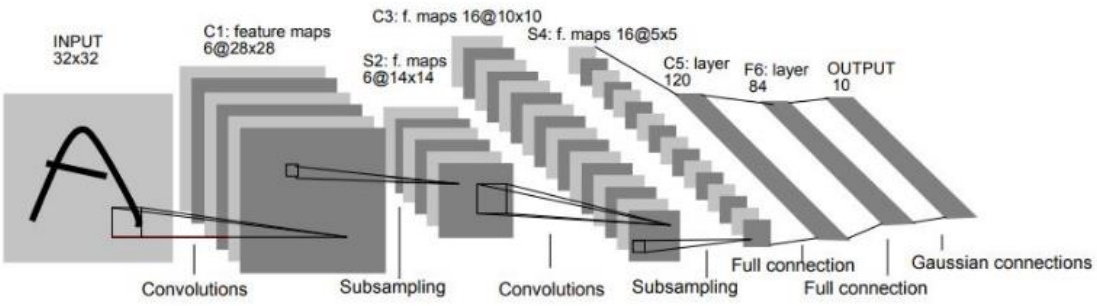


Figure 3.16 : Architecture simple de Lenet-5(1998) **[MRCCNNA]**

3.4.2 AlexNet (2012) :

AlexNet contient **[ICCNN] :**

- 8 couches : 5 couches convolutives et 3 couches d'entièrement connectées (Fully connected).

- Normalisation de la réponse locale.
- Empiler quelques couches supplémentaires sur LeNet-5
- Il est plus grand et plus profond que LeNet-5

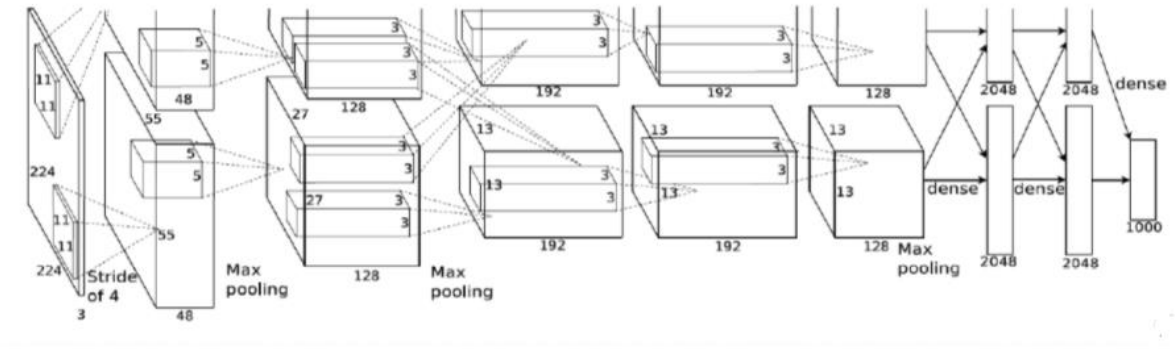


Figure 3.17 : Classification ImageNet avec AlexNet **[ICCNN].**

3.4.3 ZFNet (2013)

Le ZFNet **[ICCNN] :**

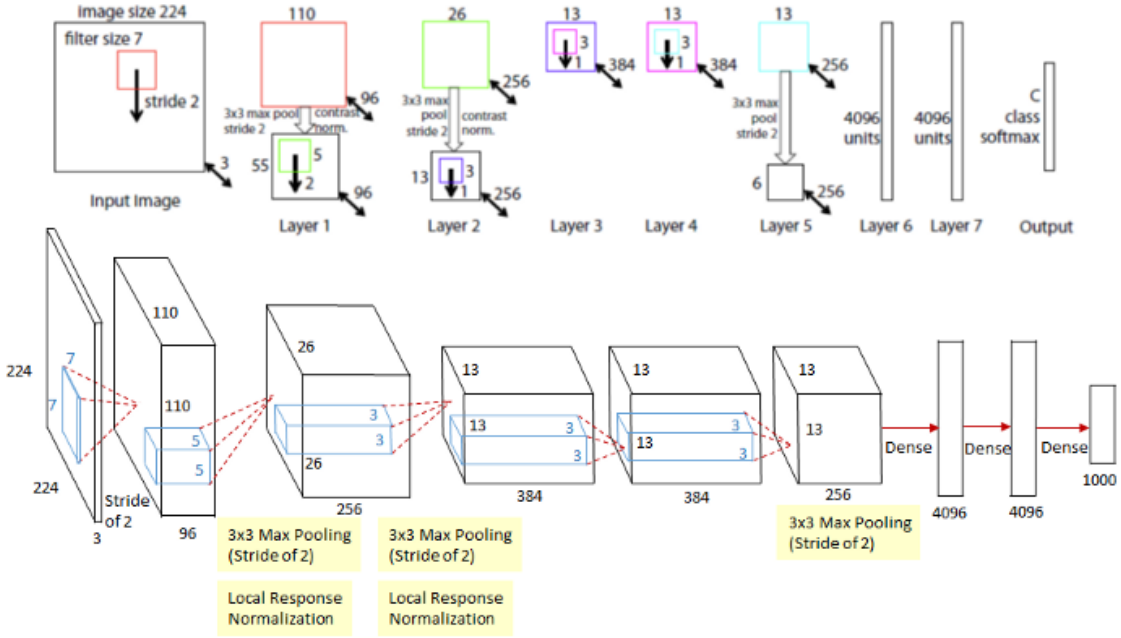


Figure 3.18 : Image illustre l'architecture de ZFNET **[SHTRIC].**

- Configuration presque similaire à AlexNet
- Petit filtre et foulé.
- Technique de visualisation – Deconvonet.

- Fournit une méthode de la matrice d'activation dans les couches supérieures vers l'entrée.
- CNN commandé à l'inverse.
- Détermine ce que chaque couche apprend.
- Donne un aperçu de la sélection de modèles.

3.4.4 VGGNet (2015)

Cela veut dire “Visual Geometry Group”, cette architecture (**Figure 3.19)** contient **[ICCNN] [RKICNNA]**:

- 13 couches de convolution (Convolutional layer) et 4 couches pooly.
- 3 couches entièrement connectées (Fully connected).
- Utilise des filtres de plus petite taille (2 × 2 et 3 × 3).
- Moins de temps de formation (training) comme : Réseau plus profond avec une petite taille de filtre, Pré-initialisation de certaines couches.
- Augmentation des données : Retournement horizontal, Décalage de couleur RVB, Scintillement de l'échelle.

Figure 3.19 : Image illustre l’architecture de VGGNET **[ICCNN].**

3.4.5 GOOGLENET (2014)

- Cette architecture (**Figure 3.20**) a été le gagnant du défi de classification d'images ILSVRC 2014.
- Plus profond et plus compliqué.
- Module de lancement avec réduction de dimensionnalité.
- Couches de pooling moyennes au lieu de la couche entièrement connectée (Fully connected).
- Est constitué de 22 couches profondes **[CNNA]**.

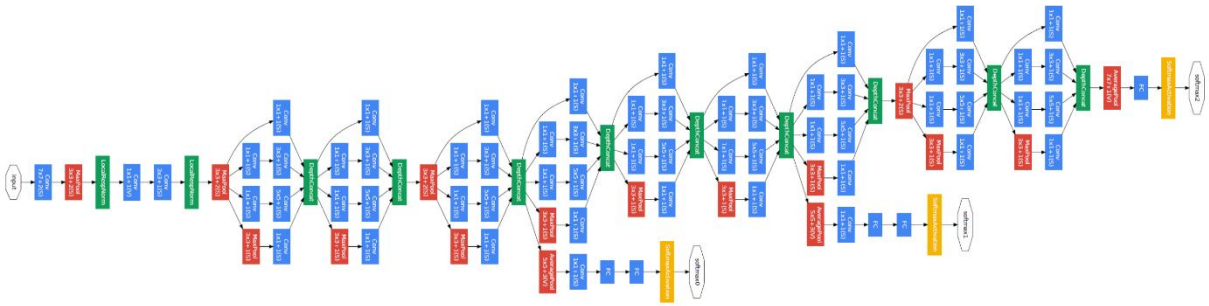


Figure 3.20 : Image illustre l'architecture de GoogleNet **[CNNA].**

3.4.6 ResNet (2015)

- Appelé « Residual Neural Network » (**Figure 3.21**).
- Le gagnant dans le défi ILSRVC 2015.
- Des réseaux plus profonds.
- Elle copie directement la matrice d'entrée dans la deuxième sortie de transformation et additionne la sortie dans la fonction ReLU finale (**Figure 3.22**).
- Retirer fully connected layer.
- Réseau résiduel profond : facile à former et optimiser, gagne en précision grâce à la profondeur **[ICCNN] [PCNNA]**.

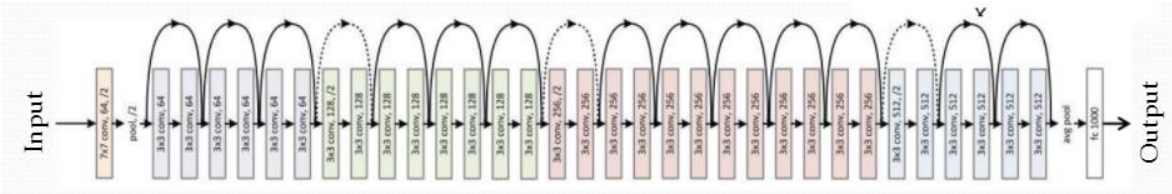


Figure 3.21 : Image illustre l'architecture de ResNET **[ICCNN].**

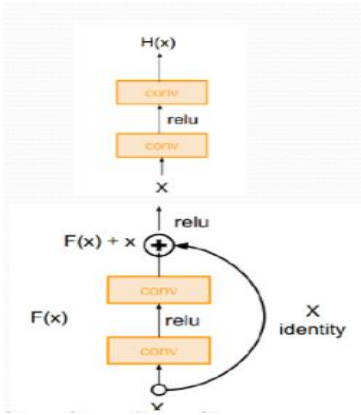


Figure 3.22 : l'architecture de ResNET (la transformation et l'additionnement de la sortie dans la fonction ReLU finale) **[ICCNN].**

3.4.7 DenseNet (2017)

- Dense block.
- Réutilisation des fonctionnalités.
- Petit nombre de filtres et moins de paramètres.
- Fonction de transformation non linéaire.
- Utilise des fonctionnalités de tous niveaux de complexité **[ICCNN].**

3.4.8 CapsNet (2018)

- CapsNet est composé de capsules.
- La capsule est écrasée comme un vecteur entier.
- Routage dynamique par accord.
- Problèmes dans CNN traditionnel : Le sous-échantillonnage perd les informations spatiales entre les entités de niveau supérieur **[ICCNN].**

3.5 Conclusion

D'après la notion de convolution et le principe de fonctionnement de ANN, on remarque que la convolution a pour avantage de n'avoir que quelques poids à calculer (ceux du filtre), et qu'elle les réutilise pour l'ensemble de l'image tandis que les ANN vont avoir un poids unique à calculer pour chaque neurone**.** Ce qui réduit encore considérablement le nombre de calculs.

Pour simplifier l'explication du processus de la couche de convolution, il faut savoir qu'une image se représente en 3D, il y a 2 dimensions qui correspondent à la largeur et à la hauteur de l'image et une troisième dimension qui correspond à la composante couleur (on rappelle : un pixel est composé de composantes rouge, vert et bleu -RVB). Ainsi une image sera un tableau de taille 3D (largeur, hauteur, RVB).

Quand on effectue une convolution sur une image, on aura ainsi 3 cartes de caractéristiques en sortie (une pour chaque composante couleur). Il faut savoir aussi qu'on peut effectuer plusieurs convolutions sur une même image (ou sur une même carte de caractéristiques). L'idée est que chaque convolution que l'on effectue sur une image correspondra à une caractéristique particulière de cette image**.**

On a un long vecteur qui comprend les caractéristiques les plus pertinentes de l'image à la sortie de la partie convolutive. On branche chaque valeur de ce vecteur à un neurone du réseau de neurone de la partie classification, que l'on appelle réseau de neurone entièrement connecté, elle permet de classer les images que l'on souhaite **Figure 3.23.**

Enfin, on conclue que CNN n'est qu'un type d'ANN.

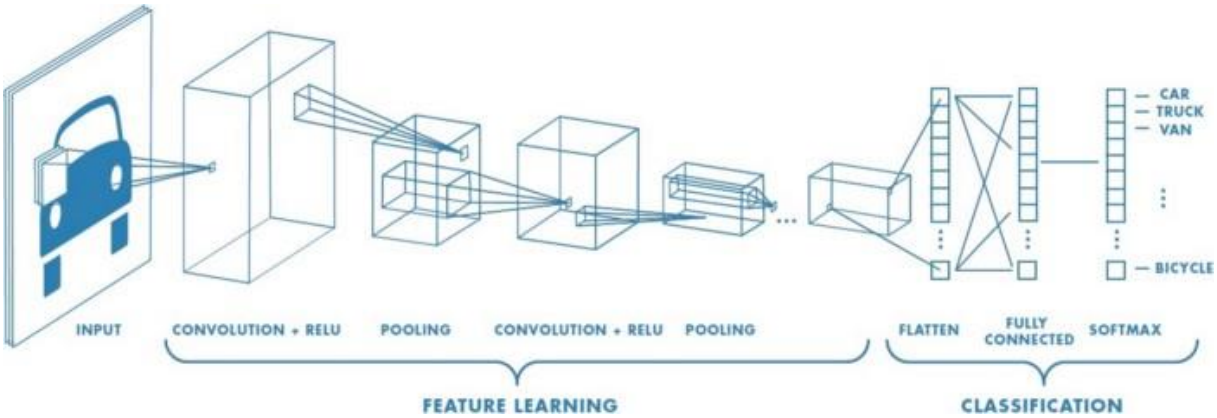


Figure 3.23 : Système neuronaux convolutive en générale.

Vu que c'est trop d'informations à apprendre, alors on résume comment un CNN fonctionne à nouveau **[CNNPP] :**

1. Donner l'image d'entrée **(Figure 3.24)** dans la couche de convolution, sélectionner les paramètres de convolution, de remplissage et de taille de filtre puis effectuer une convolution sur l'image.

2. Effectuer un regroupement (pooling) sur la sortie de la couche de convolution pour réduire la taille, ajouter plus de couches convolutives jusqu'à ce que vous soyez satisfait.

3. L'aplatissement (Flattening) converti la sortie de la couche convolutive sous forme vectorielle (Code CNN). Ce vecteur est l'entrée de la couche entièrement connectée.

4. la sortie de la couche entièrement connectée donne la classe en utilisant une fonction d'activation telle que la fonction Softmax ou sigmoïde **(Figure 3.24).**

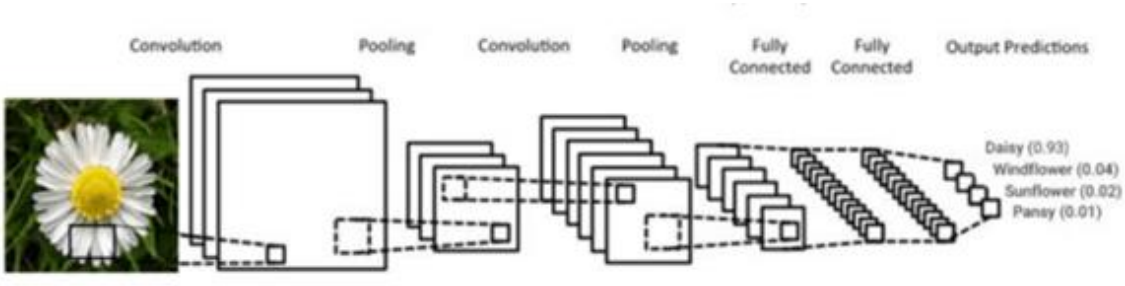


Figure 3.24: Artificial Neural Networks (ANN) and convolutional neural Networks (CNN).

Chapitre 4 Résultats pratiques

4.1 Introduction

Dans ce chapitre, nous allons introduire la partie pratique. Nous ferons l'apprentissage d'une base de données audio par CNN ainsi que le teste de notre model d'apprentissage.

Les sons que nous allons distinguer font partie du son de l'ensemble de données urbainound8k qui se compose de plus de 8000 sons de 10 classes (catégories), mais nous n'utiliserons que "7464".

Afin de visualiser un signal audio, nous allons les convertir en mel spectrogrammes, ce sont des représentations visuelles 2D de fréquence variant dans le temps.

Une fois que nous aurons les représentations des mel spectrogrammes pour nos données, elles seront traitées par des réseaux de neurones convolutifs.

CNN peut discriminer les modèles spectro-temporels. En d'autres termes, CNN est capable de capturer des modèles à travers le temps et la fréquence des spectrogrammes. Ceci est important pour faire la distinction entre le bruit dans les sons de notre ensemble de données.

L'architecture de CNN que nous choisirons est proposé par Justin Salamon et Juan Pablo Bello (SB_CNN) dans cet article **[CNNDESC]**.

4.2 Les outils utilisés

Nous avons utilisé Python comme langage de programmation sous l'environnement de développement Google Colab.

4.2.1 Python

Python est un langage de programmation open source créé par le programmeur Guido van Rossum en 1991**,** il est le plus utilisé dans le domaine du Machine Learning, du Big Data et de la Data Science. Avec le progrès de l'analyse de données dans toutes les industries, c'est d'ailleurs devenu l'un de ses principaux cas d'usage. La plupart des bibliothèques utilisées pour la science des données ou le Machine Learning ont des interfaces Python. Donc, ce langage est devenu l'interface de commande de haut niveau la plus connus pour les bibliothèques de Machine Learning et autres algorithmes numériques.

Il s'agit d'un langage de programmation interprété, qui ne n'oblige pas d'être compilé pour fonctionner. Il permet d'exécuter le code Python sur n'importe quel ordinateur, ceci permet de voir rapidement les résultats d'un changement dans le code. Par contre, ceci rend ce langage plus lent qu'un langage compilé comme le C.

Vu que le langage de programmation est de haut niveau, Python permet aux programmeurs de se concentrer sur ce qu'ils font plutôt que sur la manière dont ils le font. Ainsi, écrire des programmes prend moins de temps que dans un autre langage et il s'agit d'un langage idéal pour les débutants**.** Le cas d'usage essentiel du Python est le scripting et l'automatisation des interactions avec les navigateurs web.

4.2.2 Google Colab

Colaboratory connu "Colab" est un produit de Google Research. Il permet à n'importe qui d'écrire et d'exécuter le code Python de son choix par le biais du navigateur. C'est un environnement de développement particulièrement adapter au Machine Learning, à l'analyse de données et à l'apprentissage. En termes plus techniques, Colab est un service hébergé de notebooks Jupyter qui ne nécessite aucune configuration et permet d'accéder gratuitement à des ressources informatiques, dont des GPU**.**

Les codes dans Colab sont exécutés sur une machine virtuelle propre dans un compte Colab, aussi les machines virtuelles sont supprimées lorsqu'elles sont inactives

pendant un certain temps et ont une durée de vie maximale imposée par le service Colab.

La différence entre Jupyter et Colab c'est que Jupyter est le projet Open Source sur lequel Colab est basé. Colab permet d'utiliser et de partager des notebooks Jupyter avec d'autres personnes, sans avoir besoin de télécharger, d'installer ni d'exécuter quoi que ce soit, les notebooks Colab sont stockés dans Google Drive.

a Types de GPU disponibles dans Colab

Les types de GPU disponibles dans Colab peuvent varier au fil du temps. Cette variation est nécessaire pour maintenir un accès gratuit aux ressources de Colab. Les GPU disponibles dans Colab incluent souvent les modèles K80, T4, P4 et P100 de Nvidia. Il n'est pas possible de choisir le type de GPU mis à disposition par Colab à un moment donné.

b Temps d'exécution des notebooks dans Colab

Les notebooks sont exécutés par connexion à des machines virtuelles (VM) dont la durée de disponibilité maximale va jusqu'à 12 heures. Les notebooks se déconnectent des VM en cas d'inactivité prolongée. La durée de disponibilité maximale ainsi que le délai d'expiration en cas d'inactivité peuvent varier au fil du temps ou suivant votre utilisation. Ce sont des contraintes nécessaires pour permettre à Colab de proposer un accès gratuit à ces ressources informatiques.

c La quantité de mémoire disponible dans Colab

La capacité de mémoire des machines virtuelles de Colab varie au fil du temps. Elle restera cependant stable pendant la durée d'utilisation d'une même VM. (L'ajustement de la mémoire au fil du temps permet de continuer à proposer Colab gratuitement.) Une machine virtuelle avec de la mémoire supplémentaire peut vous être attribuée automatiquement si Colab détecte que vous en aurez probablement besoin.

4.3 DataSet

On a choisi pour notre projet comme base de données Urban Sound 8k **[US8K]**. Cette base de données contient 8732 extraits sonores classer (<= 4s) de sons urbains de 10 classes : air_conditioner, car_horn, children_playing, dog_bark, forage, enginge_idling, gun_shot, jack Hammer, siren et street_music. Les classes sont tirées de la taxonomie du son urbain.

Tous les extraits sont tirés d'enregistrements de terrain téléchargés sur www.freesound.org. Les fichiers sont prétriés en 10 dossiers (dossiers nommés fold1-fold10) pour aider à la reproduction et à la comparaison avec les résultats de classification automatique.

En plus des extraits sonores, un fichier CSV contenant des métadonnées sur chaque extrait est également fourni (8732 fichiers audio de sons urbains au format WAV).

Ceci comprend :

- slice_file_name: Le nom du fichier audio. Le nom prend le format suivant : [fsID] - [classID] - [occurrenceID] - [sliceID] .wav.
- fsID : L'ID Freesound de l'enregistrement à partir duquel cet extrait (tranche) est tiré
- Start : L'heure de début de la tranche dans l'enregistrement Freesound original
- End : L'heure de fin de la tranche dans l'enregistrement Freesound original
- Salience : Une cote de saillance (subjective) du son. 1 = premier plan, 2 = arrière-plan.
- Fold : Le numéro de pli (1-10) auquel ce fichier a été attribué.
- ClassID : Un identifiant numérique de la classe sonore :

  0 = climatiseur

  1 = car_horn

  2 = jeux_enfants

  3 = écorce de chien

  4 = perçage

  5 = moteur_idling

6 = coup de feu

7 = marteau-piqueur

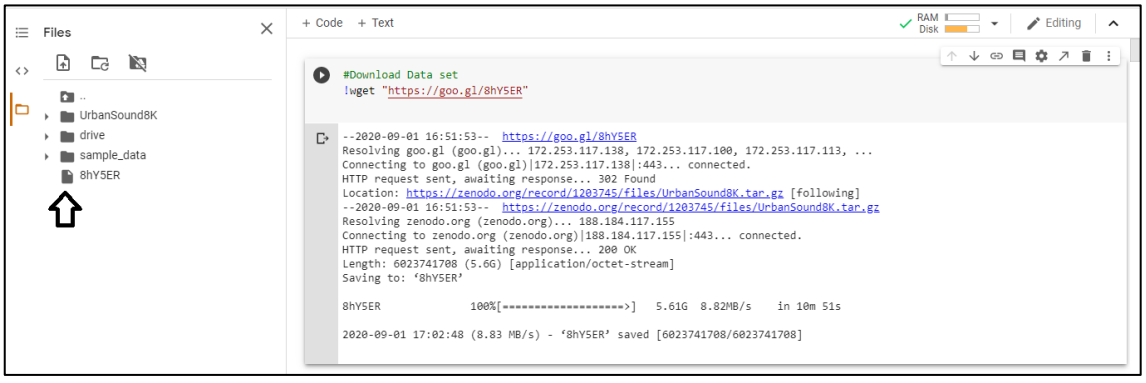
8 = sirène

9 = street_music

- Class: Le nom de la classes: air_conditioner, car_horn, children_playing, dog_bark, forage, engine_idling, gun_shot, jackhammer, sirène, street_music.

Pour le téléchargement de notre base de données dans python en Google colab, nous avons utilisé Wget qui est un programme en ligne de commande non interactif de téléchargement de fichiers depuis le Web. Elle supporte les protocoles HTTP, HTTPS et FTP ainsi que le téléchargement au travers des proxies http.

Une fois le téléchargement terminé, on aura un fichier de 5.6 GB nommé « 8hY5ER » de format «.tar ».



Puis, on ouvre notre fichier télécharger 8hY5ER de format « .tar » en utilisant une instruction de l'ouverture.

Après la fin de cette exécution, nous aurons un dossier qui représente notre base de données nommé UrbanSound8K.

Enfin, notre base de données est prête à l'utilisation après avoir terminé son téléchargement et son ouverture.

4.4 Prétraitement et Feature extraction

4.4.1 Les bibliothèques

Nous commençons par l'importation des bibliothèques suivantes :

« Keras » : notre deep learning framework basé sur le back-end TensorFlow pour créer et entrainer CNN.

« Librosa » : qui nous permet de lire et d'écrire et de jouer avec des fichiers audios.

« Numpy » : est une bibliothèque mathématique pour exprimer les features maps d'entrées et les sorties de matrices.

« Pandas » : est une bibliothèque qui permet de manipuler les données stockées dans des tables à l'aide de tuple de données, nous l'utilisons pour lire les metadata.

« Random » : est utilisé pour mélanger notre ensemble de données.

« Matplotlib » : est une bibliothèque complète pour créer des visualisations statiques, animées et interactives en Python.

```
[7] #Library section
    import keras
    from keras.layers import Activation, Dense, Dropout, Conv2D, \
                             Flatten, MaxPooling2D
    from keras.models import Sequential
    import librosa
    import librosa.display
    import numpy as np
    import pandas as pd
    import random
    import matplotlib

    import warnings
    warnings.filterwarnings('ignore')



    !pip install matplotlib
    import matplotlib.pyplot as plt

    Requirement already satisfied: matplotlib in /usr/local/lib/python3.6/dist-packages (3.2.2)
    Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib) (2.4.7)
    Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib) (1.2.0)
    Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-packages (from matplotlib) (0.10.0)
    Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib) (2.8.1)
    Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.6/dist-packages (from matplotlib) (1.18.5)
    Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from cycler>=0.10->matplotlib) (1.15.0)
```

4.4.2 Lecture de database

- Nous lisons le fichier Metadata de l'Urbansound8k qui contient un tuple d'informations sur les fichiers audios (comme le nom de la classe et les heures de début et de fin…) pour cela, nous déclarons une variable nommée « data » qui contient la fonction qui va lire le fichier Metadata :

```
# Read Data
data = pd.read_csv('UrbanSound8K/metadata/UrbanSound8K.csv')
data.head(5)
```

	slice_file_name	fsID	start	end	salience	fold	classID	class
0	100032-3-0-0.wav	100032	0.0	0.317551	1	5	3	dog_bark
1	100263-2-0-117.wav	100263	58.5	62.500000	1	5	2	children_playing
2	100263-2-0-121.wav	100263	60.5	64.500000	1	5	2	children_playing
3	100263-2-0-126.wav	100263	63.0	67.000000	1	5	2	children_playing
4	100263-2-0-137.wav	100263	68.5	72.500000	1	5	2	children_playing

Tableau 4.1 : le fichier Metadata de l'Urbansound8k pour les 5 première ligne.

  - Les tuples sont des collections d'objets ordonnés et invariables. Ce sont des séquences, tout comme les listes. Les différences entre les tuples et les listes sont que les tuples ne peuvent pas être modifiés contrairement aux listes **[PDC]**.

- Après, nous utilisons «data.shape» (shape donne une indication du nombre de dimensions dans le tableau) pour déterminer le nombre des lignes (audios) et le nombre de colonne (slice_file_name, fsID, start, end, salience, fold, classID, class) dans notre variable « data ».

```
[10] data.shape

(8732, 8)
```

  - Donc Metadata contient 8 colonnes et 8732 lignes

- Nous appliquons un code pour afficher le tableau qui illustre la distribution des audios de chaque 10 classes dans chaque dossier "fold" :

Le tableau :

	index	jackhammer	dog_bark	drilling	air_conditioner	street_music	children_playing	engine_idling	siren	car_horn	gun_shot
0	fold1	120	100	100	100	100	100	96	86	36	35
1	fold2	120	100	100	100	100	100	100	91	42	35
2	fold3	120	100	100	100	100	100	107	119	43	36
3	fold4	120	100	100	100	100	100	107	166	59	38
4	fold5	120	100	100	100	100	100	107	71	98	40
5	fold6	68	100	100	100	100	100	107	74	28	46
6	fold7	76	100	100	100	100	100	106	77	28	51
7	fold8	78	100	100	100	100	100	88	80	30	30
8	fold9	82	100	100	100	100	100	89	82	32	31
9	fold10	96	100	100	100	100	100	93	83	33	32

Tableau 4.2 : le tableau qui illustre la distribution des audios de chaque 10 classes dans chaque dossier "fold"

La plupart de ces clips audios durent environ 3 à 4 secondes.

4.4.3 La débarrassions des clips audio parasites

Pour une entrée uniforme dans le modèle, nous ne voulons pas des clips sonores parasites qui sont trop courts pour obtenir des informations utiles, alors nous nous débarrassons d'eux en calculant la durée des clips audios.

- Pour le calcul de la durée des clips audios, nous déclarons une variable « valid_data » comme de suite :
  - « Valid_data » : elle contient la variable « data » dont nous prenons que 4 colonnes spécifiques parmi les 8 colonnes du tableau Metadata qui sont : slice_file_name, fold, classID, class. Puis on fait la soustraction qui doit être supérieur ou égale 3s entre la fin et le début de chaque son. Par conséquent, nous avons dégager tous les clips sonores courts comme nous l'observons dans l'exécution du « valid_data.shape » : le nombre de lignes a diminué par rapport au nombre de lignes en Metadata.

4.4.4 L’extraction des Mel spectrogrammes pour les clips audios

Pour mieux comprendre cette opération on convertir quelque exemple de mel spectrogramme sur quelque clips audio, après on bien explique l’opération de conversion de tous nos données

a Des exemples sur Mel spectrogramme des clips audio

- Nous convertissons les clips audios au mel spectrogramme de l’échelle logarithmique en utilisant l'instruction "librosa" puis nous affichons la figure du spectre.
- Voici un exemple de Mel spectrogramme d'un clip sirène, nous prenons en considération les trois premières secondes du clip pour garder une taille uniforme.
    - Notons que la couleur indique le volume, c’est à dire si la couleur est plus claire, le son sera plus fort.

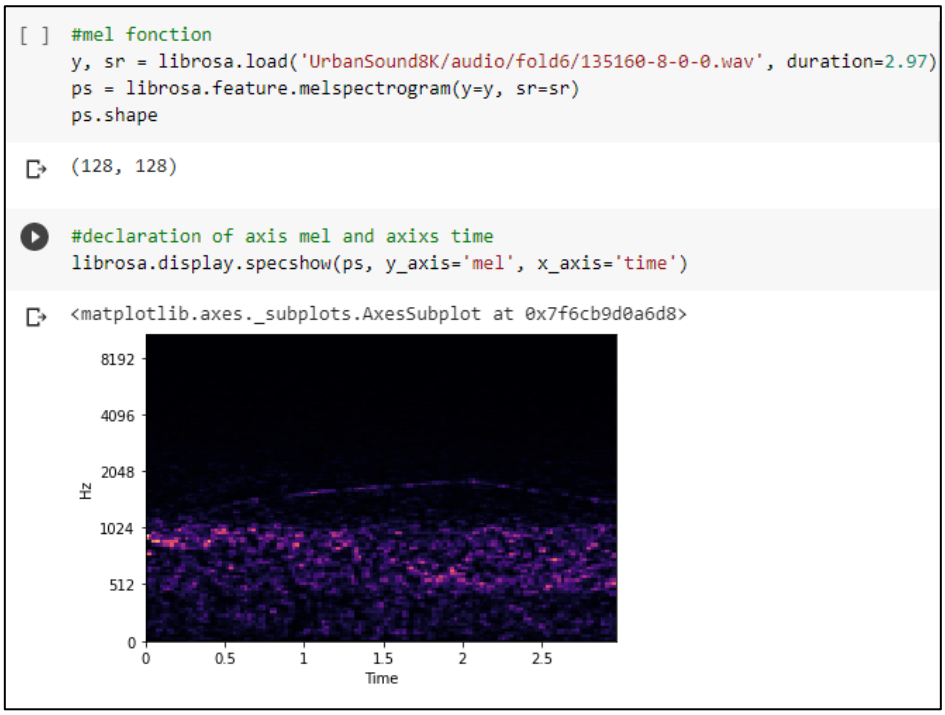


Figure 4.1 : exemple de Mel spectrogramme d'un clip sirène.

➢ Voici un autre exemple de Mel spectrogramme d'un clip children playing :

```
# Example of a children playing spectrogram
y, sr = librosa.load('UrbanSound8K/audio/fold9/13579-2-0-16.wav', duration=2.97)
ps = librosa.feature.melspectrogram(y=y, sr=sr)
ps.shape
```

```
(128, 128)
```

```
librosa.display.specshow(ps, y_axis='mel', x_axis='time')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6cb98d96d8>
```

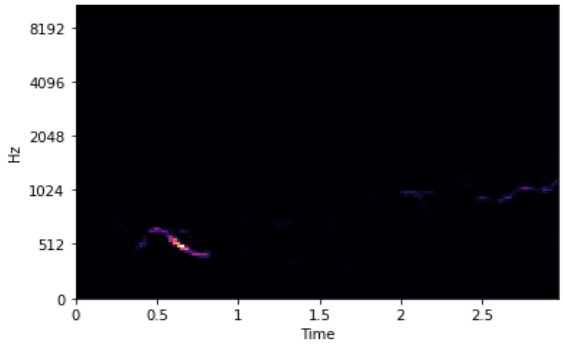


Figure 4.2 : un exemple de Mel spectrogramme d'un clip children playing.

➢ Pour mieux observer la différence voici un exemple de Mel spectrogramme d'un clip drilling :

```
# Example of a drilling spectrogram
y, sr = librosa.load('UrbanSound8K/audio/fold9/137815-4-0-0.wav', duration=2.97)
ps = librosa.feature.melspectrogram(y=y, sr=sr)
ps.shape
```

```
(128, 128)
```

```
librosa.display.specshow(ps, y_axis='mel', x_axis='time')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6cb6c04198>
```

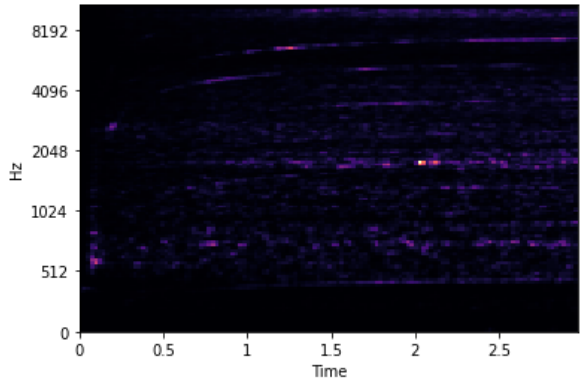


Figure 4.3 : exemple de Mel spectrogramme d'un clip drilling.

b Conversions de tous les clips audios de valid_data en mel spectrogramme

- Nous devons maintenant convertir tous les clips audios de « valid_data » en mel spectrogramme de l'échelle logarithmique, pour ce faire nous devons d'abord définir le path de « valid_data »

- L'entrée du CNN contient des patchs temps-fréquence (TF-patchs) extrait de la représentation en Mel spectrogramme à échelle logarithmique du signal audio pour cela :
  - Nous itérons sur tous les samples (les clips audios) de valid_data que nous avons commodément une liste dans les métadonnées et nous appliquons un Mel spectrogramme pour chacun de ces clips audios. Plus précisément, pour extraire des spectrogrammes mel à échelle logarithmique avec 128 composants (bandes) en utilisant une taille de fenêtre de 23 ms.
  - Les extraits de notre ensemble de données d'évaluation sont de durée variable (jusqu'à 4 s), nous fixons la taille de l'entrée TF-patch X à 3 secondes Cela conduit également à 128 composants à travers le temps (128 trames par un seul clip audio), soit les entrées globales $X \in R\ 128 \times 128$, sont une matrice 128 * 128 de nombres réels.
  - Nous nous retrouvons avec 7467 échantillons valides qui sont introduits dans notre CNN.

- En déclarent un vecteur D qui va présenter la conversion de « valid_data » en mel spectrogramme (cette étape prendre entre 30 à 50 min)

- Nous confirmons notre résultat de conversion des clips audio en mel spectrogramme :

```
[ ]  print("Number of samples: ", len(D))

     Number of samples:  7467
```

4.4.5 Déclaration de paramètres d'entrées du CNN

- Nous déclarons une nouvelle variable « dataset », elle contient le vecteur « D », puis en utilise la fonction « random.shuffle(dataset) » qui réorganise notre base de donnée.

- Après la réorganisation, nous déclarons les deux entrés qui sont :
  - « Train » contient 7000 images.
  - « Test » contient 467 images.
- Ensuite, nous remodelons (reshape) les entrées (train, test) en 128 * 128 * 1 pour l'entrée CNN et nous encodons les étiquettes de classe en utilisant un " One-Hot encoding " pour 10 classes.
  - One-Hot encoding : il prend une colonne contenant des données catégorielles (classes) qui ont été étiquetées. Les nombres sont remplacés par 1 et 0 (binaire) selon quelle colonne à quelle valeur **[LEVOHE]**.
- Enfin, nous avons affiché le nombre de samples dans les deux entrées « train » et « test ».

4.5 Model Training

L'architecture CNN (Deep Convolutional Neural Network) proposé dans cette étude **[CNNDESC]** est composé de 3 couches convolutives entrelacés avec 2 opérations de pooling (regroupement), suivies de 2 couches entièrement connectées (denses).

L'architecture de CNN proposée est paramétrée comme suit **[CNNDESC]** :

• **1ére Couche :** contient 24 filtres et la matrice de chaque filtre est de (5,5), cela signifie que la forme est de (24,1,5,5). Cela est suivi par (4,2) strided maxpooling sur les deux dimensions (temps et fréquence respectivement) et une activation d'unité linéaire redressée (ReLU) fonction h (x) = max (0, x).

• **2éme Couche :** comporte 48 filtres et la matrice de chaque filtre est de (5,5), cela signifie que la forme est de (48, 24, 5, 5). Comme la 1ere couche, ceci est suivi de (4,2) maxpooling strided et une fonction d'activation ReLU.

• **3éme Couche :** contient 48 filtres et la matrice de chaque filtre est de (5,5), c'est-à-dire que la forme est de (48, 48, 5, 5). Ceci est suivi d'un ReLU fonction d'activation (pas de pooling).

• **4éme Couche :** 64 unités cachées, c'est-à-dire que la forme est de (2400, 64), suivi d'une fonction d'activation ReLU.

• **5éme Couche :** 10 unités de sortie, c'est-à-dire que la forme est de (64,10), suivi d'une fonction d'activation softmax.

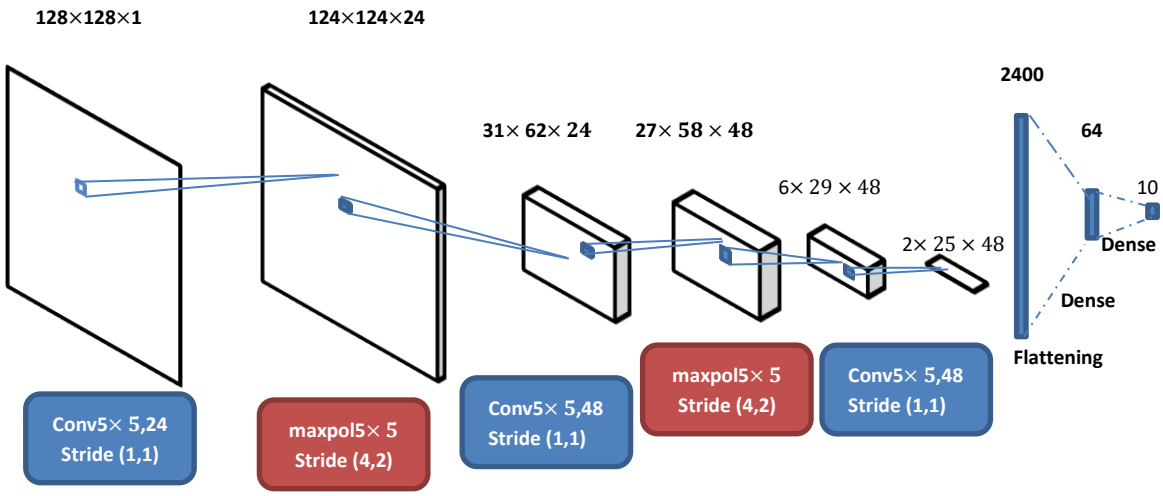


Figure 4.4 : Model d'entrainement utilisé pour CNN.

- Nous introduisons d'abord la couche où nous effectuons la convolution avec 24 filtres de (5*5) et stride de 1. La sortie de cette convolution est une carte de caractéristiques (features map) "124 * 124", mais puisque nous utilisons 24 de ces filtres, la sortie est devenue "124 * 124 * 24".

  Ensuite, nous appliquons maxpooling où chaque filtre est de 4 par 2 et un stride de 4 sur la hauteur et 2 sur la largeur. Cela conduit à une sortie de "31 * 62" pour

une seule couche. Mais nous appliquons à 24 cartes de caractéristiques de l'entrée convoluée, donc le volume de sortie sera "31 * 62 * 24".

Nous appliquons maintenant le RELU d'activation qui ne change pas la dimensionnalité.

- Une fois que cela est fait, nous appliquons une autre séquence de convolution, de pooling et d'activation. Pour la couche de convolution suivante, l'entrée est convoluée avec 48 filtres de (5*5) la convolution avec chaque filtre sans padding conduit à une carte de caractéristiques de "27 * 58". Puisque nous avons 48 de tels filtres la sortie est un volume 3D de forme "27 * 58 * 48".

  Une fois de plus, nous échantillonnons les fonctionnalités en appliquant maxpooling ou chaque filtre est (4*2) avec stride de 4 sur la hauteur et 2 sur la largeur, comme dans la dernière couche de pooling, cela conduit à une sortie "6 * 29" pour une seule couche. Puisqu'une couche de pooling est appliquée à chacune des 48 couches, la sortie est un volume 3D de la même profondeur 48, à ce volume 3D, nous appliquons une autre activation Relue.

- Ensuite, nous appliquons simplement un autre cycle d'activation de convolution sans pooling, elle est similaire à la précédente où nous convoluons le volume d'entrée avec 48 de (5*5) filtres sans padding conduit à un volume 3D de forme "2 * 25 * 48". L'activation ReLU suivante ne change pas de forme.

- À présent nous sommes à la partie finale de notre modèle des couches entièrement connectées (fully connected layer). Le volume "2 * 25 * 48" s'aplatit (flattening) pour former un vecteur dimensionnel de 2400 qui est sous une forme admissible pour les couches entièrement connectées. Nous appliquons ensuite une couche cachée de 64 neurones avec l'activation puis dropout (suppression). La couche finale est une couche softmax avec 10 neurones parce que nous avons 10 classes de sons de sortie et nous avons donc défini notre réseau de neurones à convolution.

Si vous voulez savoir exactement ce que fait chaque couche et comment ces dimensions sont calculées voir notre « chapitre 2 » « partie2 ».

Output :

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 124, 124, 24)      624
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 31, 62, 24)        0
_________________________________________________________________
activation (Activation)      (None, 31, 62, 24)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 27, 58, 48)        28848
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 6, 29, 48)         0
_________________________________________________________________
activation_1 (Activation)    (None, 6, 29, 48)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 2, 25, 48)         57648
_________________________________________________________________
activation_2 (Activation)    (None, 2, 25, 48)         0
_________________________________________________________________
flatten (Flatten)            (None, 2400)              0
_________________________________________________________________
dropout (Dropout)            (None, 2400)              0
_________________________________________________________________
dense (Dense)                (None, 64)                153664
_________________________________________________________________
activation_3 (Activation)    (None, 64)                0
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                650
_________________________________________________________________
activation_4 (Activation)    (None, 10)                0
=================================================================
Total params: 241,434
Trainable params: 241,434
Non-trainable params: 0
_________________________________________________________________
```

Tableau 4.3 : l'architecture proposée de CNN.

Le Dropout : ajouter une couche de dropout pour surmonter dans une certaine mesure le problème du surajustement (overfitting). L'abandon (dropout) désactive au hasard une fraction des neurones pendant le processus d'entraînement, réduisant ainsi la dépendance à l'ensemble d'entraînement. Le nombre de fractions de neurones que vous souhaitez désactiver est déterminé par un hyperparamètre, qui peut être réglé en conséquence. De cette façon, la désactivation de certains neurones ne permettra pas au réseau de mémoriser les données d'entraînement car tous les neurones ne seront pas actifs en même temps et les neurones inactifs ne pourront rien apprendre **[CTCNNP].**

4.6 Résultats

Finalement, les derniers codes contiennent :

- Model sauvegarde : Model Checkpoint callback est utilisé en conjonction avec l'entraînement à l'aide de model.fit () pour enregistrer un modèle à la fin de chaque époque et ne conserver que le modèle qui a atteint les « meilleures performances » jusqu'à présent.

- Model Compile définit la fonction de loss, l'optimiseur et les metrics **[DMCOWB]** :

    - **Metrics = ['accuracy'] :** accuracy metrics est une fonction utilisée pour juger des performances de votre modèle. Accuracy metrics calcule le taux de précision pour toutes les prédictions **[KAM]**.
    - **Loss = "categorical_crossentropy" :** Il s'agit d'une activation Softmax plus une perte d'entropie croisée. Si nous utilisons cette perte, nous entraînerons un CNN pour produire une probabilité sur les classes C pour chaque image. Il est utilisé pour la classification multi-classes **[MLTAL]**.

- Model fitting (L'ajustement du modèle) contient x=X_train, y=y_train, epochs=68, batch_size=64, validation_data= (X_test, y_test) **[FITING].**
    - **X et y :** sont les entrées de l'apprentissage.
    - **Batch_size :** est un hyperparamètre qui définit le nombre d'échantillons à traiter. Les valeurs de batch_size sont des valeurs entières **[DBBAE]**.
    - **Le nombre de Epochs :** correspond au nombre de passages complets dans l'ensemble de données d'apprentissage. Le nombre d'époques peut être défini sur une valeur entière comprise entre 1 et l'infini **[DBBAE]**.
    - **X_test, y_test** : sont les entrés de test.

- Mettons en perspective l'évaluation de notre modèle et traçons les graphiques de accuracy et de loss entre les données d'entraînement et de validation.

Output :

```
[ ] Epoch 1/68
    109/110 [============================>.] - ETA: 0s - loss: 0.1321 - accuracy: 0.9741
    Epoch 00001: val_loss did not improve from 0.37815
    110/110 [==============================] - 4s 37ms/step - loss: 0.1317 - accuracy: 0.9741 - val_loss: 0.8739 - val_accuracy: 0.9079
    Epoch 2/68
    109/110 [============================>.] - ETA: 0s - loss: 0.1026 - accuracy: 0.9743
    Epoch 00002: val_loss did not improve from 0.37815
    110/110 [==============================] - 3s 31ms/step - loss: 0.1033 - accuracy: 0.9740 - val_loss: 0.8864 - val_accuracy: 0.9079
    Epoch 3/68
    109/110 [============================>.] - ETA: 0s - loss: 0.0877 - accuracy: 0.9775
    Epoch 00003: val_loss did not improve from 0.37815
    110/110 [==============================] - 3s 31ms/step - loss: 0.0875 - accuracy: 0.9776 - val_loss: 0.7080 - val_accuracy: 0.9208
    Epoch 4/68
    109/110 [============================>.] - ETA: 0s - loss: 0.1043 - accuracy: 0.9743
    Epoch 00004: val_loss did not improve from 0.37815
    110/110 [==============================] - 3s 31ms/step - loss: 0.1040 - accuracy: 0.9744 - val_loss: 0.5966 - val_accuracy: 0.9251
    Epoch 5/68
    109/110 [============================>.] - ETA: 0s - loss: 0.1077 - accuracy: 0.9709
    Epoch 00005: val_loss did not improve from 0.37815
    110/110 [==============================] - 3s 31ms/step - loss: 0.1073 - accuracy: 0.9710 - val_loss: 0.9881 - val_accuracy: 0.9186
    Epoch 6/68
    109/110 [============================>.] - ETA: 0s - loss: 0.1076 - accuracy: 0.9763
    Epoch 00006: val_loss did not improve from 0.37815
    110/110 [==============================] - 3s 31ms/step - loss: 0.1076 - accuracy: 0.9761 - val_loss: 0.8473 - val_accuracy: 0.9165
    Epoch 7/68
    109/110 [============================>.] - ETA: 0s - loss: 0.2089 - accuracy: 0.9670
    Epoch 00007: val_loss did not improve from 0.37815
    110/110 [==============================] - 3s 31ms/step - loss: 0.2116 - accuracy: 0.9667 - val_loss: 0.9065 - val_accuracy: 0.8994
```

```
    Epoch 8/68
    109/110 [============================>.] - ETA: 0s - loss: 0.1619 - accuracy: 0.9643
    Epoch 00008: val_loss did not improve from 0.37815
    110/110 [==============================] - 3s 31ms/step - loss: 0.1613 - accuracy: 0.9644 - val_loss: 0.7271 - val_accuracy: 0.8972
    Epoch 9/68
    109/110 [============================>.] - ETA: 0s - loss: 0.0643 - accuracy: 0.9798
    Epoch 00009: val_loss did not improve from 0.37815
    110/110 [==============================] - 3s 31ms/step - loss: 0.0643 - accuracy: 0.9797 - val_loss: 0.4808 - val_accuracy: 0.9229
    Epoch 10/68
    109/110 [============================>.] - ETA: 0s - loss: 0.1184 - accuracy: 0.9752
    Epoch 00010: val_loss did not improve from 0.37815
    110/110 [==============================] - 3s 31ms/step - loss: 0.1185 - accuracy: 0.9751 - val_loss: 0.7419 - val_accuracy: 0.8951
    Epoch 11/68
    109/110 [============================>.] - ETA: 0s - loss: 0.0880 - accuracy: 0.9774
    Epoch 00011: val_loss did not improve from 0.37815
    110/110 [==============================] - 3s 31ms/step - loss: 0.0880 - accuracy: 0.9773 - val_loss: 0.5331 - val_accuracy: 0.9165
    Epoch 12/68
    109/110 [============================>.] - ETA: 0s - loss: 0.1303 - accuracy: 0.9752
    Epoch 00012: val_loss did not improve from 0.37815
    110/110 [==============================] - 3s 31ms/step - loss: 0.1298 - accuracy: 0.9753 - val_loss: 0.7898 - val_accuracy: 0.9229
    Epoch 13/68
    109/110 [============================>.] - ETA: 0s - loss: 0.2111 - accuracy: 0.9644
    Epoch 00013: val_loss did not improve from 0.37815
    110/110 [==============================] - 3s 31ms/step - loss: 0.2111 - accuracy: 0.9641 - val_loss: 0.9487 - val_accuracy: 0.9058
    Epoch 14/68
    109/110 [============================>.] - ETA: 0s - loss: 0.1221 - accuracy: 0.9743
    Epoch 00014: val_loss did not improve from 0.37815
    110/110 [==============================] - 3s 31ms/step - loss: 0.1225 - accuracy: 0.9741 - val_loss: 0.7720 - val_accuracy: 0.9058
```

```
    Epoch 15/68
    109/110 [============================>.] - ETA: 0s - loss: 0.0830 - accuracy: 0.9806
    Epoch 00015: val_loss did not improve from 0.37815
    110/110 [==============================] - 3s 31ms/step - loss: 0.0828 - accuracy: 0.9807 - val_loss: 0.7341 - val_accuracy: 0.9165
    Epoch 16/68
    109/110 [============================>.] - ETA: 0s - loss: 0.0709 - accuracy: 0.9805
    Epoch 00016: val_loss did not improve from 0.37815
    110/110 [==============================] - 3s 31ms/step - loss: 0.0706 - accuracy: 0.9806 - val_loss: 0.6773 - val_accuracy: 0.9229
    Epoch 17/68
    109/110 [============================>.] - ETA: 0s - loss: 0.0695 - accuracy: 0.9828
    Epoch 00017: val_loss did not improve from 0.37815
    110/110 [==============================] - 3s 31ms/step - loss: 0.0695 - accuracy: 0.9827 - val_loss: 0.5883 - val_accuracy: 0.9251
    Epoch 18/68
    109/110 [============================>.] - ETA: 0s - loss: 0.0514 - accuracy: 0.9851
    Epoch 00018: val_loss did not improve from 0.37815
    110/110 [==============================] - 3s 31ms/step - loss: 0.0513 - accuracy: 0.9851 - val_loss: 0.7521 - val_accuracy: 0.9079
    Epoch 19/68
    109/110 [============================>.] - ETA: 0s - loss: 0.6513 - accuracy: 0.9384
    Epoch 00019: val_loss did not improve from 0.37815
    110/110 [==============================] - 3s 31ms/step - loss: 0.6491 - accuracy: 0.9386 - val_loss: 0.7569 - val_accuracy: 0.9036
    Epoch 20/68
    109/110 [============================>.] - ETA: 0s - loss: 0.1319 - accuracy: 0.9702
    Epoch 00020: val_loss did not improve from 0.37815
    110/110 [==============================] - 3s 31ms/step - loss: 0.1315 - accuracy: 0.9703 - val_loss: 0.6079 - val_accuracy: 0.9143
    Epoch 21/68
    109/110 [============================>.] - ETA: 0s - loss: 0.1040 - accuracy: 0.9768
    Epoch 00021: val_loss did not improve from 0.37815
    110/110 [==============================] - 3s 31ms/step - loss: 0.1036 - accuracy: 0.9769 - val_loss: 0.7219 - val_accuracy: 0.9079
```

```
Epoch 29/68
109/110 [============================>.] - ETA: 0s - loss: 0.0624 - accuracy: 0.9841
Epoch 00029: val_loss did not improve from 0.37815
110/110 [==============================] - 4s 32ms/step - loss: 0.0622 - accuracy: 0.9841 - val_loss: 0.4727 - val_accuracy: 0.9143
Epoch 30/68
109/110 [============================>.] - ETA: 0s - loss: 0.0467 - accuracy: 0.9865
Epoch 00030: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 32ms/step - loss: 0.0468 - accuracy: 0.9864 - val_loss: 0.4531 - val_accuracy: 0.9251
Epoch 31/68
109/110 [============================>.] - ETA: 0s - loss: 0.0463 - accuracy: 0.9834
Epoch 00031: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 0.0497 - accuracy: 0.9829 - val_loss: 0.4890 - val_accuracy: 0.9122
Epoch 32/68
109/110 [============================>.] - ETA: 0s - loss: 0.0437 - accuracy: 0.9842
Epoch 00032: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 0.0437 - accuracy: 0.9843 - val_loss: 0.3882 - val_accuracy: 0.9186
Epoch 33/68
109/110 [============================>.] - ETA: 0s - loss: 0.0384 - accuracy: 0.9867
Epoch 00033: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 0.0385 - accuracy: 0.9867 - val_loss: 0.4862 - val_accuracy: 0.9165
Epoch 34/68
109/110 [============================>.] - ETA: 0s - loss: 0.0358 - accuracy: 0.9875
Epoch 00034: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 0.0357 - accuracy: 0.9876 - val_loss: 0.4371 - val_accuracy: 0.9208
Epoch 35/68
109/110 [============================>.] - ETA: 0s - loss: 0.0433 - accuracy: 0.9882
Epoch 00035: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 0.0432 - accuracy: 0.9883 - val_loss: 0.4952 - val_accuracy: 0.9293
```

```
Epoch 37/68
109/110 [============================>.] - ETA: 0s - loss: 0.0682 - accuracy: 0.9835
Epoch 00037: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 0.0680 - accuracy: 0.9836 - val_loss: 0.6330 - val_accuracy: 0.9208
Epoch 38/68
109/110 [============================>.] - ETA: 0s - loss: 0.0469 - accuracy: 0.9851
Epoch 00038: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 0.0468 - accuracy: 0.9851 - val_loss: 0.6820 - val_accuracy: 0.9165
Epoch 39/68
109/110 [============================>.] - ETA: 0s - loss: 0.0650 - accuracy: 0.9867
Epoch 00039: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 0.0650 - accuracy: 0.9866 - val_loss: 0.5760 - val_accuracy: 0.9229
Epoch 40/68
109/110 [============================>.] - ETA: 0s - loss: 0.0608 - accuracy: 0.9795
Epoch 00040: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 0.0614 - accuracy: 0.9794 - val_loss: 0.4896 - val_accuracy: 0.9165
Epoch 41/68
109/110 [============================>.] - ETA: 0s - loss: 0.0760 - accuracy: 0.9808
Epoch 00041: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 0.0757 - accuracy: 0.9809 - val_loss: 0.4737 - val_accuracy: 0.9015
Epoch 42/68
109/110 [============================>.] - ETA: 0s - loss: 0.0557 - accuracy: 0.9822
Epoch 00042: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 0.0555 - accuracy: 0.9823 - val_loss: 0.6055 - val_accuracy: 0.9229
Epoch 43/68
109/110 [============================>.] - ETA: 0s - loss: 0.0489 - accuracy: 0.9848
Epoch 00043: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 0.0488 - accuracy: 0.9849 - val_loss: 0.6124 - val_accuracy: 0.9336
```

```
Epoch 22/68
109/110 [============================>.] - ETA: 0s - loss: 0.1889 - accuracy: 0.9586
Epoch 00022: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 0.1885 - accuracy: 0.9586 - val_loss: 0.7367 - val_accuracy: 0.9143
Epoch 23/68
109/110 [============================>.] - ETA: 0s - loss: 0.0967 - accuracy: 0.9776
Epoch 00023: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 0.0964 - accuracy: 0.9777 - val_loss: 0.6801 - val_accuracy: 0.9186
Epoch 24/68
109/110 [============================>.] - ETA: 0s - loss: 0.0654 - accuracy: 0.9818
Epoch 00024: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 0.0652 - accuracy: 0.9819 - val_loss: 0.6285 - val_accuracy: 0.9251
Epoch 25/68
109/110 [============================>.] - ETA: 0s - loss: 0.0571 - accuracy: 0.9849
Epoch 00025: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 0.0569 - accuracy: 0.9850 - val_loss: 0.6021 - val_accuracy: 0.9101
Epoch 26/68
109/110 [============================>.] - ETA: 0s - loss: 0.0884 - accuracy: 0.9762
Epoch 00026: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 0.0881 - accuracy: 0.9763 - val_loss: 0.5176 - val_accuracy: 0.9229
Epoch 27/68
109/110 [============================>.] - ETA: 0s - loss: 0.0535 - accuracy: 0.9829
Epoch 00027: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 0.0534 - accuracy: 0.9830 - val_loss: 0.5448 - val_accuracy: 0.9229
Epoch 28/68
109/110 [============================>.] - ETA: 0s - loss: 0.0489 - accuracy: 0.9860
Epoch 00028: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 32ms/step - loss: 0.0490 - accuracy: 0.9859 - val_loss: 0.4749 - val_accuracy: 0.9186
```

```
Epoch 44/68
109/110 [============================>.] - ETA: 0s - loss: 0.3018 - accuracy: 0.9599
Epoch 00044: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 0.3007 - accuracy: 0.9600 - val_loss: 0.6944 - val_accuracy: 0.8758
Epoch 45/68
109/110 [============================>.] - ETA: 0s - loss: 0.2338 - accuracy: 0.9619
Epoch 00045: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 0.2332 - accuracy: 0.9619 - val_loss: 0.7012 - val_accuracy: 0.8951
Epoch 46/68
109/110 [============================>.] - ETA: 0s - loss: 1.0579 - accuracy: 0.9369
Epoch 00046: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 1.0551 - accuracy: 0.9367 - val_loss: 0.5803 - val_accuracy: 0.8630
Epoch 47/68
109/110 [============================>.] - ETA: 0s - loss: 0.2963 - accuracy: 0.9415
Epoch 00047: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 0.2970 - accuracy: 0.9414 - val_loss: 0.5460 - val_accuracy: 0.8951
Epoch 48/68
109/110 [============================>.] - ETA: 0s - loss: 0.1577 - accuracy: 0.9614
Epoch 00048: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 0.1577 - accuracy: 0.9613 - val_loss: 0.4624 - val_accuracy: 0.9143
Epoch 49/68
109/110 [============================>.] - ETA: 0s - loss: 0.1578 - accuracy: 0.9715
Epoch 00049: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 0.1596 - accuracy: 0.9711 - val_loss: 0.5576 - val_accuracy: 0.9101
Epoch 50/68
109/110 [============================>.] - ETA: 0s - loss: 0.4687 - accuracy: 0.9497
Epoch 00050: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 0.4671 - accuracy: 0.9499 - val_loss: 0.4963 - val_accuracy: 0.9036
```

```
Epoch 51/68
109/110 [============================>.] - ETA: 0s - loss: 0.1547 - accuracy: 0.9633
Epoch 00051: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 0.1549 - accuracy: 0.9631 - val_loss: 0.4201 - val_accuracy: 0.9079
Epoch 52/68
109/110 [============================>.] - ETA: 0s - loss: 0.0884 - accuracy: 0.9739
Epoch 00052: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 0.0881 - accuracy: 0.9740 - val_loss: 0.4529 - val_accuracy: 0.9036
Epoch 53/68
109/110 [============================>.] - ETA: 0s - loss: 0.0714 - accuracy: 0.9786
Epoch 00053: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 32ms/step - loss: 0.0712 - accuracy: 0.9787 - val_loss: 0.4309 - val_accuracy: 0.9229
Epoch 54/68
109/110 [============================>.] - ETA: 0s - loss: 0.0833 - accuracy: 0.9794
Epoch 00054: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 0.0834 - accuracy: 0.9793 - val_loss: 0.4850 - val_accuracy: 0.9251
Epoch 55/68
109/110 [============================>.] - ETA: 0s - loss: 0.0707 - accuracy: 0.9779
Epoch 00055: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 0.0708 - accuracy: 0.9779 - val_loss: 0.4524 - val_accuracy: 0.9229
Epoch 56/68
109/110 [============================>.] - ETA: 0s - loss: 0.0683 - accuracy: 0.9829
Epoch 00056: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 0.0680 - accuracy: 0.9830 - val_loss: 0.5072 - val_accuracy: 0.9251
Epoch 57/68
109/110 [============================>.] - ETA: 0s - loss: 0.0609 - accuracy: 0.9792
Epoch 00057: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 32ms/step - loss: 0.0608 - accuracy: 0.9791 - val_loss: 0.5252 - val_accuracy: 0.9079
```

```
Epoch 58/68
109/110 [============================>.] - ETA: 0s - loss: 0.0629 - accuracy: 0.9828
Epoch 00058: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 0.0628 - accuracy: 0.9829 - val_loss: 0.4404 - val_accuracy: 0.9122
Epoch 59/68
109/110 [============================>.] - ETA: 0s - loss: 0.0631 - accuracy: 0.9786
Epoch 00059: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 0.0629 - accuracy: 0.9787 - val_loss: 0.4785 - val_accuracy: 0.9186
Epoch 60/68
109/110 [============================>.] - ETA: 0s - loss: 0.0502 - accuracy: 0.9841
Epoch 00060: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 0.0501 - accuracy: 0.9841 - val_loss: 0.5280 - val_accuracy: 0.9251
Epoch 61/68
109/110 [============================>.] - ETA: 0s - loss: 0.0481 - accuracy: 0.9861
Epoch 00061: val_loss did not improve from 0.37815
110/110 [==============================] - 3s 31ms/step - loss: 0.0480 - accuracy: 0.9861 - val_loss: 0.4470 - val_accuracy: 0.9272
Epoch 62/68
109/110 [============================>.] - ETA: 0s - loss: 0.0618 - accuracy: 0.9849
Epoch 00062: val_loss improved from 0.37815 to 0.36735, saving model to model-062-0.985000-0.922912.h5
110/110 [==============================] - 4s 32ms/step - loss: 0.0616 - accuracy: 0.9850 - val_loss: 0.3674 - val_accuracy: 0.9229
Epoch 63/68
109/110 [============================>.] - ETA: 0s - loss: 0.0518 - accuracy: 0.9854
Epoch 00063: val_loss did not improve from 0.36735
110/110 [==============================] - 3s 31ms/step - loss: 0.0527 - accuracy: 0.9851 - val_loss: 0.5446 - val_accuracy: 0.9229
Epoch 64/68
109/110 [============================>.] - ETA: 0s - loss: 0.0981 - accuracy: 0.9774
Epoch 00064: val_loss did not improve from 0.36735
110/110 [==============================] - 3s 31ms/step - loss: 0.0979 - accuracy: 0.9774 - val_loss: 0.4851 - val_accuracy: 0.9165
```

```
Epoch 65/68
109/110 [============================>.] - ETA: 0s - loss: 0.0854 - accuracy: 0.9801
Epoch 00065: val_loss did not improve from 0.36735
110/110 [==============================] - 3s 31ms/step - loss: 0.0853 - accuracy: 0.9800 - val_loss: 0.6061 - val_accuracy: 0.9165
Epoch 66/68
109/110 [============================>.] - ETA: 0s - loss: 0.0744 - accuracy: 0.9821
Epoch 00066: val_loss did not improve from 0.36735
110/110 [==============================] - 3s 31ms/step - loss: 0.0750 - accuracy: 0.9820 - val_loss: 0.5380 - val_accuracy: 0.9229
Epoch 67/68
109/110 [============================>.] - ETA: 0s - loss: 0.0615 - accuracy: 0.9865
Epoch 00067: val_loss did not improve from 0.36735
110/110 [==============================] - 3s 31ms/step - loss: 0.0617 - accuracy: 0.9863 - val_loss: 0.5345 - val_accuracy: 0.9315
Epoch 68/68
109/110 [============================>.] - ETA: 0s - loss: 0.0426 - accuracy: 0.9871
Epoch 00068: val_loss did not improve from 0.36735
110/110 [==============================] - 3s 31ms/step - loss: 0.0426 - accuracy: 0.9871 - val_loss: 0.3872 - val_accuracy: 0.9229
15/15 [==============================] - 0s 8ms/step - loss: 0.3872 - accuracy: 0.9229
```

❖ **Résultat finale :**

```
Test loss: 0.3871501684188843
Test accuracy: 0.9229121804237366
Number of train samples 7000
Number of test samples 467
Accuracy: 92.29%
loss: 38.72%
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

Eh bien, après quelques minutes d'attente, nous avons finalement obtenu le résultat final. Nous pouvons voir ici que les scores de précision à la dernière époque sont respectivement de 98,71 % pour les données d'entraînement et 92,29 % pour le test.

- Et la sortie ressemble aux deux images ci-dessous :

➢ `accuracy : 0.9871 ; val_accuracy : 0.9229`

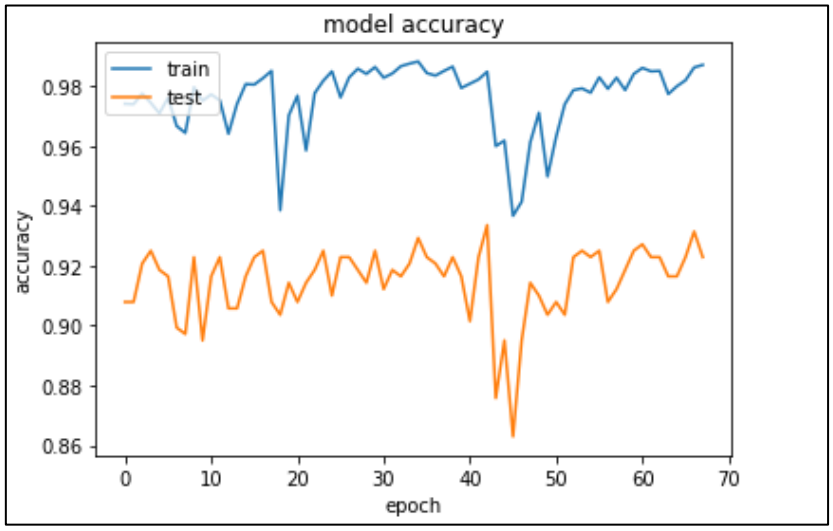


Figure 4.5 : Le model accuracy pour l'apprentissage et le teste.

- `loss : 0.0426 ; val_loss : 0.3872`

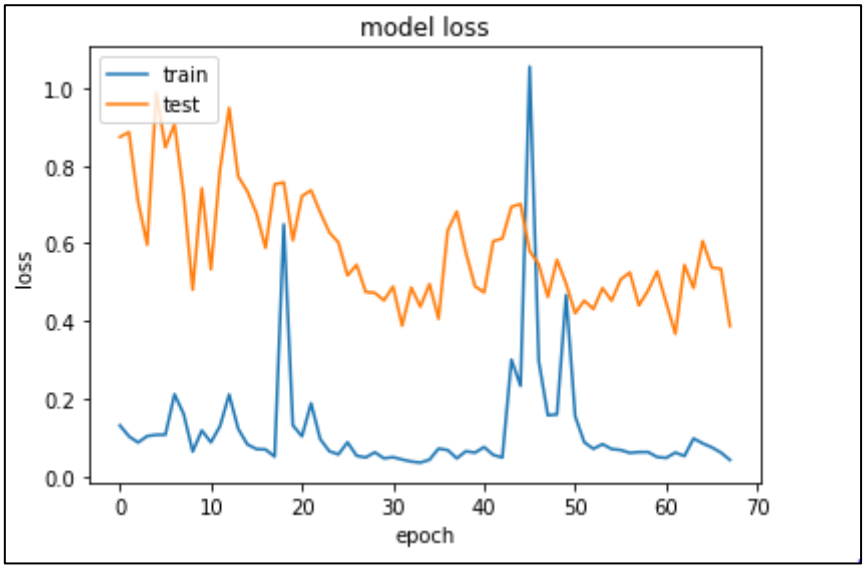


Figure 4.6 : Le model loss pour l'apprentissage et le teste.

D'après les deux graphiques ci-dessus, nous pouvons voir que notre modèle fonctionne plutôt bien car il atteint la précision finale de 98% et 92% pour les données de train et de test respectivement. En outre, les deux valeurs de loss sont un peu stables mais ils diminuent au fil du temps, également à mesure que le nombre d'époques augmente. Par conséquent, nous pouvons conclure que ce classificateur CNN ne souffre pas du tout de surajustement

4.7 Conclusion

Dans ce chapitre, nous avons démontré qu'il est possible d'appliquer une approche neuronale, qui a été déjà testé dans le domaine de la reconnaissance et de la classification sonore environnementale avec des résultats extrêmement positifs. En raison du succès de ce type d'approche, nous avons utilisé CNN pour notre base de données audios, et nous avons obtenu des résultats assez élevés. Notre architecture du CNN a été testé sur 10 classes avec une accuracy de 92%.

L'architecture du CNN qu'on a choisie pour notre travail est proposé par **[CNNDESC]** qui a été spécialement conçus pour une base de données audio**.**

L'architecture proposée fonctionne sur la représentation par mel spectrogramme des trames audio d'entrée, et démontre son efficacité dans la classification des sons environnementaux (ESC) en obtenant une grande précision.

La sélection des deux hyperparamètres principaux (epoch, batch_size) a une très grande influence dans les résultats. Nous avons sélectionné ces deux hyperparamètre à partir de plusieurs tests que nous avons effectués. Enfin, nous avons trouvé que les meilleures valeurs étaient : epoch = 68, et batch_size = 64.

Nous avons conclu que malgré la complexité de la classification audio par CNN les résultats fournis sont assez élevés.

Conclusion générale

Le but de ce travail était d'évaluer si la convolution de réseaux de neurones pouvait être appliquer avec succès à la classification judicieuse des environnements sonores, en particulier compte tenu des natures des ensembles de données disponibles dans ce domaine. Il semble qu'ils constituent une solution durable à ce problème. Les expériences menées montrent qu'un modèle convolution surpasse les approches courantes basées sur des fonctionnalités former manuellement, et atteint un niveau similaire à celui des autres méthodes d'apprentissage des fonctionnalités.

Notre travail montre que les réseaux de neurones convolutifs peuvent être appliqués efficacement dans la classification environnementale sonore (ESC), même avec des ensembles de données limités. Bien que, en tenant compte des temps de formation beaucoup plus longs, le résultat est loin d'être révolutionnaire.

Bibliographies

[EDAOSED] H. YKHLEF, F. YKHLEF and S. CHIBOUB, “Experimental Design and Analysis of Sound Event Detection Systems: Case Studies”, 6th International Conference on Image and Signal Processing and their Applications, ISPA 2019, 24-25, Nov 2019, Mostaghanem, Algeria.

[TSNOPC] Stéphane Gautier, Arnaud Margollé, (avril2020). Traitement du signal numérique optique, Photométrie. Colorimétrie, Bibliothèque nationale, paris

[CSSST] Maxime Baelde. Modèles génératifs pour la classification et la séparation de sources sonores en temps réel. Méthodologie [stat.ME]. Université de Lille 1, 2019. Français.fftel-02399081f

[SDACSE] Asma Rabaoui, Manuel Davy, Stéphane Rossignol, Zied Lachiri, Noureddine Ellouze. 11-14 septembre 2007. Sélection de descripteurs audio pour la classification des sons environnementaux avec des SVMs mono-classe. Unité de recherche Signal, Image et Reconnaissance des formes, ENIT, BP 37, Campus Universitaire, 1002 le Belvédère, Tunis Tunisie T´el : + (216) 71 87 68 00 – Fax : + (216) 71 87 68 00. CNRS/Laboratoire d’Automatique, de Génie Informatique et Signal INRIA Futur équipe SequeL, BP 48, 59651 Villeneuve d’Ascq cedex, France T´el : 03 20 67 60 13 – Fax : 03 20 33 54 18

[CASCFL] Joachim Flocon-Cholet. Classification audio sous contrainte de faible latence. Traitement du signal et de l’image [eess.SP]. Université Rennes 1, 2016. Français. ffNNT: 2016REN1S030ff. fftel-01395495f

[GPEE] Geoffroy Peeters: A large set of audio features for sound description (similarity and classification) in the cuidado project. Rapport technique, IRCAM, 2004.

[ASIBF] Jérôme Gauthier. Analyse de signaux et d'images par bancs de filtres : applications aux géosciences. Autre. Université Paris-Est, 2008. Français. ffNNT : 2008PEST0211ff.fftel-01128245f

[ZCODA] Zegadi Cherifa. (26/10/2011). Optimisation des descripteurs audio par l'analyse en composantes indépendantes, université de science et la technologies houari Boumediene.

[CASAF] Slim Essid. Classification automatique des signaux audio-fréquences : reconnaissance des instruments de musique. Traitement du signal et de l'image [eess.SP]. Université Pierre et Marie Curie - Paris VI, 2005. Français.ffpastel-00002738f

[ADESA] Sébastien LECOMTE. (Le 9 décembre 2013). Classification partiellement supervisée par SVM. Application à la détection d'évènements en surveillance audio. DOCTEUR de l'UNIVERSITE DE TECHNOLOGIE DE TROYES Spécialité : OPTIMISATION ET SURETE DES SYSTEMEs.

[MAEPTIM] : Kévin Bailly. Méthodes d'apprentissage pour l'estimation de la pose de la tête dans des images monoculaires. Interface homme-machine [cs.HC]. Université Pierre et Marie Curie - Paris VI, 2010. Français. fftel-00560836f

[ADBASOF] Songyot Nakariyakul et David P. Casasent: Adaptive branch and bound algorithm for selecting optimal features. Pattern Recognition Letters, 28: 1415–1427, 2007.

[FBAOFS] Petr Somol et Pavel Pudil: Fast branch & bound algorithms for optimal feature selection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(7):900–912, 2004.

[FSMFSNCF] P. Pudil, F.J. Ferri, J. Novovicova et Kittler J.Floating search methods for feature selection with nonmonotonic criterion functions. Proceedings of the 12th IAPR International Conference on Pattern Recognition, 2:279–283, 1994.

[CTCAS] Rémy Kesler, Marc El-Bèze, Juan-Manuel Torres-Moreno. (25-29 octobre 2004), CLASSIFICATION THÉMATIQUE DE COURRIELS AVEC APPRENTISSAGE SUPERVISÉ, SEMI

SUPERVISÉ.ET NON SUPERVISÉ. Laboratoire d'Informatique d'Avignon - Université d'Avignon et des Pays de Vaucluse 339 chemin des Meinajaries, Agroparc, BP 1228, 84911 Avignon Cedex 9, France.Tel. : +33 (0) 4 90 84 35 09

[AS] GUILLAUME SAINT-CIRGUE. (Juillet 2, 2019), Apprentissage Supervisé : Introduction, Londres

[MLIAA] Metomo JOSEPH BERTRAND RAPHAËL. (10/10/2017), Machine Learning : Introduction à l'apprentissage automatique

[ADULL] Ricco RAKOTOMALALA. (2005), Arbres de Décision, Laboratoire ERIC. Université Lumière Lyon 25, av. Mendés France

[PASUW] Hadj-Tayeb Karima, Approche de partitionnement pour un apprentissagenon supervisé des Usagers du Web (Amélioration del'approche k-means). Département d'Informatique Université des sciences et de la technologie d'Oran, Mohamed Boudiaf (USTO) Oran, AlgérieJded f cnn

[IECNN] Purit Punyawiwat,Natchuta Wattanapenpaiboon.(Feb 8, 2018),Interns Explain CNN

[CNNPP] Pawan Reddy Ulindala.(May 7.2020),Convolutional Neural Networks(CNN's) — A practical perspective

[TCBGDL] Anne Bonner. (Feb 2, 2019), The Complete Beginner's Guide to Deep Learning: Convolutional Neural Networks and Image Classification

[ICCNN] Farhana Sultana,A.Sufian.(November 2018),image classification using CNN.University of Gour Banga, India

[6TANN] Kishan Maladkar. (15/01/2018),6 Types of Artificial Neural Networks Currently Being Used in Machine Learning

[ANNML] Sheetal Sharma. (August 8, 2017), Artificial Neural Network (ANN) in Machine Learning

[ANNBDL] John and Willie Leone. (19 sep 2014). Artificial neural network-based design for dual lateral well applications, Department of Energy and Mineral Engineering, Pennsylvania State University, 118 Hosler Building, 814-865-6082 PA, USA article

[MRCCNNA] Muhammad Rizwan. (30 septembre 2018), LeNet-5 – A Classic CNN Architecture.

[RKICNNA] Raimi Karim. (Jul 29, 2019), Illustrated: 10 CNN Architectures

[SHAR] Sik-Ho Tsang. (Aug 8, 2018), Review: LeNet-1, LeNet-4, LeNet-5, Boosted LeNet-4 (Image Classification).

[RAIAN] Richmond Alake. (aug 14, 2020), Implementing AlexNet CNN Architecture Using TensorFlow 2.0+ and Keras

[SHTRIC] Sik-ho Tsang. (aug 19,2018), Review:ZFN__ winner of ILSVRC 2013(image classification)

[PCNNA] Prabhu. (Mar 15, 2018), CNN Architectures — LeNet, AlexNet, VGG, GoogLeNet and ResNet

[CNNA] siddharth das. (nov16,2017), CNN architectures:LeNet,VGG,GoogleNet,ResNet and more...

[AFINN] SAGAR SHARMA.(Sep 6, 2017). Activation Functions in Neural Networks.

[SOVAUUACR] Bachu R.G., Kopparthi S., Adapa B., Barkana B.D.Separation of Voiced and Unvoiced using Zero crossing rate and Energy of the Speech Signal,Electrical Engineering Department School of Engineering, University of Bridgeport.

[CNNDESC] Justin Salamon, Juan Pablo Bello. (Novembre 2016), Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification, IEEE SIGNAL PROCESSING LETTERS.

[DBBAE] Jason Brownlee, (July 20, 2018). in Deep Learning, Difference Between a Batch and an Epoch in a Neural Network

Webographie

[ACKPP] https://boowiki.info/art/algorithmes-de-classification/k-voisins-les-plus-proches

[KPPV] https://info.blaisepascal.fr/nsi-les-k-plus-proches-voisins

[LDAS] https://le-datascientist.fr/5-apprentissage supervise

[SIUPMFCC] https://www.mathworks.com/help/audio/examples/speaker-identification-using-pitch-and-mfcc

[ANNCK] http://www.cs.kumamoto-u.ac.jp/epslab/ICinPS/Lecture-2.pdf

[CTCNNP] https://www.datacamp.com/community/tutorials/convolutional-neural-networks-python

[FITING] https://www.datarobot.com/wiki/fitting/

[DMCOWB] https://stackoverflow.com/questions/47995324/does-model-compile-initialize-all-the-weights-and-biases-in-keras-tensorflow

[KAM] https://keras.io/api/metrics/

[MLTAL] https://developers.google.com/machine-learning/crash-course/descending-into-ml/training-and-loss

[US8K] https://urbansounddataset.weebly.com/urbansound8k

[GTKTMS] https://towardsdatascience.com/getting-to-know-the-mel-spectrogram

[MFCCT] http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/#computing-the-mel-filterbank

[SPFML] https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html

[IMEEM] https://www.datasciencecentral.com/profiles/blogs/7-important-model-evaluation-error-metrics-everyone-should-know

[LEVOHE] https://medium.com/@contactsunny/label-encoder-vs-one-hot-encoder-in-machine-learning-3fc273365621

[PDC]https://python.developpez.com/cours/DiveIntoPython/php/frdiveintopython/native_data_types/tuples

[AFFNN] https://www.codespeedy.com/activation-function-for-neural-network/

[WITDCNN] https://ai.stackexchange.com/questions/5546/what-is-the-difference-between-a-convolutional-neural-network-and-a-regular-neur

[ANN] https://www.ntirawen.com/2018/12/artificial-neural-network-ann.html

[ANNATT] https://www.elprocus.com/artificial-neural-networks-ann-and-their-types/

[BPNN] https://www.guru99.com/backpropogation-neural-network.html

[GUCNN] https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/

[MUTRN] https://www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-m-app-rn.pdf

[KGCS] https://kongakura.fr/article/Classification

[ASVNS] https://le-datascientist.fr/apprentissage-supervise-vs-non-supervise

[DBSUS] https://fr.sawakinome.com/articles/programming/difference-between-supervised-and-unsupervised-machine-learning

[ICDLPPK] https://data-flair.training/blogs/image-classification-deep-learning-project-python-keras/

[CNNIQ] https://iq.opengenus.org/convolutional-neural-networks/

[CSDDV] https://openclassrooms.com/fr/courses/4470531-classez-et-segmentez-des-donnees-visuelles/5082166-quest-ce-quun-reseau-de-neurones-convolutif-ou-cnn

[WDKR] http://we.dosaaf-kvl.ru/151