République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Normale Supérieure d'Enseignement Technologique

Département : Mathématique et Informatique

Spécialité : Informatique

Mémoire de fin d'étude En vue de l'obtention du diplôme : Professeur d'Enseignement Moyenne

Classification des Sons environnementaux en utilisant des techniques de Deep Learning

Présenté par

Remadnia Roumaissa Maghnia

Khireddine Marwa

Encadré Par :

Mme. Abdoune Leila

Jury de Soutenance

Dr. Mezghache Reda	Enset-Skikda	Président
Dr. Salah Halima	Enset-Skikda	Examinateur
Dr.Bekhoche Safia	Enset-Skikda	Examinateur

Année Universitaire : 2024/2025

Dédicace

Je dédie ce mémoire à ma chère mère, qui a tant sacrifié pour moi. Son courage, sa force et son amour inconditionnel ont toujours été une source d'inspiration. Ce travail est aussi le fruit de ses efforts et de ses prières silencieuses.

À mon père, qui est toujours présent à mes côtés, me soutenant et m'encourageant dans chaque étape de ma vie. Son appui moral et ses conseils ont été essentiels dans mon parcours.

À ma seule et unique sœur, que j'aime profondément. Ta présence, ton affection et ta complicité me sont précieuses dans ce monde.

Et enfin, à mes amis, pour leur soutien, leurs encouragements et leur amitié sincère tout au long de cette aventure.

Remadnia Roumaissa Maghnia

Dédicace

Je rends grâce à Dieu, le Tout-Puissant, pour m'avoir accordé la force, la patience et la persévérance tout au long de ce parcours.

À ma mère bien-aimée,
pour son amour inconditionnel, ses prières silencieuses et sa tendresse infinie qui ont toujours été ma plus grande source de force.

À mon père cher,
pour son soutien constant, ses sacrifices et ses mots simples qui m'ont portée dans les moments de doute.

À mes frères et sœurs,
pour leur affection sincère et leur présence rassurante, même dans le silence.

À mon encadrant,
pour sa disponibilité, ses conseils précieux et sa confiance, qui ont grandement enrichi ce travail.

À mon binôme et amie Roumaissa,
pour sa coopération, sa motivation et les efforts partagés avec respect et complicité.

À tous ceux qui ont cru en moi, de près ou de loin,
je dédie ce modeste travail avec tout mon respect et ma profonde reconnaissance.

Khireddine Marwa

Remerciements

Tous mes Remerciement s'adressant tout d'abord à tout puissant ALLAH, d'avoir guidé mes pas vers le chemin du savoir.

Je tiens à adresser mes plus sincères remerciements à toutes les personnes qui m'ont aidé et soutenu dans la réalisation de ce mémoire.

Tout d'abord, un grand merci à mon directeur de mémoire Madame **Abdoune Leila**, pour sa disponibilité, ses conseils judicieux et son accompagnement tout au long de ce travail. Sa rigueur et sa bienveillance ont été essentielles à l'aboutissement de ce projet.
Je remercie également mes collègues et amis pour leur soutien moral et leurs encouragements, qui m'ont permis de rester motivé et concentré, même dans les moments les plus difficiles.

Je tiens à exprimer ma profonde gratitude aux membres du jury**, Dr. MezghacheReda**, **Dr. Salah Halima** et **Dr. Bekhouche Safia**, pour le temps qu'ils ont consacré à l'évaluation de mon travail, pour leurs remarques constructives, ainsi que pour leur bienveillance tout au long de cette étape importante.

Leur expertise et leurs conseils ont grandement contribué à enrichir cette recherche et à m'encourager dans mon parcours académique.

Je n'oublie pas mafamille, qui a toujours cru en moi et m'a soutenu avec beaucoup d'amour et de patience. Leur soutien inconditionnel a été un véritable moteur pendant cette aventure.

Enfin, merci à tous ceux, proches ou moins proches, qui ont contribué de près ou de loin à la réussite de ce travail.

Résumé

L'objectif principal de ce mémoire est de construire un modèle de classification basé sur les techniques d'apprentissage automatique, Nous utiliserons à cette effet un modèle de réseaux de neurones convolutifs (CNN) pour identifier et classer correctement des signaux sonores appartenant à la base de données ESC50, cette base de données contient 50 classe de sons et chaque classe contient un nombre de fichier sonores.

Dans ce travail on mettra en évidence l'efficacité des modèles à base de CNN dans la reconnaissance et la classification des sons à partir de leur représentant spectral ce qui sera L'objectif de notre partie application.

Abstract:

The main objective of This report is to build a classification model based on machine learning techniques. To this end, we will use a convolutional neural network (CNN) model to correctly identify and classify audio signals belonging to the ESC50 database. This database contains 50 classes of sounds, and each class contains a number of audio files.

In this work, we will highlight the effectiveness of CNN-based models in the recognition and classification of sounds from their spectral representation, which will be the objective of our application section.

ملخص:

الهدف الرئيسي من هذه الرسالة هو بناء نموذج تصنيف قائم على تقنيات التعلم الآلي. ولتحقيق ذلك، سنستخدم نموذج الشبكة العصبية التلافيفية (CNN) لتحديد وتصنيف الإشارات الصوتية بدقة من قاعدة بياناتCSc50.تحتوي هذه القاعدة على 50 فئة صوتية، يحتوي كل منها على عدد من ملفاتالصوت.

في هذا العمل، سنسلط الضوء على فعالية النماذج القائمة على CNN في التعرف على الأصوات وتصنيفها بناءً على تمثيلها الطيفي، وهو ما سيُشكل هدف قسم التطبيق لدينا.

Sommaire

Chapitre 4 : Implémentation

Liste des acronymes

Abréviation	**Signification**
AA	apprentissage automatique
CNN	Convolution Neural Network
DL	deep Learning
GMM	Gaussien Mixture Model
RNN	Récurrentneural network
SVM	Support Vector Machine
HMM	Hidden Marcov Models
MFCC	Mel-frequency cepstral coefficients
STFT	Short-time fourier transform
DT	Arbre de décision
NB	Naïve Baye sienne (NB).
ANN	Réseau neuronal artificiel
LR	Régression logistique
KNN	K-Nearest Neighbors
IA	Intelligence Artificielle
LM	Learning Machine
SLR	Régression Linéaire Simple
DTR	Régression de L’arbre de décision
DTFR	Régression de foret d’arbres désicionnels

Liste des Figures

Liste des tableaux

Introduction

Dans un monde de plus en plus connecté, les systèmes capables d'interpréter les signaux sonores occupent une place centrale dans de nombreuses applications telles que la surveillance intelligente, les assistants vocaux, la domotique, ou encore le diagnostic médical. La classification automatique des sons consiste à identifier, à partir d'un enregistrement audio, la catégorie ou l'événement sonore auquel il appartient (ex. : aboiement de chien, sirène, parole humaine). Cette tâche complexe exige une compréhension fine des caractéristiques acoustiques, souvent non linéaires et fortement variables.

L'émergence du deep Learning a profondément transformé ce domaine, en permettant la modélisation automatique de représentations pertinentes à partir des données brutes, surpassant largement les méthodes traditionnelles fondées sur des descripteurs manuellement définis.

Problématique

Malgré les avancées notables des méthodes d'apprentissage profond, la classification des sons reste un défi technique en raison de la diversité des environnements sonores, de la complexité des signaux audio, et du bruit ambiant. Comment concevoir un modèle de deep Learning efficace capable de distinguer avec précision différents types de sons, à partir de représentations audio pertinentes (comme les spectrogrammes, les MFCCs ou les Mel-Spectrograms) ? Quel type de représentation est le plus adapté pour améliorer la performance d'un réseau de neurones convolutifs dans un contexte de classification audio ?

Motivation

Ce travail s'inscrit dans la volonté d'exploiter la puissance du deep Learning, en particulier les réseaux de neurones convolutifs (CNN), pour résoudre un problème à fort impact pratique : la classification automatique des sons. Le développement d'un tel système pourrait contribuer à améliorer des domaines variés comme la sécurité, la santé, ou l'interaction homme-machine. De plus, cette étude permet d'approfondir les compétences en traitement du signal, en conception de modèles d'apprentissage automatique, et en évaluation de performances sur des données réelles.

Organisation du document

Le premier chapitre de ce mémoire explore les fondements de la classification des sons. Il s'attarde sur la définition des sons, les différentes catégories sonores et l'importance de leur reconnaissance automatique. Ce chapitre met également en lumière les défis techniques liés à cette tâche, tels que la variabilité des sources sonores, le bruit de fond ou encore la complexité des signaux audio. Il examine aussi les applications concrètes de cette technologie dans des secteurs variés comme la sécurité, la domotique, ou encore la surveillance environnementale. Enfin, cechapitre

introduit les principales étapes du processus de classification sonore, depuis la capture du signal audio jusqu'à l'interprétation du résultat par un système intelligent.

Le deuxième chapitre présente les bases de l'apprentissage automatique (machine learning), en détaillant ses phases principales, ses différents types (supervisé, non supervisé, par renforcement, etc.) et les méthodes classiques utilisées dans ce domaine. Cette partie prépare le terrain pour l'introduction de l'apprentissage profond (deeplearning), une branche de l'intelligence artificielle qui a révolutionné la reconnaissance de patterns complexes, notamment dans les signaux audio. Elle expose les principes fondamentaux du deeplearning, avec une attention particulière portée aux réseaux de neurones artificiels, et notamment à deux architectures essentielles dans le traitement des sons : les réseaux de neurones convolutifs (CNN) et les réseaux de neurones récurrents (RNN).

Les troisième et quatrième chapitres sont dédiés à la conception du système de classification sonore et à son implémentation pratique, respectivement. On y présente l'architecture proposée, les choix méthodologiques, les outils utilisés, ainsi que les résultats expérimentaux obtenus et leur évaluation.

Ainsi, ce mémoire propose une étude complète de la classification des sons environnementaux, en combinant une base théorique solide avec une démarche pratique basée sur les dernières avancées en deeplearning.

Chapitre 1

Classification des Sons

Introduction

Capturer, détecter, identifier les sons et alerter les utilisateurs sous forme de notifications visuelles et de vibrations est un concept connu sous le nom de reconnaissance sonore environnementale. Par conséquent, comprendre ou reconnaître le contexte des sons dans l'environnement est très important pour pouvoir agir sur le son qui se produit. Par exemple, évacuer un bâtiment lorsque vous entendez une alarme incendie ou prendre soin d'un enfant lorsque vous entendez des cris. De telles activités s'appuient sur l'audition humaine, qui filtre intelligemment et reconnaît rapidement les sons, envoyant des signaux au cerveau pour passer à l'étape suivante [26].

Dans ce chapitre, nous explorerons les concepts de base de la classification des sons et soulignerons son importance croissante dans divers domaines. Nous évoquerons les défis auxquels nous sommes parfois confrontés dans la reconnaissance des sons, en plus de ses nombreuses applications pratiques, comme la surveillance des sons. Diagnostic de santé, le transport et domotique. Enfin, nous détaillerons les principales étapes qui composent le processus de classification des sons.

1.1. Contexte de la classification des Sons

La classification des sons est un domaine de recherche en plein essor qui combine les techniques de traitement du signal acoustique avec les avancées de l'intelligence artificielle. Son objectif principal est de permettre aux machines de reconnaître, classer et interpréter les sons en fonction de leurs caractéristiques acoustiques. Ces sons proviennent de diverses sources, naturelles et artificielles, des informations essentielles sur notre environnement. Cependant, l'automatisation de cette classification reste un défi technique majeur en raison de la variété des sources sonores et des conditions d'enregistrement complexes.

Ce domaine est essentiel dans de nombreuses disciplines telles que la linguistique, la phonétique, l'acoustique, la musique et la psychologie cognitive, car il permet de mieux comprendre les phénomènes sonores en fonction de leur nature, de leur origine et de leurs caractéristiques. Ainsi, la classification de son joue un rôle clé en fournissant des outils permettant d'analyser et d'organiser les sons dans différents environnements, qu'ils soient naturels, industriels ou domestiques. Grâce aux avancées technologiques, les méthodes de classification des sons sont désormais appliquées dans de nombreux secteurs, tels que la surveillance acoustique dans les systèmes de sécurité, les diagnostics de santé basés sur l'analyse du bruit corporel, la détection de problèmes mécaniques ou environnementaux dans les transports, et même dans la domotique pour améliorer la qualité de vie.

Les recherches actuelles continuent d'explorer des approches novatrices pour surmonter les défis posés par la diversité et la complexité des environnements sains.

1.2. La reconnaissance des sons

Dans cette section, nous discuterons de ce qu'est la reconnaissance de son, en commençant par identifier ce qu'est un son, et en mentionnant les types qui existent sois selon la fréquence ou la source.

1.2.1. Définition du son

Le son est défini comme une sensation auditive résultant de la vibration d'un corps, qui se propage sous forme d'ondes acoustiques dans un milieu élastique, tel que l'air ou l'eau. Pour qu'un son existe, trois éléments sont nécessaires

- Une source sonore : Cela peut être un objet solide, liquide ou gazeux qui vibre.
- Un milieu de propagation : Les vibrations doivent se transmettre à travers un milieu (comme l'air) ; elles ne peuvent pas se propager dans le vide.
- Un récepteur : L'oreille humaine ou animale qui perçoit ces vibrations [15][16][17].

1.2.2. Catégories dessons selon la fréquence

On distingue la première catégorie des sonsqui est basée sur leur fréquence, qui est essentielle pour comprendre comment ils sont perçus par l'oreille humaine. Voici les principales catégories :

1) **Infrasons :** Les infrasons sont des sons dont la fréquence est inférieure à 20 Hz, en dehors de la plage d'audition humaine [1][2].
   Ils sont généralement ressentis plutôt qu'entendus, souvent associés à des phénomènes naturels comme les tremblements de terre ou le tonnerre [1][4].
   Les infrasons peuvent parcourir de grandes distances et sont parfois utilisés par certains animaux pour la communication sur de longues distances, comme les éléphants [1][6].
2) **Sons Audibles :** Les sons audibles se situent dans la plage de 20 Hz à 20 kHz (20 000 Hz), qui est la plage que l'oreille humaine peut percevoir [5][8].Cette catégorie inclut la majorité des sons de la parole et de la musique.
   La perception des sons varie en fonction de leur fréquence: plus la fréquence estélevée, plus le son est aigu, tandis que les fréquences basses produisent des sons graves [5][7].
3) **Ultrasons:** Les ultrasons sont des sons dont la fréquence est supérieure à 20kHz[1][8]. .
   Ils ne sont pas audibles pour Homme, mais peuvent être perçus par certains animaux comme les chauves-souris et les dauphins [1][6] .
   Les ultrasons sont largement utilisés dans diverses applications, notamment enéchographie médicale et dans certains dispositifs de nettoyage[1].

1.2.3. Catégories des sons selon la source

Il existe aussi trois principales catégories de sons, classées en fonction de leur source. Comme nous l'avons expliqué précédemment, il existe diverses sources naturelles et artificielles, qui reflètent la richesse du monde sonore. Pour approfondir notre compréhension de ce domaine, il est important de reconnaître ces trois catégories principales qui forment la base des sons :

1. Sons purs : Ce sont des sons "idéaux", caractérisés par une seule fréquence. On les rencontre rarement dans la nature, mais ils sont utilisés en acoustique et en musique électronique comme base de création sonore. Imaginez une flûte qui émet une seule note claire et continue : c'est un exemple de son pur.

2. Les sons complexes : La plupart des sons que nous percevons au quotidien sont des sons complexes. Ils sont composés de plusieurs fréquences qui se superposent. La voix humaine, le chant des oiseaux, le bruit d'une voiture sont autant d'exemples de sons complexes. Chaque son complexe possède une "couleur" unique, appelée timbre, qui permet de le différencier des autres.

3. Bruit : Le bruit est un son complexe, mais il se distingue par son caractère aléatoire et désordonné. Il est souvent perçu comme indésirable ou gênant. Le bruit du trafic routier, le brouhaha d'une foule, le grincement d'une porte sont autant d'exemples de bruits [25].

1.3. Importance de la classification des sons

- **Comprendre la langue et la communication :** La classification des sons permet de comprendre comment les langues fonctionnent. En analysant les phonèmes, on peut expliquer les similarités et différences entre les langues. Par exemple, le « r » final en anglais britannique est souvent omis (prononcé [car]), contrairement à l'anglais américain où il est prononcé clairement. Ces variations phonétiques influencent la compréhension entre locuteurs [9].

- **Apprendre une langue :** Pour maîtriser une langue étrangère, il est crucial de comprendre ses sons spécifiques. Chaque langue a des phonèmes uniques qui peuvent ne pas exister dans la langue maternelle de l'apprenant. La classification des sons aide les enseignants à concevoir des exercices pour améliorer la reconnaissance et la production de ces sons, réduisant ainsi les erreurs de prononciation. Par exemple, le son [θ] (comme dans "think") n'existe pas en français, ce qui pousse souvent les francophones à le remplacer par [s] ou [t] [10].

- **Applications technologiques et médicales :** Dans le domaine médical, la classification des sons est utilisée pour analyser des signaux audio complexes,

comme les bruits corporels pour détecter des anomalies. Par exemple, un stéthoscope électronique peut enregistrer les sons pulmonaires et utiliser l'intelligence artificielle pour identifier rapidement des problèmes tels que des fissures associées à une pneumonie3. En ingénierie, cette classification permet également d'analyser le bruit des machines pour prédire des dysfonctionnements [11].

- **Performance musicale et créativité :** En musique, les sons sont classés selon leur hauteur, durée, timbre et intensité. Cette classification permet aux compositeurs d'utiliser les propriétés sonores pour créer et interpréter des œuvres musicales. Par exemple, un compositeur peut utiliser un logiciel comme Ableton Live pour analyser le spectre sonore d'un instrument afin de créer une harmonie dans une composition[3][12].

- **Identifier les sons dans l'environnement :** La classification des sons environnementaux permet de reconnaître et de comprendre les signaux acoustiques naturels ou artificiels. Cela est particulièrement utile pour les études écologiques et la surveillance des écosystèmes. Par exemple, les écologistes utilisent des logiciels comme Raven Pro pour analyser les chants des oiseaux et identifier des espèces spécifiques d'une forêt tropicale5. Cette méthode aide à surveiller les populations d'espèces menacées et à évaluer la biodiversité [13].

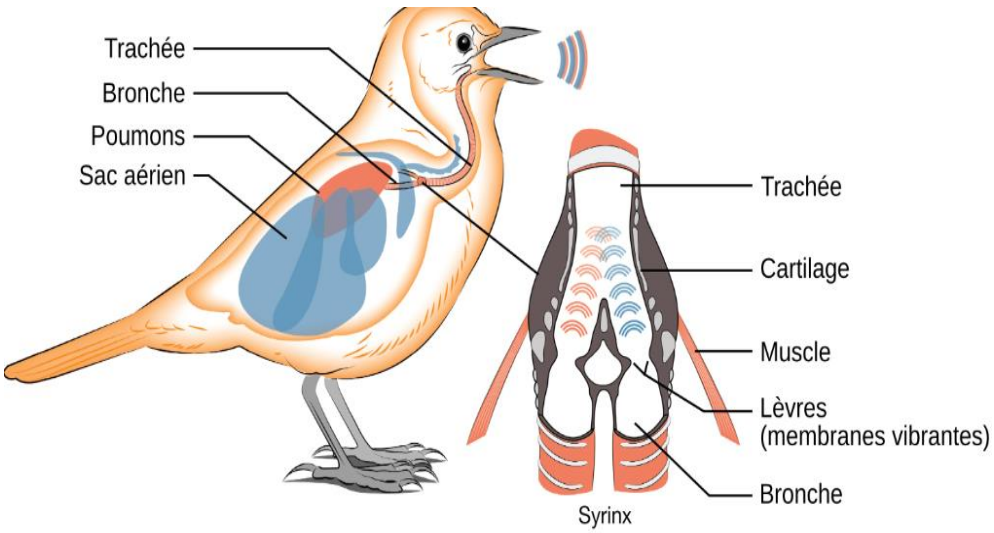


Figure1. 1 :Position et anatomie de la syrinx des oiseaux chanteurs [44].

- **Reconnaissance et traitement automatique du son :** Les technologies modernes reposent sur la reconnaissance vocale et le traitement automatique du son grâce à des algorithmes capables de classer et de traiter des signaux audio complexes. Ces systèmes permettent de distinguer les voix humaines

dans des environnements bruyants et comprennent les commandes vocales. Par exemple, les assistants vocaux comme Alexa et Google Assistant utilisent des modèles d'intelligence artificielle pour distinguer la voix de l'utilisateur du bruit de fond6. Cette capacité permet une réponse précise aux commandes même dans un environnement bruyant [14].

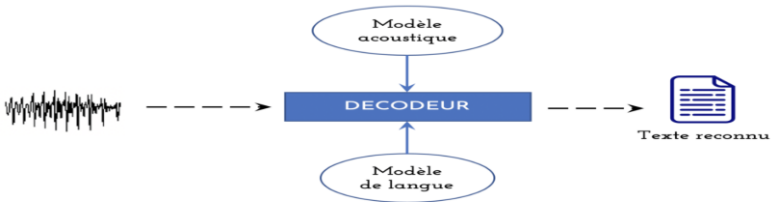


Figure1. 2 : Architecture classique (et hybride) d'un Système de Reconnaissance Automatique de la Parole (SRAP)[43].

1.4. Les principaux défis

L'application de classification des sons environnementaux présente plusieurs défis et exigences techniques. Ces systèmes visent à identifier, analyser et catégoriser différents types de sons présents dans l'environnement, comme le bruit urbain, les sons naturels ou industriels, pour des usages comme la surveillance sonore, l'amélioration de la qualité de vie, ou la recherche scientifique.

Variabilité des sons : La variabilité des sons environnementaux pose un défi majeur pour la classification automatique. Les sons tels que le bruit d'une rue animée, le chant des oiseaux et le bruit des vagues diffèrent considérablement en fréquence, intensité et durée, ce qui complique leur classification précise [28], [29], [30].

Bruitage et interférence : Les enregistrements sonores peuvent être affectés par des bruits de fond, comme le vent ou les véhicules, rendant l'analyse plus complexe. Il est essentiel de distinguer les sons d'intérêt du bruit de fond pour une classification efficace [28] [30].

Complexité des données acoustiques : Les données acoustiques sont souvent complexes et nécessitent des algorithmes avancés pour extraire des caractéristiques significatives. Ces caractéristiques doivent être robustes pour s'adapter à divers environnements et contextes [29][30].

Étiquetage des données : La collecte et l'étiquetage des données pour l'entraînement des modèles d'apprentissage automatique peuvent être coûteuses et longues. Les sons doivent être classifiés manuellement, ce qui nécessite une expertise et une bonne connaissance des différents types de sons environnementaux [30].

Problèmes d'échelle : Les environnements dynamiques posent un défi pour maintenir la précision des modèles à travers différents contextes géographiques et

temporels. Les recherches actuelles explorent l'adaptabilité des modèles à ces variations grâce à l'utilisation de techniques comme le transfert d'apprentissage et l'optimisation des plans d'échantillonnage[28] [29].

1.5. Les applications pratiques

La classification des sons est un domaine de recherche en pleine croissance qui combine des techniques de traitement du signal audio avec des avancées en intelligence artificielle. Son objectif principal est de permettre aux machines de reconnaître, classer et interpréter les sons en fonction de leurs propriétés acoustiques. Ce domaine est fondamental pour de nombreuses disciplines telles que la linguistique, la phonétique, l'acoustique, la musique et la psychologie cognitive, car il permet de mieux comprendre les phénomènes acoustiques selon leur nature, leur origine et leurs propriétés. La classification sonore joue donc un rôle fondamental en fournissant les outils nécessaires pour analyser et organiser les sons dans différents environnements, qu'ils soient naturels, industriels ou domestiques. Grâce aux progrès technologiques, les méthodes de classification sonore sont désormais appliquées dans de nombreux secteurs tels que la surveillance acoustique, transport, santé et domotique. Ces applications se distinguent par leurs objectifs spécifiques.

Tbleau1. 1 :**Les domaines de la classification des sons.**

Domaine	Nom du projet	Objectif
Surveillance Acoustique	Détection, classification et localisation d'événements acoustiques en présence de bruit de fond pour la surveillance acoustique de situations dangereuses[57].	– *Détecter* des événements sonores d'intérêt dans des environnements bruyants. – *Classifier* ces événements pour identifier leur nature (e.g. accident, intrusion). – *Localiser* avec précision les sources sonores pour une réponse rapide dans des situations d'urgence
Santé	Classification automatique des sons respiratoires à l'aide de réseaux de neurones convolutifs [56].	– *Améliorer le diagnostic médical* en fournissant un outil précis pour détecter les maladies respiratoires, telles que l'asthme ou la broncho-pneumopathie chronique obstructive. – *Faciliter la surveillance à distance* des patients avec des solutions basées sur l'IoT.
Transport	Système intelligent de détection précoce des pannes de véhicules	– *Interaction naturelle* : Utiliser des microphones pour capter les sons du véhicule sans intervention intrusive,

	basé sur les sons [55].	permettant une surveillance continue et en temps réel. – *Sécurité renforcée* : Identifier rapidement les anomalies sonores indiquant des problèmes mécaniques, afin de prévenir les pannes potentielles et d'assurer la sécurité des passagers.
Domotique	Reconnaissance des sons de l'environnement dans un contexte domotique [35]	– **Favoriser une interaction naturelle** entre les utilisateurs et leur environnement. – **Renforcer la sécurité** en détectant les situations de détresse. Les technologies audio ont constitué le cœur du projet, avec deux axes de recherche principaux : ➢ **Reconnaissance des sons de l'environnement** pour identifier les bruits spécifiques au sein d'une maison intelligente. ➢ **Reconnaissance vocale adaptée aux personnes âgées**, prenant en compte leurs besoins particuliers].

1.6. Étapes typiques de la classification des sons

La classification des sons suit un processus structuré qui comprend plusieurs étapes, depuis la détermination de leur nature jusqu'à leur répartition en catégories distinctes. Voici une méthode détaillée pour organiser les sons :

1. Acquisition du signal : L'acquisition du signal est la première étape cruciale dans le traitement des sons. Elle consiste à enregistrer l'audio à l'aide de capteurs tels que des microphones. Cette étape permet d'obtenir un signal audio brut, qui est une représentation numérique des ondes sonores captées. Les microphones convertissent les variations de pression de l'air en signaux électriques, qui sont ensuite numérisés pour une utilisation ultérieure. La qualité de cette acquisition est déterminante pour les étapes suivantes, car un signal de mauvaise qualité peut introduire des erreurs dans le traitement [18].

2. Prétraitement : Le prétraitement vise à améliorer la qualité du signal audio en réduisant le bruit et en éliminant les artefacts indésirables. Cette étape comprend plusieurs techniques :

- **Filtrage :** Des filtres numériques peuvent être appliqués pour atténuer les fréquences indésirables, comme le bruit de fond, ce qui permet de clarifier le signal principal.
- **Échantillonnage :** L'échantillonnage consiste à convertir le signal analogique en un format numérique en prenant des mesures à intervalles réguliers. La fréquence d'échantillonnage doit être suffisamment élevée pour capturer toutes les nuances du son.

- **Normalisation :** Cette technique ajuste le niveau du signal pour qu'il soit uniforme, ce qui facilite les comparaisons ultérieures.

Ces opérations garantissent que le signal est propre et prêt à être analysé dans les étapes suivantes [18].

3. Extraction de caractéristiques :L'extraction de caractéristiques est une étape essentielle où les données audio sont transformées en représentations numériques exploitables. Cette phase permet de condenser l'information tout en préservant les éléments significatifs du signal. Les techniques courantes incluent :

Spectrogrammes : Ces représentations visuelles montrent comment les différentes fréquences d'un son évoluent dans le temps, permettant de visualiser les composantes fréquentielles.

Coefficients cepstraux en fréquence Mel (MFCC) : Les MFCC sont largement utilisés dans la reconnaissance vocale et la classification des sons, car ils capturent les caractéristiques perceptuelles essentielles du son tout en réduisant la dimensionnalité des données.

L'objectif est d'extraire des caractéristiques qui seront significatives pour la classification, facilitant ainsi l'identification des différents types de sons [18][19].

4. Apprentissage et classification : La dernière étape consiste à utiliser des modèles d'apprentissage automatique pour classer les sons selon des catégories prédéfinies. Ce processus implique :

- **Entraînement des modèles :** Les algorithmes d'apprentissage automatique (comme les réseaux de neurones, les machines à vecteurs de support ou les forêts aléatoires) sont entraînés sur un ensemble de données contenant des exemples étiquetés. Le modèle apprend à reconnaître les patterns associés à chaque catégorie sonore.
- **Validation et test :** Une fois le modèle entraîné, il est testé sur un ensemble de données distinct pour évaluer sa performance. Cela permet de s'assurer que le modèle peut généraliser ses connaissances à de nouveaux échantillons audio [18].

Ces étapes acquisition du signal, prétraitement, extraction de caractéristiques, apprentissage et classification. Constituent le processus global de classification des sons. Chaque étape est essentielle pour garantir que le système puisse identifier et classer efficacement différents types de sons, qu'il s'agisse de musique, de parole ou d'autres bruits environnementaux. Ce processus garantit une classification méthodique et cohérente, en s'appuyant sur divers outils, qu'il s'agisse de dispositifs physiques ou de logiciels spécialisés.

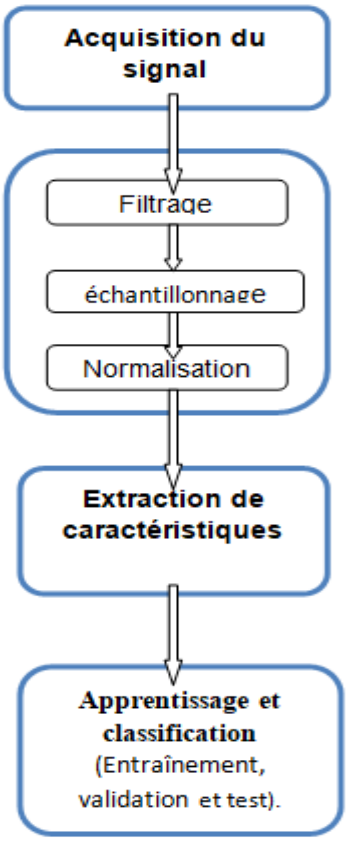


Figure1. 3 :les étapes de classification des sons

.

1.7. Conclusion

La classification des sons constitue un aspect central du traitement des signaux, ayant des applications pratiques et diversifiées dans plusieurs domaines. Malgré l'abondance des défis techniques, les avancées de l'IA et de l'apprentissage profond augmentent sans cesse la précision et la performance des modèles. Cette technologie laisse entrevoir un futur plein d'innovations avancées, améliorant la sécurité, la santé et le bien-être de notre vie quotidienne.

Dans ce chapitre, nous avons présenté la nature des sons, leurs catégories, ainsi que les enjeux et les défis liés à leur classification. Nous avons également mis en évidence les nombreuses applications pratiques possibles dans divers domaines.

Chapitre 2

Méthodes d’apprentissage automatique

2.1. Introduction

Le domaine de l'intelligence artificielle est scindé en plusieurs sous domaines imbriqués (présentés figure 1). L'apprentissage automatique est un sous domaine de l'intelligence artificielle consistant à apprendre par l'expérience ou par une base de données des règles implicites pour répondre à un problème donné. Ce domaine s'oriente spécialement autour de l'analyse statistique de données d'entraînement. Dans le cadre de l'apprentissage automatique, nous verrons qu'il existe de nombreux algorithmes utilisant des modèles mathématiques variés. Le réseau de neurones est un de ces modèles, certainement le plus répandu et celui utilisé dans les domaines les plus diverses. L'apprentissage profond est un ensemble de techniques tirant parti des réseaux de neurones pour résoudre des problèmes complexes. Ces techniques sont très utilisées, notamment dans le domaine du traitement d'image, le traitement de séries temporelles (reconnaissance des sons...)[27].

Dans ce chapitre, nous nous intéressons à l'apprentissage automatique, sa définition, ainsi que les définitions de certains types.Où nous faisons un panorama sur les méthodes d'apprentissage automatique, puis nous mettons l'accent sur les méthodes de deep Learning. Nous achevons ce chapitre par une explication détaillée de l'algorithme utilisé dans l'approche proposée.

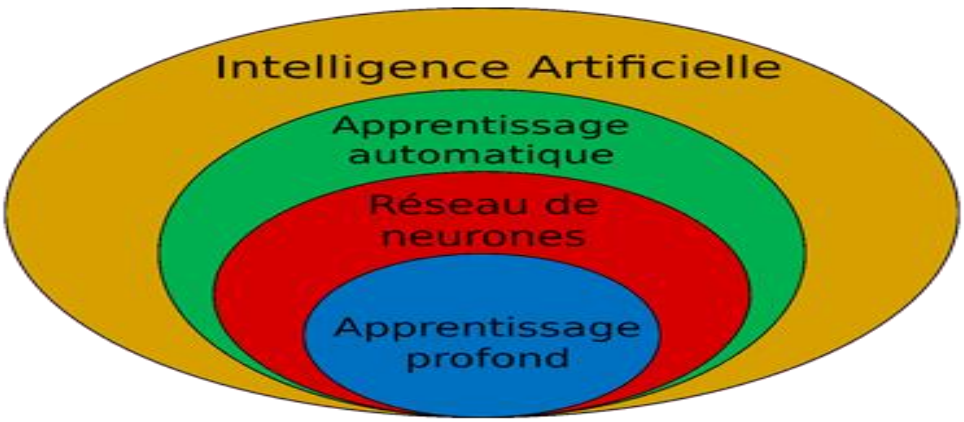


Figure 2. 1 : Schéma de décomposition du domaine de l'intelligence artificielle et de ces sous-domaines [27].

2.2. L'apprentissage automatique

2.2.1. Définition

L'apprentissage automatique ou machine Learning est un sous-domaine de l'intelligence artificielle. Il s'agit d'un ensemble de transformations de données et de techniques mathématiques qui permettent aux logiciels d'apprendre à partir de

données historiques afin de s'ajuster automatiquement pour améliorer les performances d'une tâche spécifique.

Le mot apprentissage, dans l'apprentissage automatique, est une métaphore de la façon dont les humains (ou tout organisme intelligent) peuvent progressivement améliorer leurs performances grâce à des observations et des leçons tirées d'expériences passées.

L'apprentissage automatique présente de nombreux avantages qui justifient sa popularité croissante. En effet, la disponibilité des données et l'augmentation de la puissance de calcul permettent de construire des logiciels « intelligents » capables de résoudre des problèmes de plus en plus complexes et diversifiés. L'apprentissage automatique permet aux ordinateurs d'apprendre en permanence et d'améliorer leurs capacités d'apprentissage, dépassant même dans certains cas les capacités humaines. [20].

Voici quelques exemples de problèmes relevant de l'apprentissage automatique.

Exemple 1 : Supposons que l'on dispose d'une collection d'articles de journaux. Comment identifier des groupes d'articles portant sur un même sujet?

Exemple 2 : Supposons que l'on dispose d'un certain nombre d'images représentant des chiens, et d'autres représentants des chats. Comment classer automatiquement une nouvelle image dans une des catégories « chien » ou « chat » ? [31].

2.2.2. Les phases de l'apprentissage automatique

De manière générale, les algorithmes d'apprentissage automatique se séparent en plusieurs phases.

1. **Phase d'entraînement (ou d'apprentissage) :** le modèle choisi est soumis à un grand nombre d'exemples significatifs. Le système cherche alors à apprendre des règles implicites en se basant sur ces données (appelées données d'entraînement). Cette phase d'entraînement précède généralement l'utilisation du modèle, bien que certains systèmes continuent d'apprendre indéfiniment s'ils disposent d'un retour sur les résultats (on appelle cela de l'apprentissage en ligne).
2. **Phase d'inférence :** Le modèle entraîné peut être utilisé sur de nouvelles entrées. Les entrées fournies lors de la phase d'inférence peuvent être traitées même si elles n'ont pas été vues par le modèle lors de la phase d'apprentissage. En effet, grâce à l'extraction de règles implicites, le modèle peut se généraliser à des entrées inconnues [27].

2.3. Types d'apprentissage automatique

Le Machine Learning se décline sous différents types de modèles, qui emploient chacun des techniques algorithmiques différentes. Selon la nature des données et le résultat souhaité, l'un de ces quatre modèles d'apprentissage peut être utilisé : supervisé, non supervisé, semi-supervisé ou par renforcement. Dans chacun de ces modèles, une ou plusieurs techniques algorithmiques peuvent être appliquées. Tout dépend des ensembles des données qui seront utilisés et de l'objectif visé au niveau des résultats. Par nature, les algorithmes de Machine Learning sont conçus pour classifier des éléments, repérer des patterns, prévoir des résultats et prendre des décisions éclairées. Les algorithmes peuvent être mis en œuvre individuellement ou en groupe dans le but d'atteindre la plus grande précision possible lorsque les données utilisées sont complexes et imprévisibles[21]. Il existe plusieurs types d'apprentissage automatique, mais ils peuvent être largement classés en quatre catégories principales : L'apprentissage automatique se décline en plusieurs types, chacun ayant ses propres caractéristiques et applications :

- L'apprentissage supervisé.
- L'apprentissage non supervisé.
- L'apprentissage semi-supervisé.
- L'apprentissage par renforcement.

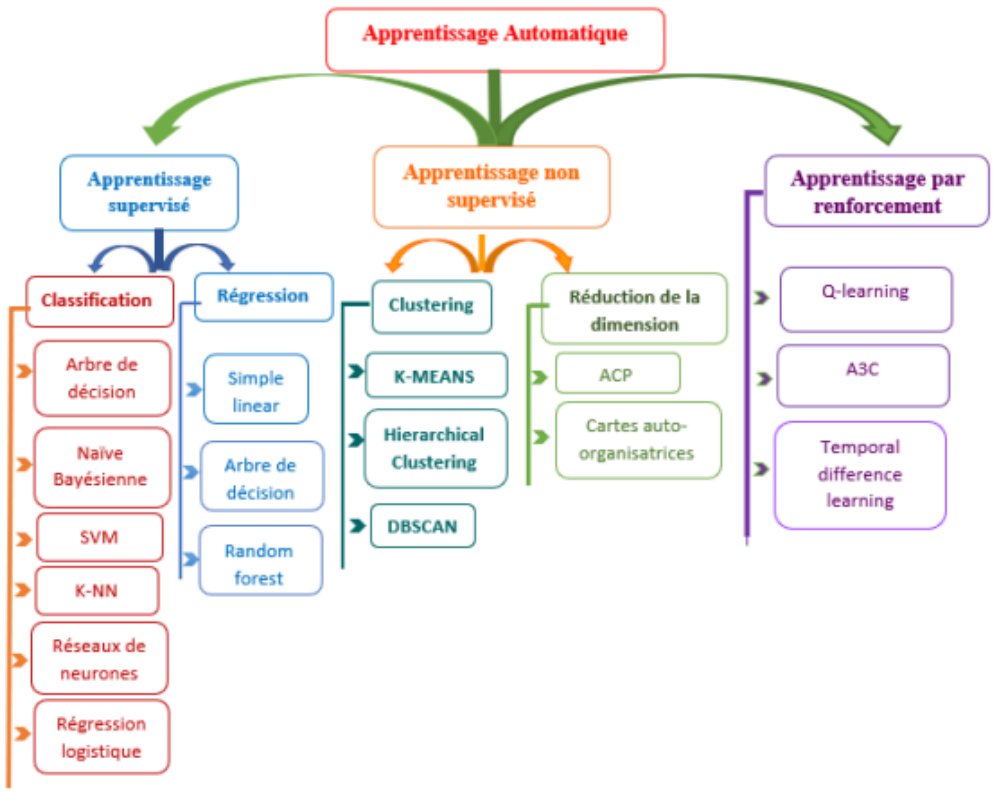


Figure 2. 2 : Les types d'apprentissage automatique [22]

2.3.1. L'apprentissage supervisé

L'apprentissage supervisé, aussi appelé machine Learning supervisé, fait partie des méthodes d'apprentissage automatique. Il consiste à entraîner un algorithme à l'aide de données d'entrée et de sortie connues et étiquetées. Le but est que l'algorithme apprenne les associations entre les entrées et les sorties pour qu'il puisse faire des prédictions précises sur de nouvelles données pour lesquelles la sortie est inconnue.

Concrètement, l'algorithme reçoit un jeu de données d'entraînement contenant des exemples annotés. Chaque exemple est constitué d'une entrée (les caractéristiques) et d'une sortie correspondante souhaitée. Dans le cas d'un problème de classification d'images, les entrées peuvent être des images étiquetées par la classe à laquelle elles appartiennent (chat, chien, etc.).

L'algorithme analyse alors les données d'entraînement pour trouver des motifs et des règles permettant de mapper les entrées aux sorties. Il optimise un modèle mathématique en minimisant l'erreur entre les prédictions et les vraies étiquettes. Une fois entraîné, le modèle est utilisé pour faire des prédictions sur de nouvelles données inconnues [32].

Donc L'apprentissage supervisé utilise des ensembles de données connus, établis et classés pour trouver des modèles. Développons l'idée précédente des images de chiens et de chats. Vous pourriez avoir un énorme ensemble de données avec des milliers d'animaux différents représentés sur des millions d'images. Vu que les types d'animaux sont connus, ils ont pu être groupés et étiquetés avant de les transmettre à l'algorithme d'apprentissage automatique supervisé pour qu'il apprenne à les comprendre.

L'algorithme supervisé compare à présent l'entrée à la sortie et l'image à l'étiquette du type d'animal. Il apprendra à terme à reconnaître un certain type d'animal dans les nouvelles photos qu'il rencontre [23].

Figure 2. 3: illustration de l'apprentissage supervisé. L'algorithme va apprendre à partir des images étiquetées de chiens et de chats. L'image dans le coin inférieur droit est la nouvelle entrée qu'il devra étiqueter [33].

L'apprentissage supervisé est généralement utilisé pour de la classification et la régression.

- **La Classification**

La classification est un processus consistant à trouver une fonction qui aide à diviser l'ensemble de données en classes en fonction de différents paramètres. Dans la classification, un programme informatique est entraîné sur l'ensemble de données d'entraînement et, sur la base de cet entraînement, il catégorise les données en différentes classes. La tâche de l'algorithme de classification est de trouver la fonction de mappage pour mapper l'entrée (x) à la sortie discrète (y).
Les algorithmes de classification peuvent être divisés en plusieurs types, on peut citer quelques-uns [22]:

- Arbre de décision (DT).
- Naïve Baye sienne (NB).
- Machine à vecteurs de support (SVM)
- Les K-voisins les plus proches (KNN).
- Réseau neuronal artificiel (ANN).
- Régression logistique (LR).

Arbre de décision :L'arbre de décision (DecisionTree ou DT) est une technique de base de l'analyse dedonnées. Il a été utilisé dans plusieurs domaines comme l'apprentissage automatique, la reconnaissance de forme, le diagnostic médical, etc.

Dans sa forme la plus simple, un arbre de décision est un type d'organigramme quimontre un chemin clair vers une décision. En termes d'analyse de données, il s'agit d'untype d'algorithme qui inclut des instructions de contrôle (conditionnelles) pour classer lesdonnées.

Un arbre de décision commence à un point unique (nœud) qui se ramifie ensuite dansdeux directions ou plus. Chaque direction générale offre différents résultats possibles,incorporant une variété de décisions et d'événements aléatoires jusqu'à ce qu'un résultatfinal soit atteint [22]. Lorsqu'ils sont montrés visuellement, leur apparence ressemble à unarbre, d'où son nom.

La Classification naïve bayésienne : La classification naïve bayésienne (NB) est une autre méthode d'apprentissage supervisé, elle est listée parmi les modèles probabilistes les plus connus. Il s'agit d'un typede classification simple (" naïf") reposant sur le théorème de Bayes.

Les algorithmes Naïves Bayes sont utilisés dans la catégorisation et la classification de documents. Cette technique permet d'estimer la probabilité de chaque classe parmi les exemples trouvés, et donner une étiquette à la classe la plus probable [22].

Machines à vecteurs de support : La machine à vecteurs support (Support Vector Machine ou SVM), est un algorithmed'apprentissage automatique supervisé utilisé en particulier pour les problèmes de classification.Cet algorithme place toutes les données sous forme de points dans une zone de ndimensions de caractéristiques et les classifier avec l'hyperplan qui différencie les classes.Il est très efficace pour traiter les données en masse, grâce aux vecteurs de supportpour la prédiction [22].

Les k-plus proches voisins : La méthode des k-plus proches voisins (KNN) est une approche de classification supervisée, elle consiste à déterminer pour chaque nouvel individu que l'on veut classer,la liste des k plus proches voisins parmi les individus déjà classé. L'individu est affecté à la classe qui contient le plus d'individus parmi ces k plus proches voisins.Cette méthode nécessite de choisir une distances (la distance Euclidienne, la distance deManhattan, distance de Jaccard, etc.), et donc le nombre k de voisins à prendre en compte.L'inconvénient majeur de cette approche est qu'elle est incapable d'analyser des données volumineuses pour un ensemble d'apprentissage qui nécessite un large espace déstockage[22].

Les réseaux de neurones artificiels : Les réseaux de neurones artificiels (ANN) sont l'un des principaux outils les plusutilisés dans l'apprentissage automatique. Ce sont des structures inspirées des circuitsnerveux du système nerveux humain et se composent d'un groupe d'unités informatiquesinterconnectées appelées neurone artificiel [22].

Un neurone artificiel consiste en une fonction de transfert avec plusieurs entrées pourune seule sortie. Dans un réseau, ces neurones sont connectés entre eux, la sortie d'unneurone pouvant être connectée à un ou plusieurs autres neurones.

Chaque neurone envoieet reçoit des impulsions d'autres neurones, et ces changements sont utilisées pour prédirela classe des données en entrée [22].

Un réseau de neurones très commun est le perceptron multicouche. Dans unperceptron, les neurones sont organisés en couches (une ou plusieurs) où chaque coucheprend en entrée l'ensemble des sorties de la couche précédente. Pourentraîner un réseau de neurones, on ajuste les poids associés à chaque connexion entredeux neurones [22].

La Régression logistique : La régression logistique (RL) est l'une des plus importantes techniques statistiques et d'exploration de données employées par les statisticiens et les chercheurs pour l'analyseet la classification des ensembles de données à réponse binaire. Certains des principauxavantages de la LR sont qu'elle peut naturellement fournir des probabilités et s'étendreaux problèmes de classification multi-classes. Un autre avantage est que la plupart desméthodes utilisées dans l'analyse des modèles LR suivent les mêmes principes que ceux utilisés dans la régression linéaire [22].

- **La régression**

La régression est un processus de recherche des corrélations entre les variables dépendantes et indépendantes. Il aide à prédire les variables continues telles que la prévision des tendances du marché, la prévision des prix des maisons, etc. La tâche de l'algorithme de régression est de trouver la fonction de mappage pour mapper la variable d'entrée (x) à la variable de sortie continue (y). En d'autres termes, la variable de sortie correspondante n'est pas une classe mais une valeur mathématique (valeur réelle) ; un pourcentage ou une valeur absolue. Par exemple, une probabilité pour une machine de tomber en panne (15 % ,20 %, etc.) ou le prix de vente idéal d'un appartement en fonction de critères comme la surface, le quartier, etc. Les algorithmes les plus courants de la régression sont :

- La Régression linéaire simple (SLR).
- La Régression de l'arbre de décision (DTR).
- La Régression de foret d'arbres décisionnels (DTFR)

La principale différence entre les modèles de régression et de classification est que les algorithmes de régression sont utilisés pour prédire **des valeurs continues** (scores de test), tandis que les algorithmes de classification prédisent des **valeurs discrètes** (spam/pas spam, homme/femme, vrai/faux) [24].

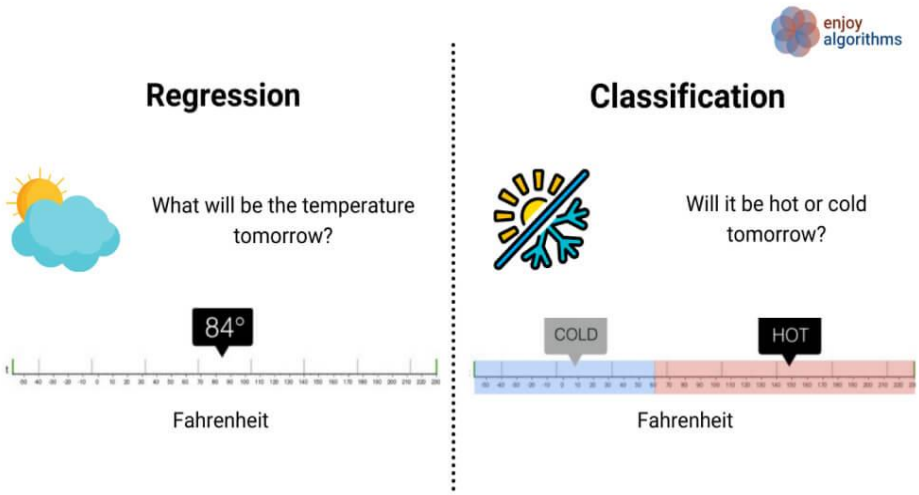


Figure 2. 4: différence entre classification et régression [24].

2.3.2. Apprentissage non supervisé

Contrairement à l'apprentissage supervisé, l'apprentissage non supervisé utilise un ensemble de données non étiquetées. L'objectif est de découvrir des structures, des patterns ou des relations inhérentes aux données. Les algorithmes de regroupement (clustering) et les techniques de réduction de dimensionnalité font partie de l'apprentissage non supervisé

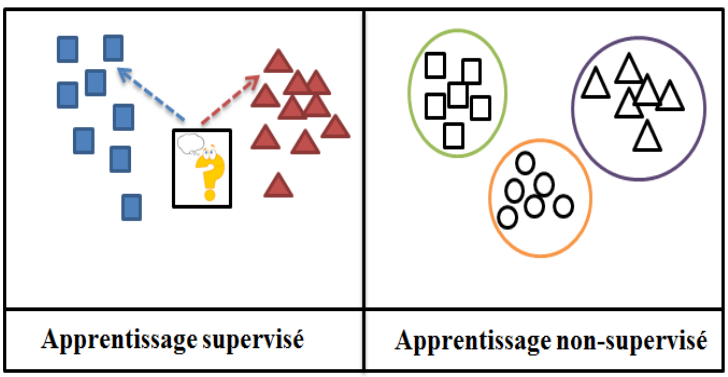


Figure 2. 5:différence entre apprentissage supervisé et apprentissage non supervisé[37].

L'Apprentissage supervisé et non supervisé se distinguent donc par la nature de leurs données d'entrée. En apprentissage supervisé, le modèle s'appuie sur des données d'entrée et de sortie étiquetées, tandis qu'en apprentissage non supervisé, il travaille uniquement à partir de données non étiquetées pour identifier des structures ou des classifications.

Le but d'un modèle non supervisé consiste à découvrir des informations à partir d'un large ensemble de données, tandis qu'en apprentissage supervisé, il s'agit de prédire des valeurs de sortie pour de nouvelles données [36].

2.3.3. Apprentissage semi-supervisé

Comme son nom l'indique, ce modèle d'apprentissage automatique est un juste milieu entre l'apprentissage supervisé et non supervisé, qui utilise des données étiquetées et non étiquetées pour entraîner l'ordinateur. L'apprentissage automatique semi-supervisé est notamment utilisé lorsque l'algorithme ne dispose pas de suffisamment de données pour apprendre. On utilise alors un plus petit nombre de données structurées pour alimenter la machine, afin de guider celle-ci dans son processus de catégorisation des données non structurées [34].

2.3.4. Apprentissage par renforcement

Dans l'apprentissage par renforcement, un agent interagit avec un environnement et apprend à prendre des actions afin de maximiser une récompense cumulative. L'agent explore l'environnement, reçoit des récompenses ou des pénalités en fonction de ses actions, et ajuste sa politique pour prendre des décisions optimales[48].

2.4. Méthodes d'apprentissage automatique

2.4.1. Méthodes traditionnelles

1) Modèle de Mélange Gaussien (GMM pour Gaussien Mixture Model)

Un GMM est un modèle statistique exprimé sous forme d'une somme pondérée de M composants gaussiens. Il s'appuie sur un modèle de probabilité des caractéristiques du signal sur chaque classe (x)[49] :

$$p(X) = \sum_{i=1}^{M} w_i p_i\,(X)$$

Où M est le nombre de Gaussiens utilisés dans le modèle, wi le poids assigner au composant (X) de la distribution qui est à son tours donné par :

$$p_i(x) = \frac{1}{(2\pi)^{k/2}|\boldsymbol{\sigma_i}|^{1/2}} e^{-(1/2)(X-\boldsymbol{\mu_i})\Sigma_i{}^{-1}(X-\boldsymbol{\mu_i})}$$

$\boldsymbol{\mu i}$ est la moyenne et $\boldsymbol{\sigma i}$ la matrice de covariance de la distribution.
Pendant l'apprentissage, les paramètres $\boldsymbol{\mu i}$ et $\boldsymbol{\sigma i}$ de la distribution sont estimés à l'aide de l'algorithme de maximisation de l'attente (EM) et des caractéristiques

extraites des données audio. Le résultat de l'algorithme EM est un gaussien qui correspond le mieux aux données d'apprentissage, selon le critère dit du maximum de vraisemblance. Un certain nombre de N modèles de mélange gaussiens sont créés, où N est égal au nombre de classes reconnues. Pour distinguer entre la classe considérée et le bruit, un seuil est appliqué au fichier des gaussiens de la classe considérée. Pour distinguer entre plusieurs classes d'événements acoustiques, le critère de probabilité d'postériori maximum (MAP) est utilisé pour déterminer le type de l'événement acoustique [49].

2) Séparateur à Vaste Marge (SVM)

Le séparateur à vaste marge SVM (Support Vector Machine) est un classificateur binaire basé sur la notion de marge dur. Il effectue une discrimination entre deux classes à l'aide d'un hyperplan dans l'espace des caractéristiques d'entrée de dimension k. La séparation d'un espace de 2 dimensions par un hyperplan est présentée à la Figure 2.2. Lors que les données sont linéairement séparables les vecteurs d'entrés Xi et leurs sorties Yi satisfont la condition suivante [49] :

$$y_i = \begin{cases} 1 & \text{pour } w.x_i + b \geq 1 \\ -1 & \text{pour } w.x_i + b \leq -1 \end{cases}$$

Avec w le vecteur des poids de l'hyperplan, i le nombre de vecteurs d'observation et le coefficient b une constante de l'hyperplan qui est tel que w. x + b = 0.Ainsi en supposant que les données négatives se trouvent d'un côté et les données positives de l'autre côté, le vecteur se trouvant entre $w.\ x + b = 1$ et $w.\ x + b = -1$ est **le séparateur à vaste marge** [49].

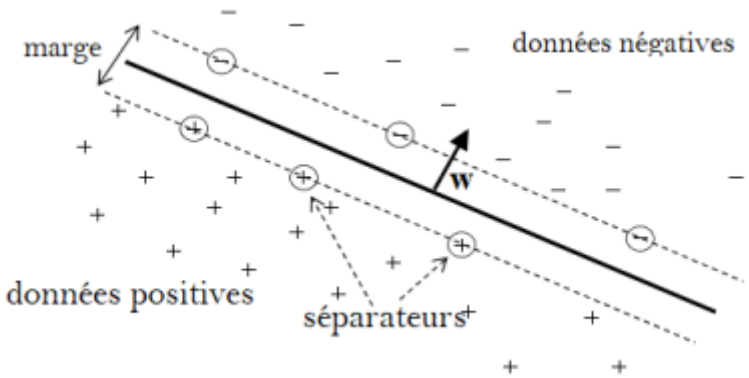


Figure 2. 6:Séparation d'un espace 2D par un hyperplan [49].

Dans la phase d'entrainement la marge entre l'hyperplan et le SVM est maximisée, cette optimisation favorise la généralisation aux données inconnues. Il peut être montré que cette marge est égale à la moitié du module du vecteur poids, bien que l'optimisation est définie par [49] :

$$w^* = \arg min ||w||^2$$

Qui satisfait à la condition ci-haut. En pratique la résolution se fait en utilisant le Lagrangien. Dans le cas où les données ne sont pas linéairement séparables, on recourt à une fonction kernel afin de le faire correspondre à un espace de plus grand dimension tel que [49] :

$$K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$$

Avec,xj deux vecteurs de données et Φ(xt) une transformation des vecteurs d'entrés en une nouvelle espace de dimension différentes . K est la fonction kernel qui fait la correspondance d'espace, le but étant d'aboutir en des données linéairement séparables. Les fonctions le plus utilisées sont la fonction polynomiale de degré d, la sigmoïdale et la tangente hyperbolique [49].

Le SVM peut être utilisé aussi pour une classification multi-classes donc non binaire, à l'aide des méthodes comme « un à plusieurs », où l'on traite les classes reconnues comme positives et les autres restant négatives ; ou encore « un à un » où $M \cdot (M - 1)/2$ classificateurs sont entrainés, un pour chaque pair de classe, la classe gagnante est celle qui a le plus gagner la comparaison « un à un » [49].

2.4.2. Méthodes modernes basées sur l'apprentissage profond

Les réseaux neuronaux et l'apprentissage en profondeur donnent de nos jours de bonnes solutions à plusieurs problèmes de reconnaissance et de class
ification d'image, raison pour Laquelle ils sont devenus sujets importants en informatique et dans l'industrie économique. Inspirés de la structure du cerveau de l'être humain, les réseaux de neurones artificiels font partie des moyens qui permettent de rendre les ordinateurs plus humains et aider ainsi les machines à raisonner comme des êtres humains. Le cerveau humain apprend d'une manière essentielle par l'expérience. Il s'agit donc d'une preuve naturelle que certains problèmes peuvent dépasser la portée des ordinateurs[54].

1. définition l'apprentissage profond

L'apprentissage profond se réfère à une classe de techniques d'apprentissage automatique, où de nombreuses couches de stades de traitement de l'informationdans des architectures hiérarchiques sont exploitées pour la classification de motifs et pour l'apprentissage de caractéristiques ou de représentations. C'est à l'intersection des domaines de recherche des réseaux de neurones, de la modélisation graphique, de l'optimisation, de la reconnaissance de

motifs et du traitement du signal. Trois raisons importantes de la popularité de l'apprentissage profond aujourd'hui sont l'augmentation drastique des capacités de traitement des puces (par exemple, les unités GPU), la réduction significative du coût du matériel informatique, et les avancées récentes dans la recherche en apprentissage automatique et en traitement des signaux/informations.

Les chercheurs actifs dans ce domaine incluent ceux de l'Université de Toronto, de l'Université de New York, de l'Université de Montréal, de Microsoft Research, de Google, d'IBM Research, de Baidu, de Facebook, de l'Université de Stanford, de l'Université du Michigan, du MIT, de l'Université de Washington, et de nombreux autres endroits. Ces chercheurs ont démontré les succès de l'apprentissage profond dans diverses applications de la vision par ordinateur, de la reconnaissance phonétique, de la recherche vocale, de la reconnaissance de la parole conversationnelle, du codage des caractéristiques de la parole et de l'image, de la classification des énoncés sémantiques, de la reconnaissance de l'écriture manuscrite, du traitement audio, de la reconnaissance d'objets visuels, de la récupération d'informations, et même dans l'analyse des molécules qui pourraient mener à la découverte de nouveaux médicaments[39].

2.Intelligence artificielle, l'apprentissage automatique et l'apprentissage profond

L'apprentissage profond (Deep Learning), l'apprentissage automatique (Machine Learning) et l'intelligence artificielle sont trois concepts interdépendants qui ont révolutionné le domaine de l'informatique (voir la figure 1.9). L'apprentissage profond est un sous-ensemble de l'apprentissage automatique, qui est un type d'intelligence artificielle (IA) [40].

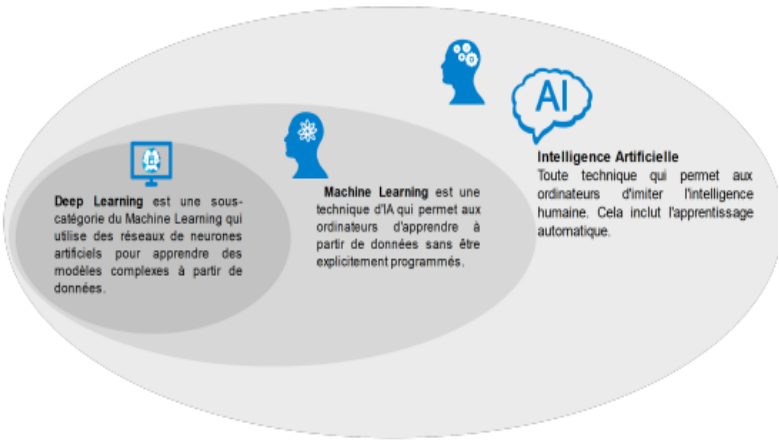


Figure 2. 7: IA et ses sous domaines [40].

L'intelligence artificielle (IA) : L'IA est la capacité des machines à imiter les fonctions cognitives humaines telles que l'apprentissage, la perception, le raisonnement et l'interaction sociale.

La machine Learning (ML) : Le ML est une technique d'IA qui permet aux ordinateurs d'apprendre à partir de données sans être explicitement programmés.

L'apprentissage automatique (AA) : L'apprentissage automatique est un sous-domaine de l'intelligence artificielle (IA) qui implique le développement d'algorithmes et de modèles statistiques permettant aux systèmes informatiques d'apprendre et de s'améliorer à partir de l'expérience sans être explicitement programmés. Le principe fondamental de l'apprentissage automatique est de permettre aux ordinateurs d'apprendre à partir de données, d'identifier des modèles et de faire des prédictions ou de prendre des décisions sur la base de cet apprentissage. Il s'agit d'entraîner le système informatique à reconnaître des modèles, des relations et des corrélations dans de vastes ensembles de données, puis d'utiliser ces connaissances pour faire des prédictions ou prendre des décisions concernant de nouvelles données.

3. les principes de l'apprentissage profond

L'apprentissage profond est un paradigme d'apprentissage automatique inspiré de l'anatomie du cerveau humain. Cet apprentissage est associé à une structure algorithmique que l'on appelle un réseau de neurones. Le neurone biologique est une cellule complexe, mais seul son fonctionnement basique a servi d'inspiration au neurone informatique Le parallèle entre neurone biologique et neurone informatique est illustré figure.
Un neurone biologique reçoit des signaux électriques d'autres neurones en amont via des points d'entrées (x0, x1, x2 et x3). Ces signaux sont accumulés à l'intérieur du corps du neurone, et si la somme de ces signaux dépasse un certain seuil, le neurone s'active et envoie à son tour un signal à des neurones en aval, Cette sortie est calculée à partir des entrées via l'équation [42].

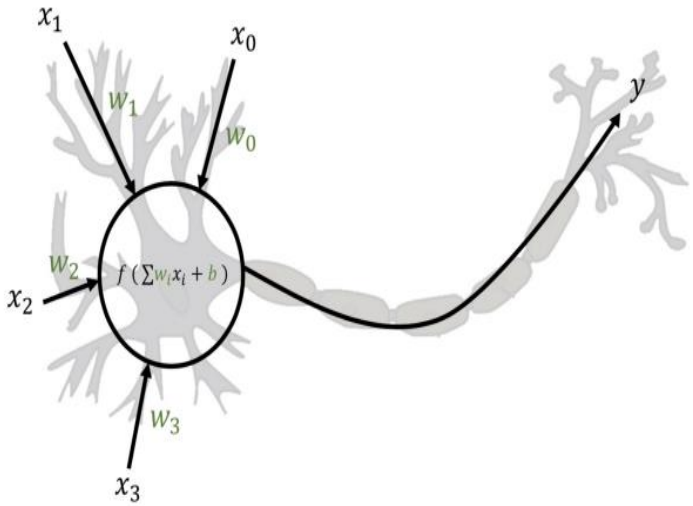


Figure 2. 8 :Schéma d'un neurone informatique superposé à un schéma de neurone biologique [42].

Un neurone informatique approxime ces principes. Il accepte en entrée un nombre fixe de nombres réels (représentant les signaux des neurones en amont) et produit en sortie (y) une valeur réelle (représentant le signal envoyé aux neurones en aval). Cette sortie est calculée à partir des entrées via l'équation :

y = f(n∑ i=0 wixi + b)

Qui représente l'accumulation de signaux électriques. Les entrées sont multipliées par des valeurs wi que l'on appelle les poids, qui représentent la force de la connexion entre ce neurone et les neurones en amont. La fonction f est une fonction dite d'activation. Il s'agit d'une fonction non-linéaire croissante. Les premières implémentations de réseaux définissaient f comme une fonction seuil : f renvoie 1 si son argument est strictement positif et 0 sinon, mais d'autres fonctions furent proposées au fil des années comme la fonction unité linéaire rectifiée (RectifiedLinear Unit en anglais, ou ReLU). Cette fonction représente l'activation du neurone si celui-ci a accumulé suffisamment de potentiel électrique. La valeur b, que l'on appelle le biais, représente l'appétence ou la résistance du neurone à s'activer. Les poids et le biais (souvent désignés collectivement sous le nom de « poids ») sont les paramètres du neurone : ce sont ces valeurs qui sont modifiées au cours d'un entraînement [42].

Un neurone informatique peut constituer à lui seul le support d'un algorithme d'apprentissage pour certaines tâches bien définies. En constituant un jeu d'entraînement composé de paires d'entrées réelles et des sorties binaires attendues, et en présentant séquentiellement ces exemples au neurone, il existe un algorithme d'entraînement qui définit comment modifier les poids de celui-ci afin de converger vers la classification attendue. Cependant, la capacité d'un neurone unique est trop faible pour qu'il soit appliqué ailleurs que sur des cas jouets. Les algorithmes d'apprentissage profond sont basés sur des ensembles de neurones que

l'on appelle des réseaux. On appelle « architecture » la structure selon laquelle les neurones sont reliés entre eux. Les premières architectures s'appelaient le perceptronmulticouche (Multi-Layer Perceptron en anglais, ou MLP). Dans un MLP, les neurones sont connectés à la fois parallèlement et séquentiellement selon une organisation en couches (figure 10). La première couche est constituée d'un certain nombre de neurones qui prennent en entrée les données. Les sorties des neurones de cette couche servent d'entrée à une deuxième couche de neurones, et ainsi de suite, jusqu'à une dernière couche dont on identifie la sortie à proposition faite par le réseau pour l'étiquetage de l'entrée. Cette structure en couches est inspirée de l'architecture neuronale du cerveau humain [42].

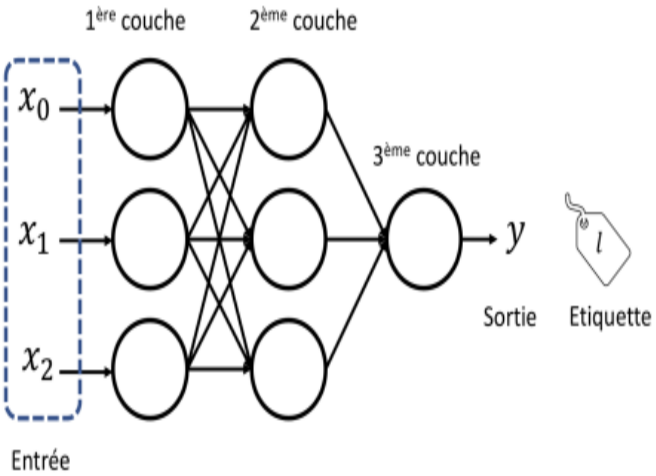


Figure 2. 9 : Un perceptron multicouche ou MLP composé de trois couches[42].

La phase **d'entraînement** d'un réseau se déroule de la façon suivante. Tous les paramètres sont initialisés aléatoirement. Un exemple issu du bloc d'entraînement est présenté au réseau. Le réseau calcule une sortie, que l'on compare à l'étiquette et on en calcule une mesure de performance. Un algorithme que l'on appelle la rétro-propagation d'erreur permet alors de calculer la contribution de chaque poids du réseau à la sortie proposée et donc à la performance pour cet exemple. La connaissance de ces contributions permet de connaître la forme locale de la fonction de performance en fonction des valeurs des poids. Une itération d'un algorithme d'optimisation est alors appliquée pour modifier les valeurs des poids afin d'améliorer la performance du réseau sur cet exemple. Un nouvel exemple est présenté et le cycle recommence. L'entraînement se poursuit jusqu'à ce que la performance se stabilise ou selon d'autres critères d'arrêt. On dit alors que le réseau a convergé ou bien est entraîné. Le réseau peut ensuite être utilisé en prédiction pour produire des étiquettes sur des données qu'on ne lui a jamais présentées [41].

4. Fonctionnement des réseaux de neurones

4.1 Définition et structure générale

Les réseaux de neurones permettent de modéliser des phénomènes complexes. Ils sont basés sur l'enchaînement de modèles linéaires et de fonctions non linéaires. Les transformations hi = ($Wihi$−1 + bi) sont appelées couches. On distingue la couche d'entrée, qui reçoit le vecteur d'entrée x, la couche de sortie, qui renvoie un vecteur de sortie y (sans lui appliquer de fonction non linéaire σ), et les couches cachées, qui reçoivent et renvoient 5 des vecteurs intermédiaires$h$. Ces vecteurs intermédiaires sont appelés représentations cachées (ou représentations latentes) et sont des variables internes au réseau de neurones. Les non linéarités σ, quant à elles, sont appelées fonctions d'activation et nous les présenterons plus en détail dans la suite de cette partie. Il est intéressant de noter que les fonctions d'activation peuvent en théorie être différentes pour chaque couche (même si ce n'est généralement pas le cas)[42].

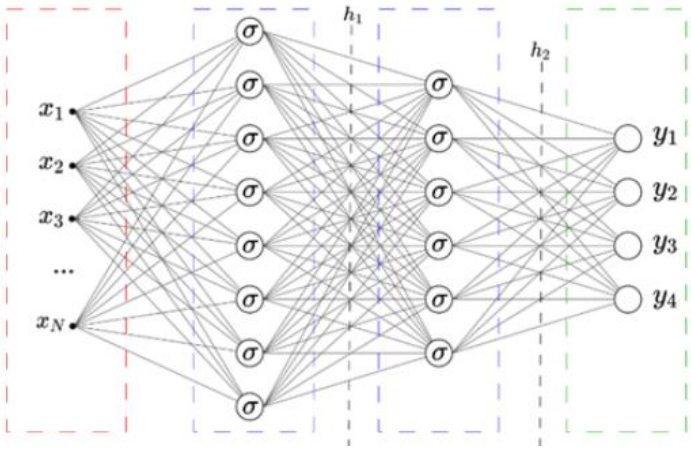


Figure 2. 10 :La fonctions d'activation : algorithmes mathématiques appliqués aux valeurs de sortie[27].

La fonction d'activation est une fonction qui prend en entrée un réel et donne en sortie un nombre comprise entre 0 et 1 qui est une pratique courante en apprentissage automatique car ce nombre permet d'évaluer la probabilité. Cette fonction est souvent non-linéaire enfin de bien représenter les problèmes de réels qui ne sont pas linéaire. Les fonctions d'activation que l'on trouve plus dans la littérature sont [49].

Sigmoïde : est une fonction non linéaire strictement croissante et continue qui donne des valeurs comprises entre 0 et 1[49]. Elle est donnée par l'équation suivant :

$$g(x) = sigma(x) = \frac{1}{1 + e^{x}}$$

Tangente hyperbolique : c'est une fonction circulaire non linéaire et continue qui donne des valeurs comprises entre -1 et 1. Son avantage est qu'elle est appropriée aux nombres négatifs [49]. Elle est donnée par l'équation :

$$g(x) = \tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

ReLU : le Rectifiedlinear unit est une fonction d'activation simple qui rend nul les nombres négative et introduit des trous dans le réseau et permet ainsi d'éviter la disparition du gradient [49]. Donnée par :

$$ReLU(x) = \max(0, x)$$

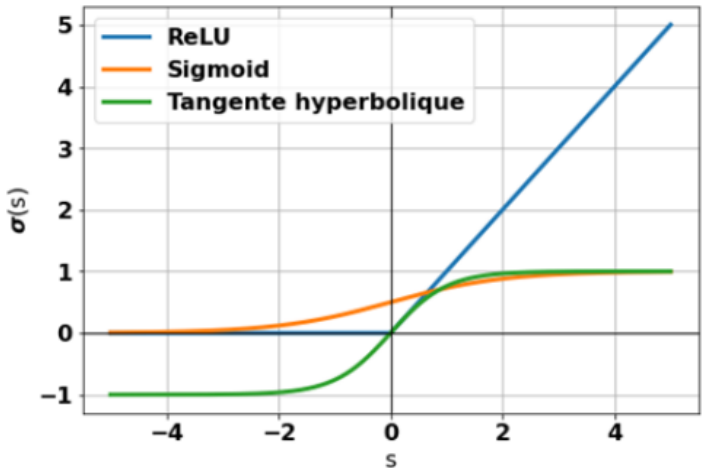


Figure 2. 11 : Évolution des différentes fonctions d'activations présentée. Les tracés sont faits pour des valeurs des neurones compris entre -5 et 5 [42].

Soft max : est une fonction très utilisé dans la classification multi-classes. Elle est donnée par :

$$softmax\ (x_i) = \frac{e^{x_i}}{\sum_m e^{x_m}}$$

$$\text{Tel que } softmax\ (x_m) > 0\ \ \forall m\ et\ \sum_m softmax\ (x_m) = 1$$

4.2 - Types des réseaux de neurones

Il existe différents types de réseaux de neurones, et ils sont classés en fonction du nombre de nœuds cachées du modèle ou encore du nombre d'entrées et de sorties de chaque nœud :

1) **Les réseaux de neurones convolutifs**

1. définition :

Les réseaux de neurones convolutifs (CNN/ConvNet) sont une classe de réseaux de neurones profonds largement utilisés dans l'analyse d'images. Ils se distinguent par l'utilisation de la convolution, une opération mathématique qui permet de capturer des caractéristiques visuelles importantes. Les CNN utilisent également des connexions spéciales appelées "by-pass" ou "skip connexion" pour améliorer la performance et la capacité d'apprentissage du réseau)[47].

2. Principe d'architecture d'un CNN :

Un réseau de neurones convolutif n'est pas seulement un réseau neuronal profond avec de nombreuses couches cachées. Il s'agit plutôt d'un réseau profond qui simule le fonctionnement du cortex visuel du cerveau pour reconnaître et classifier des images ou des vidéos, et pour découvrir un objet ou même une partie dans une image [47].

Dans le cadre de la classification audio, Les réseaux de neurones convolutifs ont une méthodologie similaire à celle des méthodes traditionnelles d'apprentissage supervisé : ils reçoivent des sons en entrée, détectent les caractéristiques de chacune d'entre elles, puis entraînent un classifier dessus [46].

Un signal audio brut se présente sous forme d'une onde qui varie en amplitude au fil du temps. Toutefois, un CNN ne peut pas traiter directement cette forme de données de manière efficace. Il est donc nécessaire de **convertir le signal en une représentation visuelle** telle qu'un **spectrogramme** ou des **coefficients MFCC**, qui serviront ensuite d'entrée au réseau convolutif pour l'apprentissage et la classification des sons [45].

Les CNN sont puissants pour la classification du son en raison de leur capacité à extraire automatiquement des caractéristiques spectrales et temporelles **des spectrogrammes**. L'amélioration des performances passe par un bon prétraitement du signal et une architecture optimisée du réseau.

Le spectrogramme est un diagramme représentant le spectre d'un phénomène périodique, associant à chaque fréquence une intensité ou une puissance. L'échelle des fréquences et celle des puissances ou intensités peuvent être linéaires ou logarithmiquesDe nos jours l'outil informatique ayant bien évolué, un ordinateur domestique possède une puissance de calcul suffisante pour réaliser un spectrogramme rapidement. Dans ce projet, nous réaliserons un programme capable d'extraire les données utiles d'un son (contenu dans un fichier audio), puis d'effectuer les calculs nécessaires pour la construction d'un spectrogramme. Enfin ce même programme synthétisera l'image correspondant au spectrogramme calculé [50].

Un spectrogramme est un outil puissant qui peut être utilisé à des fins analytiques, pratiques et créatives dans le monde de l'audio Pour générer un spectrogramme à partir d'un signal audio, plusieurs étapes sont nécessaires :

1. Lecture du Fichier Audio : Ouvrez le fichier audio à l'aide d'un programme ou logiciel capable de gérer les fichiers audio (par exemple, WAV).
2. Échantillonnage et Conversion : Le signal audio est échantillonné à intervalles réguliers et converti en données numériques
3. Calcul de la Transformée de Fourier : Divisez le signal en segments courts (quelques millisecondes) et appliquez une transformée de Fourier discrète (FFT) pour obtenir le spectre fréquentiel de chaque segment
4. Création du Spectrogramme : Empilez les spectres fréquentiels sur l'axe temporel pour former le spectrogramme, où l'intensité des couleurs représente l'amplitude des composantes fréquentielles.
5. Visualisation sous Forme d'Image : Utilisez un logiciel comme MATLAB ou Python avec Matplotlib pour visualiser et exporter le spectrogramme sous forme d'image numérique.

Le traitement de son est naturellement un problème de traitement numérique du signal et dans l'apprentissage automatique on le trouve dans la modélisation des données séquentielles. Mais en découpant les données sonores en morceaux comme pour le cadrage, le cepstre de chaque cadre dans un domaine donné (temporaire, fréquentielle, …) présente une structure spatiale qui peut être visualisé et comprise comme une image. Orientant le problème dans ce sens le traitement de son pourra appartenir au traitement d'image et les techniques de la vision artificielle peuvent se l'approprié [49].

Les CNN permettent de prendre en compte le lien qu'il y a entre des pixels proches dans une image. En d'autres termes, ils prennent en compte la dimension spatiale d'une image. Ce sont des réseaux neuronaux constitués de deux composantes : un extracteur de caractéristiques (ou features) qui comprend des couches de convolutions (convolutionallayers) et des couches de pooling (poolinglayers), et un classificateur qui correspond à un perceptron multicouche (multilayerperceptron)[48].

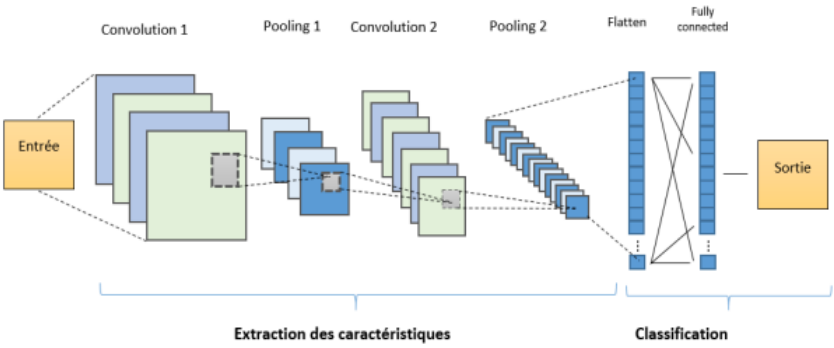


Figure 2. 12: Schéma d'un réseau de neurones convolutifs [54].

Dans les couches de convolutions, des filtres opèrent des convolutions. Ces filtres sont appelés des kernels et glissent de pixel en pixel sur l'entièreté de l'image afin de calculer de nouvelles images qui sont appelées cartes de caractéristiques (ou cartes de caractéristiques). Un kernel est une matrice de nombres qui correspondent aux poids de la couche convolutive et qui vont être ajustés au cours de l'entraînement du modèle. Une couche convolutive peut posséder plusieurs kernels. Par conséquent, plusieurs cartes de caractéristiques peuvent être calculées en sortie de cette couche. Ceci provoque une augmentation de la profondeur de l'image [48].

Dans l'extracteur de caractéristiques, les neurones vont apprendre à extraire des caractéristiques qui correspondent à des patterns présents dans l'image. De couche en couche, les cartes de caractéristiques passent dans des filtres, deviennent de plus petites tailles, plus nombreuses, plus complexes et abstraites pour la vision de l'homme mais plus révélatrices pour la vision de l'ordinateur. Par exemple, les premières couches vont mettre en évidence des lignes et des bords, les suivantes des formes, des coins ou des cercles et les dernières des pattes, des oreilles ou des museaux [48].

À la fin de l'extracteur de caractéristiques, les cartes de caractéristiques sont aplaties dans un vecteur de cartes de caractéristiques afin de passer dans le classificateur (perceptron multicouche). Celui-ci va classifier l'image dans une ou plusieurs classes [48].

3. Architecture de réseaux de neurone convolutif :

Les réseaux de neurones convolutifs (CNN), également appelés Convolutional Neural Networks, Il existe quatre types de couches pour un réseau de neurones convolutif: la couche de convolution, la couche de pooling, la couche de correction ReLU et la couche entièrement connectée (fully-connected). Lorsqu'elles sont empilées dans un ordre spécifique, ces couches forment une architecture CNN. Chacune de ces couches est traitée indépendamment et joue un rôle spécifique dans le processus de traitement des données [47].

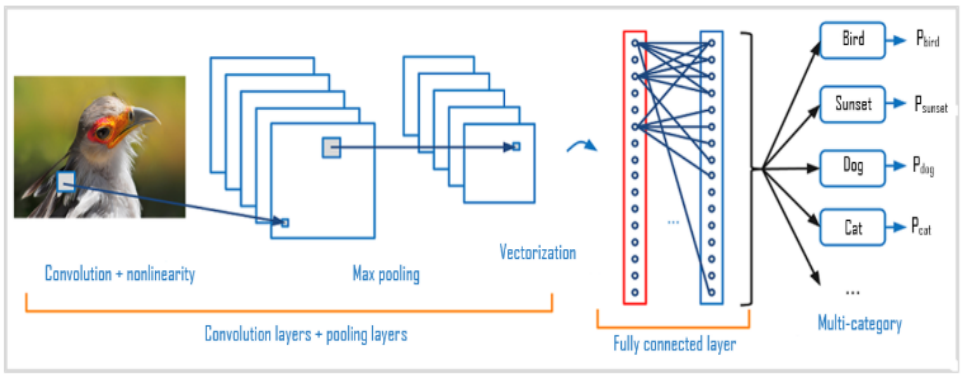


Figure 2. 13 : Architecture d'un CNN[51].

1. **La couche convolutive (CONV)** : La couche de convolution est la composante clé des réseaux de neurones, convolutifs, elle constitue toujours au moins leur première couche[48]. Les couches de convolution sont formées de ce qu'on appelle des filtres. Les filtres sont des tableaux de valeurs appelées featuremaps. Chaque couche de convolution prend en entrée une image et produit une featuremap. Chaque featuremap est obtenue en appliquant le filtre à l'image. Par exemple, si l'image est de taille 5x5 et que le filtre est de taille 3x3, la featuremap sera de taille 3x3. La couche de convolution reçoit donc en entrée plusieurs images et calcule la convolution de chacune d'entre elles avec chaque filtre. Les filtres correspondent exactement aux caractéristiques que l'on souhaite retrouver dans les images [51].

   C'est là toute la puissance des réseaux neuronaux convolutifs : ils sont capables de déterminer automatiquement les éléments discriminants d'une image en s'adaptant au problème donné. Par exemple, si le problème consiste à distinguer les chats des chiens, les caractéristiques automatiquement apprises peuvent décrire la forme des oreilles ou des pattes.

Trois hyperparamètres permettent de dimensionner le volume de la couche de convolution :

1. Profondeur de la couche : nombre de noyaux de convolution (ou nombre de neurones associés à un même champ récepteur).
2. Le pas contrôle le chevauchement des champs récepteurs. Plus le pas est petit, plus les champs récepteurs se chevauchent et plus le volume de sortie sera grand.
3. La marge (à 0) ou zeropadding : parfois, il est commode de mettre des zéros à la frontière du volume d'entrée. La taille de ce zero-padding est le troisième hyperparamètres. Cette marge permet de contrôler la dimension spatiale du volume de sortie. En particulier, il est parfois souhaitable de conserver la même surface que celle du volume d'entrée [52].

En résumé, la couche de convolution est essentielle dans un CNN. Son but est de repérer la présence d'un ensemble de *caractéristiques* dans les images reçues en entrée [52]. La convolution est un outil mathématique simple qui est très

largement utilisé pour le traitement d'image, ce qui explique que les réseaux de neurones à convolution soient particulièrement bien adaptés à la reconnaissance d'image.

La convolution agit comme un filtrage. On définit une taille de fenêtre qui va se balader à travers toute l'image (rappelez-vous qu'une image peut être vue comme étant un tableau). Au tout début de la convolution, la fenêtre sera positionnée tout en haut à gauche de l'image puis elle va se décaler d'un certain nombre de cases (c'est ce que l'on appelle le pas) vers la droite et lorsqu'elle arrivera au bout de l'image, elle se décalera d'un pas vers le bas ainsi de-suite jusqu'à ce que le filtre est parcourue la totalité de l'image[53] (voir la figure 2.14).

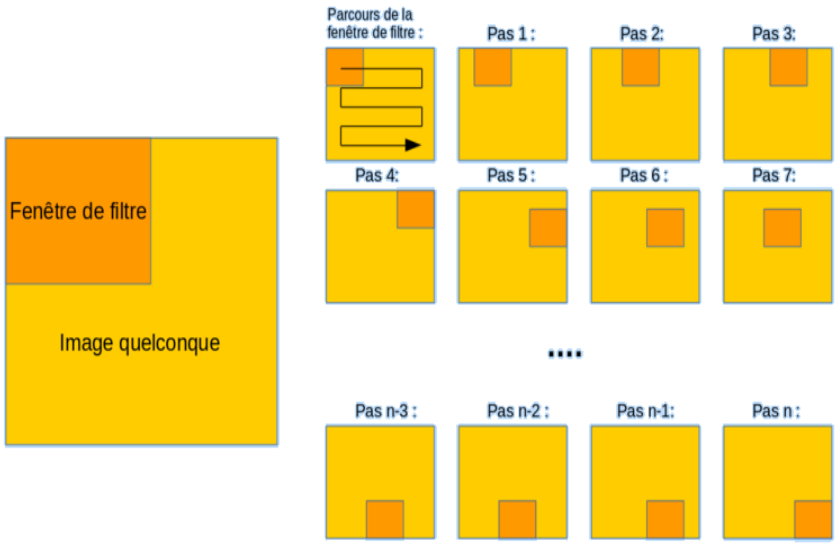
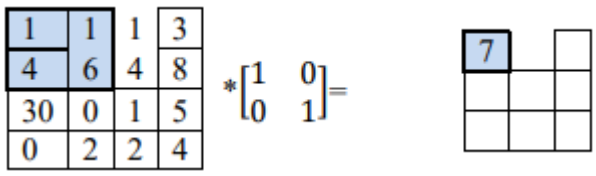


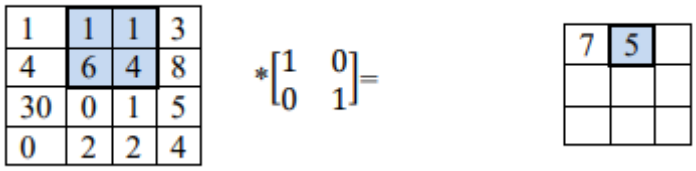
Figure 2. 14 :Schéma du parcours de la fenêtre de filtre sur l'image [53].

1	1	1	3
4	6	4	8
30	0	1	5
0	2	2	4

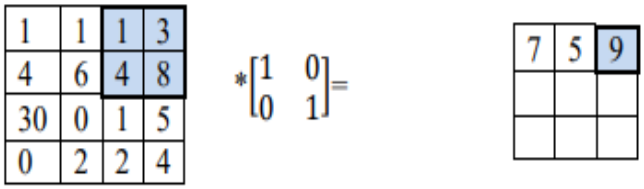
En commençant par le premier filtre, l'opération de convolution commence au coin supérieur gauche de la matrice de la même taille que le filtre de convolution. L'opération de convolution est la somme des produits des éléments qui sont situées sur les mêmes positions des deux matrices. Le résultat de 7 dans la matrice de résultat est trouvé comme suit[47]: (1×1) + (1×0) + (4×0) + (6×1) = 7



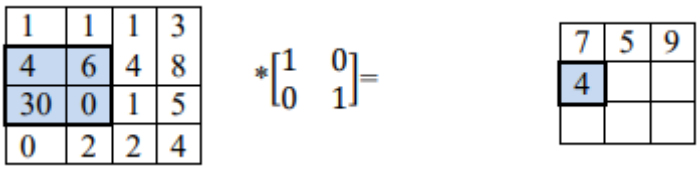
Une autre opération de convolution est menée pour la prochaine sous-matrice:



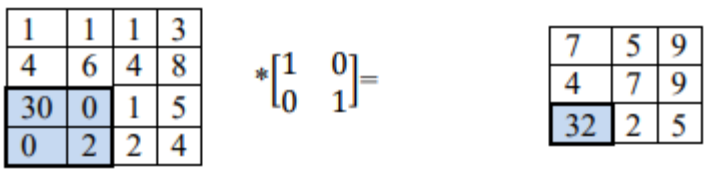
De la même manière, la troisième opération de convolution est menée, comme il est montré dans la figure



Une fois la rangée du haut terminée, la rangée suivante recommence à partir de la gauche.



Le même processus se répète jusqu'à ce que la carte des caractéristiques du filtre donné soit produite, comme le montre la figure



1	1	1	3
4	6	4	8
30	0	1	5
0	2	2	4

$$*\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}=$$

7	5	9
4	7	9
32	2	5

2. **Couche de pooling :**Le regroupement (pooling) de couches est une opération couramment utilisée après une couche convolutive dans un réseau de neurones convolutifs (CNN). Cette opération permet de réduire la dimensionnalité des caractéristiques extraites et d'introduire une certaine invariance aux translations dans les données [47].
   Ce type de couche est souvent placé entre deux couches de convolution : elle reçoit en entrée plusieurs cartes de caractéristiques, et applique à chacune d'entre elles l'opération de pooling. Une couche de pooling, agit comme une couche de réduction. Elle divise l'image en blocs et ne garde que le maximum de chaque bloc. Cela permet de réduire la dimension de l'image tout en conservant les caractéristiques les plus importantes. On obtient en sortie le même nombre de cartes de caractéristiques qu'en entrée, mais celles-ci sont bien plus petites [51].

   Il existe principalement deux types de regroupement couramment utilisés : le regroupement maximum (max pooling) et le regroupement moyen (averagepooling).

   Max Pooling : lorsque le filtre se déplace sur l'entrée, il sélectionne le pixel avec la valeur maximale à envoyer à la matrice de sortie. Cette approche tend à être utilisée plus souvent que la mise en commun moyenne.

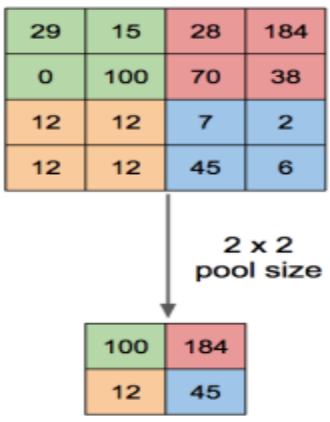


Figure 2. 15 :Pooling maximal

AveragePooling : Lorsque le filtre se déplace sur l'entrée, il calcule la valeur moyenne dans le champ réceptif à envoyer à la matrice de sortie [51].

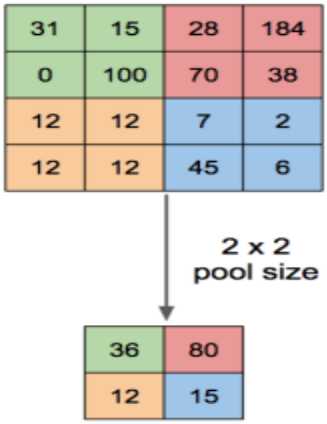


Figure 2. 16: Pooling moyen

3. **Couche entièrement connectée :** Les CNNs sont généralement formés de plusieurs couches de convolution et de pooling, suivies par une couche entièrement connectée quicombine les caractéristiques extraites par les couches précédentes pour classifier l'image, elle renvoie un vecteur de taille N, où N est le nombre de classes dans notre problème de classification d'images. Chaque élément du vecteur indique la probabilité pour l'image en entrée d'appartenir à une classe [51].

La couche fullyconnected constitue toujours la dernière couche d'un réseau de neurones, convolutif ou non, elle n'est donc pas caractéristique d'un CNN. Ce type de couche reçoit un vecteur en entrée et produit un nouveau vecteur en sortie. Pour cela, elle applique une combinaison linéaire puis éventuellement une fonction d'activation aux valeurs reçues en entrée [51].

2) Les réseaux récurrents

Les (RNN) Réseau de Neurones Récurrents sont des réseaux de neuronesdans lesquels l'information peut se propager d'avant en arrière, y compris descouches profondes aux couches plus en amont. En cela, ils sont plus prochesdu vrai fonctionnement du système nerveux des humains, qui n'est pas à sensunique. Ces réseaux ont des connexions récurrentes dans le sens où ils conserventdes informations en mémoire[28].

Les RNN présentent un caractère dynamique du fait que leurs poids dépendent non seulement des entrées apprises, mais également des sorties précédentes.
Pour ces raisons, les RNN sont particulièrement adaptés aux applications impliquant le contexte, la relation d'ordre des données et plus particulièrement au traitement de séquences temporelles telles que l'apprentissage et la génération de signaux, c'est-à-dire lorsque les données forment une séquence et ne sont pas indépendantes dans leurs successions. D'un point de vue théorique, les RNN ont un potentiel beaucoup plus important que les réseaux neuronaux conventionnels [28].

2.5. Conclusion

En résumé, l'apprentissage automatique constitue aujourd'hui l'un des piliers fondamentaux de l'intelligence artificielle. Il désigne l'ensemble des méthodes permettant aux machines d'apprendre à partir de données, sans être explicitement programmées. On distingue principalement trois types d'apprentissage : supervisé, non supervisé, et par renforcement, chacun adapté à des contextes spécifiques.

Le deeplearning, ou apprentissage profond, représente une branche avancée de l'apprentissage automatique. Il se base sur des architectures complexes de réseaux de neurones artificiels, capables de traiter d'importants volumes de données et d'extraire des représentations hiérarchiques.

Au cœur du deeplearning, les réseaux de neurones convolutifs (CNN) se révèlent particulièrement efficaces pour le traitement d'images, tandis que les réseaux de neurones récurrents (RNN) sont mieux adaptés à l'analyse des données séquentielles comme le texte ou la parole. Ces modèles, bien que techniquement complexes, sont inspirés du fonctionnement du cerveau humain et permettent aujourd'hui des avancées majeures dans de nombreux domaines, tels que la vision par ordinateur, le traitement du langage naturel, ou encore la reconnaissance vocale.

Ce chapitre a ainsi permis de poser les bases nécessaires pour comprendre l'évolution des techniques d'apprentissage automatique et les principes fondamentaux qui régissent les systèmes intelligents modernes.

Chapitre 3

Conception

3.1. Introduction

La conception du système constitue une étape fondamentale dans tout projet d'apprentissage automatique, car elle permet de définir l'architecture globale du processus de traitement et de classification. Ce chapitre joue un rôle central dans l'établissement des bases solides sur lesquelles repose la performance du modèle final. Il permet d'identifier les choix technologiques adaptés, d'anticiper les difficultés potentielles, et de structurer les étapes clés du projet. Cependant, vu le nombre élevé de détails concernant la conception de notre système, une bonne partie de nos choix sont intégrés dans le chapitre implémentation, qui présente pour nous une implémentation et une conception à la fois.

Les objectifs de ce chapitre sont multiples :

- Décrire clairement le problème de classification des sons et motiver l'usage des approches d'apprentissage profond pour le résoudre.
- Présenter les étapes de traitement allant de l'acquisition des données brutes jusqu'à leur transformation en une forme exploitable par un réseau de neurones convolutifs.
- Justifier le choix de l'architecture du modèle de Deep Learning ainsi que les paramètres et techniques employés pour en améliorer la robustesse.

Pour cela, une vue d'ensemble du pipeline est proposée, incluant la collecte des données, leur prétraitement, leur conversion en représentations spectrales pertinentes (spectrogrammes), suivie par la conception d'un modèle CNN sur mesure. Ce processus permet une extraction automatique et efficace des caractéristiques acoustiques, aboutissant à une classification précise des signaux sonores.

3.2. Présentation du Problème et Choix de l'Approche

3.2.1. Définition du problème (classification des sons environnementaux)

La classification des sons environnementaux constitue une tâche complexe en raison de la diversité des sources sonores et des variations qu'elles présentent. Les données audio peuvent inclure des voix humaines, des sons naturels, des instruments de musique ou encore des bruits urbains. Chaque catégorie sonore possède des caractéristiques acoustiques propres, rendant leur identification à la fois intéressante et difficile.

La reconnaissance des sons est un domaine central dans le traitement du signal et l'intelligence artificielle. Elle vise à analyser des séquences audio pour identifier automatiquement les classes sonores auxquelles elles appartiennent. Cette tâche s'applique à de nombreux contextes, notamment la reconnaissance vocale, la surveillance acoustique, ou encore la classification de scènes sonores. Pour cela, il est nécessaire d'extraire des caractéristiques pertinentes du signal et d'utiliser des modèles d'apprentissage capables de généraliser à partir d'exemples annotés.

.

3.2.2. Justification de l'approche Deep Learning

Le Deep Learning s'est imposé comme une approche de référence dans les tâches de classification sonore, grâce à sa capacité à modéliser des structures complexes et à apprendre automatiquement les représentations utiles à partir des données brutes.

Contrairement aux méthodes traditionnelles qui reposent sur l'extraction manuelle de caractéristiques, les modèles profonds tels que les réseaux de neurones convolutifs (CNN) peuvent apprendre directement à partir des représentations spectrales des signaux, comme les spectrogrammes ou les MFCC.

Cette capacité à traiter de grands volumes de données, combinée à une architecture adaptée, rend le Deep Learning particulièrement performant pour la reconnaissance de sons variés dans des environnements bruités ou non structurés.

Dans le domaine de la classification des sons, plusieurs méthodes coexistent, chacune offrant des avantages et des limites spécifiques en fonction du contexte d'application, de la complexité des données et des ressources disponibles.

Le tableau 3.1 fournit une comparaison entre quelques méthodes de classification traditionnelles et les méthodes de classification basées sur le deep learning, où l'objectif est d'explorer les méthodes de deep learning et de voir les résultats qu'elles peuvent fournir en utilisant un nombre bien limité de classes pour le test.

Tableau3. 1 Comparaison entre SVM, HMM et Deep Learning

Méthode	Avantages	Limites
SVM (Support Vector Machines)	Performants avec peu de données, bonne capacité de séparation des classes.	Dépend fortement des caractéristiques extraites manuellement. Faible scalabilité.
HMM (Hiden Marcov Models)	Adaptés à la modélisation temporelle. Utilisés en reconnaissance vocale.	Moins performants avec des sons complexes et bruités. Nécessitent des hypothèses fortes sur la structure du signal.
Deep learning	Apprentissage automatique des caractéristiques. Excellente performance avec données massives.	Coût computationnel élevé. Nécessite de grandes bases de données annotées.

Le deep Learning surpasse généralement ces méthodes dès lors que des données suffisantes sont disponibles et que les ressources matérielles permettent l'entraînement de modèles profonds.

3.2.3. Choix entre CNN à partir de zéro ou CNN pré-entraîné

Le choix entre entraîner un réseau CNN depuis zéro (*from scratch*) ou utiliser un modèle pré-entraîné dépend de plusieurs facteurs : la taille du jeu de données, les ressources disponibles et les objectifs de performance.

- **CNN à partir de zéro** : permet une personnalisation complète de l'architecture en fonction des données spécifiques du projet. Cependant, cette approche nécessite un volume important de données annotées et des ressources matérielles conséquentes.
- **Modèle pré-entraîné (ex. VGGish, SoundNet)** : offre une alternative efficace lorsque les données sont limitées. Ces modèles bénéficient de connaissances acquises sur de vastes corpus et peuvent être ajustés à la tâche cible grâce à l'apprentissage par transfert (*transferlearning*).

En pratique, le choix est souvent un compromis entre flexibilité, coût de calcul, et disponibilité des données. Dans le cadre de ce projet, nous avons opté pour un **modèle CNN personnalisé conçu depuis zéro**, afin de mieux contrôler la structure du réseau et son adaptation aux spectrogrammes extraits. Ensuite, une comparaison est faite avec un **CNN pré-entrainé**.

3.2.4. Présentation des défis

Les défis classification des sons sont liés à la manière dont les signaux sonores sont faits et aux conditions dans lesquelles on les enregistre.

- **Présence de bruit et interférences** : les bruits de fond, la réverbération et les perturbations acoustiques peuvent altérer les signaux.
- **Variabilité des sons** : les différences d'intensité, de durée, de fréquence et de sources rendent la généralisation difficile.
- **Similitudes acoustiques entre classes** : certains sons ont des spectres très proches (ex. : pluie vs robinet, porte qui claque vs coup de feu).
- **Contraintes des données** : jeux de données déséquilibrés, annotations limitées ou bruitées.
- **Conditions d'enregistrement** : qualité du microphone, distance à la source, environnement (intérieur/extérieur).

Ces défis exigent un traitement rigoureux des données et une conception adaptée du modèle pour maximiser la précision de la classification.

3.3. Description du Pipeline de Traitement

3.3.1. Acquisition des données

Source des fichiers audio : Les données audio proviennent généralement de différentes sources, comme des fichiers enregistrés sur des microphones (ex. :

podcasts, interviews, sons de la nature, musique), des bases de données publiques (ex. : UrbanSound8K, LibriSpeech), ou des enregistrements effectués par des appareils spécifiques. Les formats courants de fichiers audio incluent WAV, MP3, FLAC, etc.

Organisation des données : Une fois collectées, les données sont organisées en répertoires et sous-répertoires. Les fichiers audio peuvent être classés par catégories (ex. : bruit de fond, parole, musique, etc.), ou selon des métadonnées telles que la durée de l'enregistrement, le locuteur, ou le type de son. Cela permet de simplifier les étapes suivantes du traitement et l'évaluation des performances du modèle.

3.3.2. Prétraitement des données

Cette étape comprend diverses actions pour nettoyer et normaliser les données afin de les rendre prêtes à l'analyse :

Nettoyage des sons : Le nettoyage audio consiste à enlever ou réduire les bruits indésirables comme les bruits de fond, les clics ou les sifflements. Cela peut être accompli par des techniques comme la suppression de bruit (denoising) et l'utilisation de filtres passe-bas ou passe-haut pour éliminer certaines fréquences indésirables.

Normalisation : Cela implique d'ajuster le volume des enregistrements audio pour que les niveaux sonores soient cohérents à travers tous les fichiers. La normalisation peut être effectuée en ajustant le gain pour que l'amplitude du signal soit uniforme, ce qui évite d'avoir des variations de volume trop importantes.

Segmentation : La segmentation divise les fichiers audio en segments plus petits et plus maniables. Cela peut être fait en fonction des silences ou des événements sonores spécifiques, ou en découpant de longues.

Il est à noter que dans notre cas, les fichiers audio sont bien organisés dans la base de données ESC50 et aucune étape de suppression de bruit n'est effectué ici. Pour cela, nous commençons par un certain nombre de classes de fichiers.wav qui vont être transformés en spectrogrammes comme décrit dans ce qui suit.

3.3.3. Transformation en spectrogramme

Il existe plusieurs types de spectrogrammes utilisés dans la classification des sons. Chaque type permet de représenter le signal d'une manière spécifique afin de mieux en extraire les caractéristiques utiles à l'analyse.

Tableau3. 2 : Types de spectrogrammes

Type de spectrogramme	Caractéristiques	Applications principales
Spectrogramme classique (STFT)	Basé sur la **Transformée de Fourier à Court Terme (STFT)**, avec une échelle de fréquence linéaire.	- Analyse de signaux audio. - Traitement des bruits industriels.
Mel Spectrogramme	Transformation du spectrogramme classique en utilisant l'échelle **Mel**, plus proche de la perception auditive humaine.	- Reconnaissance vocale. - Classification des sons environnementaux.
MFCC (Mel-Frequency Cepstral Coefficients)	Extraction des **coefficients cepstraux** du Mel Spectrogramme via une **Transformée en Cosinus Discrète (DCT)**	- Reconnaissance automatique de la parole. - Classification audio.

- **Justification du choix retenu**

Il existe plusieurs types de représentations spectrales telles que le spectrogramme classique, le Mel spectrogramme et les MFCC.
Dans le cadre de notre projet, nous avons choisi d'utiliser le Mel spectrogramme, car il offre une représentation proche de la perception humaine du son tout en étant particulièrement efficace pour les tâches de classification audio.

3.4. Conception du Modèle de Deep Learning

1. Présentation de l'architecture du CNN

L'architecture du réseau de neurones convolutifs (CNN) repose sur une séquence de couches spécialisées conçues pour extraire et apprendre des caractéristiquesdiscriminantes à partir des données, notamment les spectrogrammes dans notre cas.

1.1. Couches utilisées (Convolution, Pooling, FullyConnected)

- **Couches de Convolution (ConvolutionalLayers)**
  Elles appliquent des filtres (ou noyaux) à l'image d'entrée afin d'extraire des caractéristiques locales comme les bords, les formes ou les textures. Chaque filtre glisse sur l'image et génère une carte de caractéristiques (*featuremap)* représentant une caractéristique spécifique.
- **Couches de Pooling (PoolingLayers)**
  Elles réduisent la dimensionnalité des *featuremaps*, limitant ainsi le nombre de paramètres et la charge computationnelle.
  - **Max Pooling** : conserve la valeur maximale dans chaque région.
  - **AveragePooling** : calcule la moyenne des valeurs.

Dans notre modèle, nous avons opté pour le **MaxPooling** afin de conserver les caractéristiques les plus saillantes tout en réduisant la dimension des cartes de caractéristiques.

- **Couches entièrement connectées (FullyConnectedLayers – FC)**
  À la fin du réseau, ces couches connectent tous les neurones entre eux pour effectuer la classification. Elles prennent les caractéristiques extraites et les transforment en sorties interprétables, comme des classes.

1.2. Vue globale de l'architecture du CNN

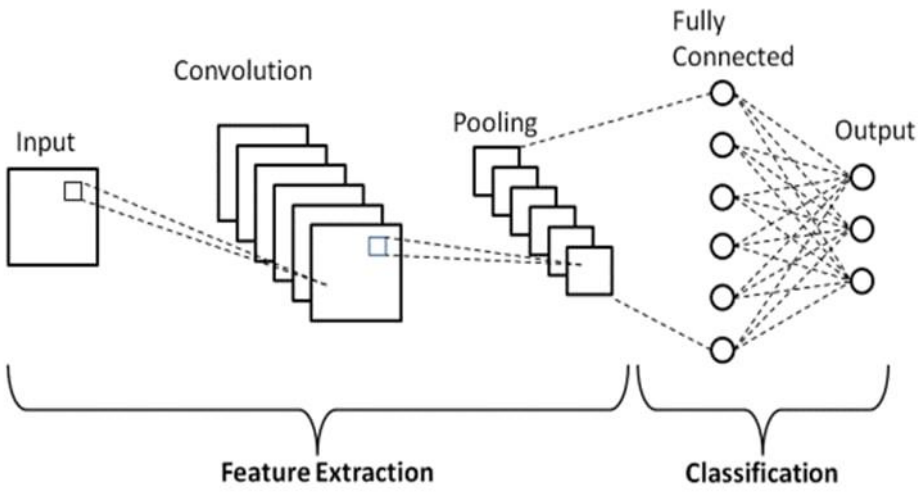


Figure3. 1: Architecture d'un CNN [51]

Explication du schéma :

Ce schéma représente une architecture CNN typique, où :

- Les couches convolutionnelles extraient progressivement les caractéristiques.
- Les activations ReLU ajoutent de la non-linéarité.
- Le pooling réduit les dimensions.
- Le flatten transforme les données en vecteur.
- Les couches FC produisent la classification finale via Softmax (ou Sigmoïde selon le type de sortie).

2. Paramètres et Fonctions utilisées

2.1 Fonction d'activation

Les fonctions d'activation jouent un rôle fondamental dans les réseaux de neurones, car elles introduisent de la non-linéarité, permettant au modèle d'apprendre des représentations complexes.
Elles sont généralement appliquées après chaque couche convolutive ou couche entièrement connectée (couches intermédiaires), sauf dans la couche de sortie, où

l'on choisit une fonction d'activation adaptée au type de tâche (classification binaire ou multiclasse).

- **ReLU (RectifiedLinear Unit)** : très utilisée pour introduire de la non-linéarité ; transforme toute valeur négative en zéro, ce qui favorise une convergence rapide.

$$f(x)=\max(0, x)$$

- **Sigmoïde** : utilisée en sortie dans le cas de la classification binaire,car elle donne une sortie comprise entre 0 et 1, interprétée comme une probabilité.

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

- **Soft max** : Utilisée en sortie pour **les tâches de classification multi classe**. Elle transforme le vecteur de sortie en une distribution de probabilité sur les classes.

$$F(xi) = \frac{ex}{\Sigma(e\text{^}(x\_j))}$$

Avec la somme Σ allant de j = 1 à n.

2.2 Fonction de perte

Dans les réseaux de neurones, la fonction de perte sert à évaluer la performance du modèle pendant l'entraînement en mesurant l'écart entre les prédictions et les valeurs réelles.

Dans notre cas, il s'agit d'un problème de classification**,**donc nous utilisons principalement, **la fonction Cross-EntropyLoss.**

- **Cross-EntropyLoss (Entropie croisée)** :
  Appropriée pour la classification, elle mesure la différence entre la distribution prédite et la distribution réelle des classes.

$$L = - \Sigma (y\_i * \log(\hat{y}\_i))$$

Avec la somme Σ allant de i = 1 à n.

- **MeanSquaredError (MSE)** :

  Utilisée pour les tâches de régression, elle mesure la moyenne des carrés des écarts entre les valeurs prédites et les valeurs cibles.

$$MSE = (1/n) * \Sigma (y\_i - \hat{y}\_i)^2$$

Avec la somme Σ allant de i = 1 à n.

2.3 Optimiseurs

Les optimiseurs ajustent les poids du réseau durant l'apprentissage en se basant sur le gradient de la fonction de perte. Le choix d'un optimiseur influence fortement la vitesse et la stabilité de convergence.

Exemples d'optimiseurs populaires :

- **Adam (Adaptive Moment Estimation)** :
  Très populaire, il ajuste dynamiquement le taux d'apprentissage en tenant compte des moments d'ordre 1 et 2 du gradient.
- **SGD (Stochastic Gradient Descent)** :
  Optimiseur classique basé sur la descente de gradient avec mini-batchs.
- **RMSprop** :
  Divise le gradient par une moyenne glissante des carrés des gradients, stabilisant ainsi l'apprentissage.

Dans ce travail, nous avons choisi d'utiliser **l'optimiseur Adam** car il est efficace, stable et permet une convergence rapide même avec un réseau de petite taille entraîné à partir de zéro.

3. Mécanismes d'amélioration du modèle

Pour améliorer la performance et la généralisation du modèle, plusieurs techniques peuvent être appliquées :

- **Data Augmentation** :
  Génération d'exemples supplémentaires en modifiant légèrement les données originales (changement d'échelle, bruit, translation…).
- **Dropout** :
  Technique de régularisation qui consiste à désactiver aléatoirement certains neurones pendant l'entraînement pour éviter le sur-apprentissage.
- **Régularisation L2 (ou L1)** :
  Ajout d'une pénalité sur les poids du modèle dans la fonction de perte pour éviter des valeurs trop grandes (ce qui réduit le risque de sur-apprentissage).

Voici un Diagramme de classes UML simplifié pour un système de classification de **classes de sons environnementaux** utilisant **réseau de neurones convolutif (CNN)**.

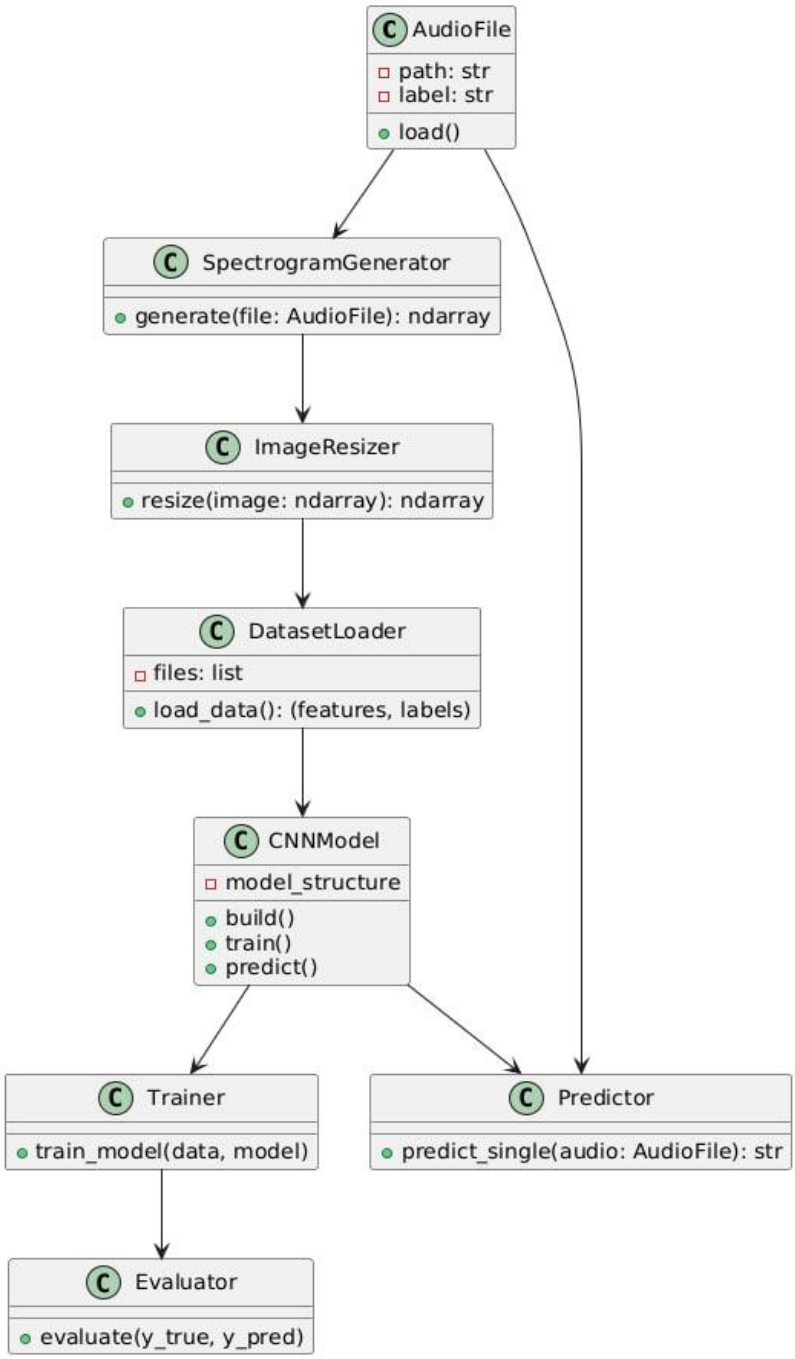


Figure3. 2 Diagramme de classe UML pour un système de classification sonore utilisant un CNN

L'architecture générale du système de classification des sons est illustrée dans la figure 3.3 suivante :

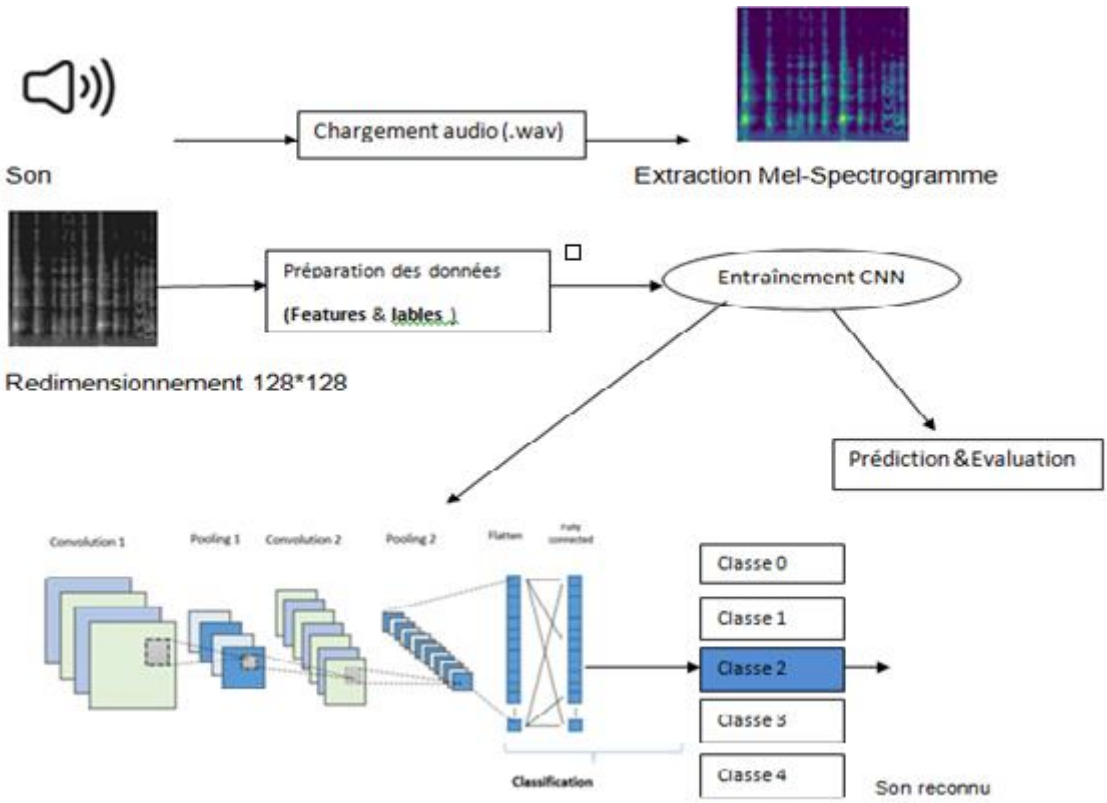


Figure3. 3: Architecture Générale du Système de Classification Sonore.

3.5. Conclusion

Dans ce chapitre, nous avons présenté de manière structurée les fondements de la conception du système de classification sonore basé sur le Deep Learning. Nous avons commencé par définir clairement le problème et expliquer pourquoi les réseaux de neurones convolutifs (CNN) représentent une solution adéquate, notamment par rapport à d'autres approches traditionnelles comme les SVM ou les HMM.

L'ensemble du pipeline a été détaillé, depuis l'acquisition et le nettoyage des données jusqu'à leur transformation en représentations spectrales. Le choix du type de spectrogramme utilisé a été motivé par ses performances dans la mise en évidence des caractéristiques acoustiques discriminantes. Enfin, la conception du modèle CNN a été justifiée par des arguments techniques et méthodologiques, en incluant des mécanismes d'amélioration pour éviter le surapprentissage.Ces choix techniques, bien que guidés par la littérature et les besoins du projet, seront évalués empiriquement dans les chapitres suivants à travers une phase d'implémentation et d'expérimentation rigoureuse.

Toute lacune dans le chapitre conception est bien comblée dans le chapitre implémentation, qui décrit tous les détails de notre démarche dans la mise en œuvre de notre système de classification des sons environnementaux.

Chapitre 4
Implémentation

4.1. Introduction

Ce chapitre décrit en détail la mise en œuvre du modèle de classification des sons basé sur un réseau de neurones convolutif (CNN). L'objectif est de présenter les différentes étapes de l'implémentation, du prétraitement des données à l'entraînement du modèle, ainsi que l'évaluation des performances et les expérimentations réalisées.

Nous commencerons par expliquer le pipeline de traitement des données, qui inclut le chargement et la transformation des fichiers audio en représentations exploitables par un CNN. Ensuite, nous détaillerons l'architecture du modèle, son entraînement et les techniques utilisées pour optimiser ses performances. Une évaluation rigoureuse sera menée à l'aide de métriques standards, suivie d'expérimentations visant à analyser l'impact de différentes configurations et approches. Enfin, nous conclurons par une synthèse des résultats et des perspectives d'amélioration possibles.

Dans cette étude, nous avons utilisé la base de données ESC-50, qui est une collection librement accessible contenant 2000 fichiers audioclassés en 50 catégories environnementales (sons humains, animaux, sons naturels, urbains, etc.). Elle est largement utilisée pour les tâches de classification audio dans le domaine de l'apprentissage profond, et constitue une référence académique pour l'évaluation des modèles de reconnaissance sonore.

Tous les traitements et expérimentations ont été réalisés dans l'environnement de développement Visual Studio Code (VS Code), en utilisant le langage Python et des bibliothèques spécialisées dans le traitement du signal et l'apprentissage profond.

Pour l'implémentation, plusieurs outils et bibliothèques ont été mobilisés, principalement en Python. Les plus importantes sont :
Librosa: pour le chargement des fichiers audio et l'extraction des caractéristiques (Mel-Spectrogram, MFCC...).
NumPy et Pandas : pour la manipulation des données sous forme de tableaux.
Matplotlib et Seaborn : pour la visualisation des spectrogrammes et des résultats.
OpenCVet Pillow : pour la manipulation d'images spectrales.
PyTorch : pour la conception, l'entraînement et l'évaluation du modèle CNN.
scikit-learn : pour le partitionnement des données et le calcul des métriques de performance.

Ces outils ont permis d'assurer une implémentation modulaire, reproductible et efficace du système de classification.

4.2. Implémentation du Pipeline de Traitement

4.2.1. Chargement et prétraitement des données

Dans cette section, nous allons expliquer comment les fichiers audio sont chargés et préparés avant d'être utilisés dans le modèle de classification. Le prétraitement est une étape essentielle dans l'apprentissage automatique, car il permet d'extraire des informations exploitables à partir des fichiers bruts. Le pipeline de prétraitement suit ces étapes :

- **Chargement des fichiers audio** depuis le jeu de données **ESC-50**.
- **Application des transformations** pour obtenir des spectrogrammes exploitables par le modèle de classification

a. Chargement des fichiers audio

Dans cette étape, nous avons pour objectif de charger les fichiers audio issus de la base de données ESC-50. Il s'agit d'une base bien connue dans le domaine de la reconnaissance de sons, contenant 2000 extraits audio d'une durée de 5 secondes chacun, répartis équitablement en 50 classes couvrant des sons humains, naturels, urbains et animaux. Les fichiers sont organisés en 5 plis (folds), ce qui facilite la gestion des données pour l'entraînement croisé (cross-validation) ou le découpage des ensembles (train, validation, test).

Le chargement correct de ces fichiers constitue une étape fondamentale, car il conditionne la qualité des transformations ultérieures (telles que les spectrogrammes, les mel-spectrogrammes et les MFCC), qui sont indispensables pour représenter les signaux audio sous une forme exploitable par un réseau de neurones convolutif (CNN).

Code : load_audio.py

```
importos
importlibrosa
importnumpyasnp

# Définir le chemin du dataset
dataset_path="dataset"
# Vérifier si le dossier existe
ifnotos.path.exists(dataset_path):
    raiseFileNotFoundError(f"Le dossier {dataset_path} n'existe pas.")

# Fonction pour charger un fichier audio
defload_audio(file_path, sr=22050):
    audio, sample_rate=librosa.load(file_path, sr=sr)  # Charger le fichier
audio
    returnaudio, sample_rate
```

```
# Charger tous les fichiers audio
audio_data= {}
forfileinos.listdir(dataset_path):
    iffile.endswith(".wav"):
        file_path=os.path.join(dataset_path, file)
        audio, sr=load_audio(file_path)
        audio_data[file] = (audio, sr)

# Vérifier un échantillon

sample_file=list(audio_data.keys())[0]
print(f"Exemple de fichier chargé : {sample_file}")
print(f"Taille du signal : {len(audio_data[sample_file][0])}")
print(f"Taux d'échantillonnage : {audio_data[sample_file][1]}")
```

Le processus commence par la définition du **chemin d'accès au dossier dataset**, lequel contient l'ensemble des fichiers audio au format **.wav**. Une fonction nommée load_audio est ensuite utilisée pour **charger ces fichiers à l'aide de la bibliothèque Librosa**, en fixant la **fréquence d'échantillonnage à 22050 Hz**. Une boucle permet alors de parcourir tous les fichiers du dossier et de **stocker dans un dictionnaire uniquement ceux qui sont au format .wav.**

Bien que les fichiers de la base **ESC-50** soient **physiquement regroupés dans un même dossier sans structure hiérarchique**, ils sont **logiquement répartis en 5 plis (folds)** selon un **schéma de validation croisée** défini dans un fichier CSV (esc50.csv). Cette organisation facilite l'application d'une **validation croisée à 5 plis** lors des expérimentations. Enfin, un exemple de fichier est affiché pour **vérifier la taille du signal audio ainsi que la fréquence d'échantillonnage utilisée**.Après exécution, la sortie attendue est illustrée dans la figure suivante :

```
Exemple de fichier chargé : 1-100032-A-0.wav
Taille du signal : 110250
Taux d'échantillonnage : 22050
```

Figure 4. 1 Résultats du chargement des fichiers audio

Le script affiche le nom du premier fichier chargé, la longueur du signal (nombre d'échantillons) et la fréquence d'échantillonnage, confirmant que le chargement s'est effectué correctement.

b. Application des transformations (spectrogrammes)

Dans cette étape, nous avons transformé les fichiers audio de la base ESC-50 en représentations visuelles à l'aide des **Mel-Spectrogrammes**, une technique de transformation spectrale adaptée à la perception auditive humaine. Cette représentation constitue une étape essentielle pour convertir les données audio en un format exploitable par un modèle CNN pour la classification.

Nous avons utilisé 40 fichiers audio par classe, qui est le nombre des échantillons de chaque classe, et nous avons sélectionné uniquement 5 catégories parmi celles disponibles dans la base ESC-50. Ce choix est motivé par deux raisons principales. D'une part, il permet de réduire le volume de données à traiter, ce qui est cohérent avec les contraintes de calcul et de mémoire liées à l'environnement de développement. D'autre part, cela nous permet de tester l'efficacité de notre modèle CNN dans un contexte de données limitées, ce qui reflète des situations réelles dans lesquelles les bases de données annotées sont restreintes. Ce paramétrage simplifié a également facilité l'analyse expérimentale tout en conservant une diversité sonore suffisante pour valider notre approche.

- **Code : transformations.py**

```
importos
importlibrosa
importlibrosa.display
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

# 🔧 Paramètres
dataset_path ="dataset"
metadata_path ="ESC-50-master/meta/esc50.csv"
output_path ="spectrograms"
nb_files_per_category =40
nb_top_categories =5

# Création du dossier de sortie
os.makedirs(output_path, exist_ok=True)

# Nettoyage du dossier de sortie
for f in os.listdir(output_path):
    os.remove(os.path.join(output_path, f))

# Lecture des métadonnées
df = pd.read_csv(metadata_path)

# 🔍 Trouver les 5 catégories avec le plus grand nombre de fichiers
category_counts = df['category'].value_counts()
top_categories                                                    =
category_counts.head(nb_top_categories).index.tolist()
print(f"📊 Top {nb_top_categories} catégories les plus fréquentes :
{top_categories}")

# Sélectionner les fichiers correspondants
df_selected = pd.DataFrame()
```

```
forcategoryin top_categories:
                    subset         =         df[df["category"]         ==
category].head(nb_files_per_category)
    df_selected = pd.concat([df_selected, subset])

# Fonction de génération du Mel-Spectrogram
defgenerate_mel_spectrogram(file_path, file_name, category):
    y, sr = librosa.load(file_path, sr=22050)
    mel_spec = librosa.feature.melspectrogram(y=y, sr=sr, n_mels=128)
    mel_spec_db = librosa.power_to_db(mel_spec, ref=np.max)

    plt.figure(figsize=(3, 3))
    librosa.display.specshow(mel_spec_db, sr=sr, cmap='viridis')
    plt.axis('off')

                  output_file        =        os.path.join(output_path,
f"{category}_{os.path.splitext(file_name)[0]}_Mel.png")
    plt.savefig(output_file, bbox_inches='tight', pad_inches=0)
    plt.close()

# Application
for _, row in df_selected.iterrows():
    file_name = row["filename"]
    category = row["category"]
    file_path = os.path.join(dataset_path, file_name)
    if os.path.exists(file_path):
        generate_mel_spectrogram(file_path, file_name, category)

print(f"✅ Générés : {len(df_selected)} Mel-Spectrograms pour
{nb_top_categories} catégories.")
```

Le script débute par la définition des chemins nécessaires, notamment vers les fichiers audio, le fichier des métadonnées et le dossier de sortie destiné à contenir les spectrogrammes générés. Cette étape permet de centraliser les ressources et de structurer l'environnement de travail. Ensuite, les métadonnées sont chargées à partir d'un fichier CSV. Ce fichier contient des informations importantes sur les échantillons sonores, telles que leur nom, leur catégorie, et leur appartenanceà un pli (*fold*),ce qui facilite leur traitement ultérieur.

Le script sélectionne ensuite un nombre fixe d'échantillons audio, soit 40 fichiers par pli. Cette opération vise à équilibrer la distribution des données entre les différentes classes et à garantir une quantité suffisante d'exemples pour l'entraînement du modèle. Afin de générer les spectrogrammes, une vérification est effectuée pour s'assurer de l'existence de tous les fichiers audio nécessaires. Cette étape de validation permet d'éviter les erreurs lors du traitement en signalant tout fichier manquant.

Si l'ensemble des fichiers est disponible, la fonction generate_mel_spectrogram() est appliquée à chaque fichier sélectionné. Cette fonction réalise plusieurs opérations : elle charge le fichier audio, calcule le Melspectrogramme, applique une conversion en décibels pour une meilleure représentation visuelle, puis sauvegarde le spectrogramme obtenu sous forme d'image PNG.

Enfin, le script affiche un récapitulatif indiquant le nombre total d'images générées, ainsi que des statistiques par catégorie. Cela permet de vérifier que le traitement a bien couvert l'ensemble des classes sonores de manière équilibrée. La figure suivante illustre le résultat d'exécution**.**

En effet, nous avons utilisé une procédure automatique qui permet de sélectionner les classes ayant le plus grand nombre des échantillons. Cependant, vu que la base utilisée contient un nombre constant des échantillons par classe, cette procédure retourne en sortie un nombre arbitraire de classes selon leur ordre. Enfin, les sons que nous avons utilisé pour le test de notre application sont les suivants : sons d'aboiement des chiens, les bris de verre, boire en sirotant, pluie, et le son des insectes.

Exemple de fichiers générés dans spectrograms/ :

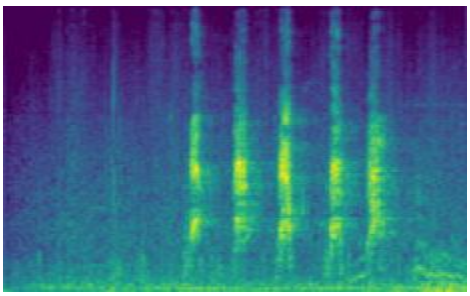


Figure 4. 2 Mel- spectrogramme du son d'aboiement d'unchien

Au total, **200 images** ont été générées, correspondant à**40 fichiers audio par pli**, répartis sur **5 folds**. Chaque image représente le **Mel-spectrogramme** d'un fichier audio individuel, offrant ainsi une représentation visuelle de son contenu fréquentiel. À la fin du processus, une **statistique par catégorie** a été produite, illustrant la **distribution des classes sonores** au sein de l'ensemble des données générées.

4.2.2. Format des données d'entrée du CNN

Avant d'entraîner notre réseau de neurones convolutionnel (CNN), nous devons formater les données correctement. Cette étape comprend la définition de la taille des images spectrogrammes pour assurer une entrée cohérente au modèle, ainsi qu'un encodage des labels pour transformer les catégories en un format compréhensible par l'algorithme.A la fin,une conversion en matrices NumPy et sauvegarde.

Ces étapes sont essentielles, car un CNN traite des images de dimensions fixes et fonctionne mieux avec des labels numériques bien organisés.

a. Taille des images spectrogrammes

Après la conversion des fichiers audio en images Mel-Spectrogrammes, nous avons standardisé la taille de ces images afin de les utiliser comme entrée pour notre réseau CNN. La taille adoptée est de **(128, 128)** pixels, résultant du paramètre n_mels = 128 et d'un contrôle sur la durée temporelle de l'image. Ce format permet de :

- Normaliser les dimensions des données.
- Faciliter l'apprentissage pour le modèle CNN.
- Réduire le coût mémoire et le temps de traitement.

➢ **Code : resize_spectrograms.py**

```
importos
importcv2

# Dossiers
input_path="spectrograms"              # Images d'origine (Mel)
output_path="resized_spectrograms"     # Images après redimensionnement

# Créer le dossier de sortie s'il n'existe pas
os.makedirs(output_path, exist_ok=True)

# Nettoyage du dossier de sortie avant de commencer
print(f"🗑 Nettoyage du dossier '{output_path}' avant de commencer...")
forfinos.listdir(output_path):
    file_path=os.path.join(output_path, f)
    ifos.path.isfile(file_path):
        os.remove(file_path)

# Taille cible pour le CNN
target_size= (128, 128)

defresize_image(file_name):
    img_path=os.path.join(input_path, file_name)
    image=cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)

    ifimageisNone:
        print(f"❌ Erreur lors de la lecture de l'image : {file_name}")
        return

    resized_image=cv2.resize(image, target_size)
    output_file=os.path.join(output_path, file_name)  # Conserver le même
nom
    cv2.imwrite(output_file, resized_image)

# Redimensionner toutes les images contenant "_Mel.png" dans leur nom
forfile_nameinos.listdir(input_path):
```

```
    iffile_name.endswith("_Mel.png"):
        resize_image(file_name)

print("✅ Redimensionnement terminé pour tous les fichiers du type : label-nom_Mel.png")
```

Nous avons utilisé la bibliothèque OpenCV pour lire et redimensionner chaque image Mel spectrogramme en niveaux de gris. Ces images étaient auparavant générées à partir des fichiers audio du dataset **ESC-50**. Le script parcourt toutes les images dans le dossier spectrograms/, les redimensionne, puis les sauvegarde dans un nouveau dossier nommé resized_spectrograms/. Le résultat d'exécution est donnée dans la firgure 4.3.

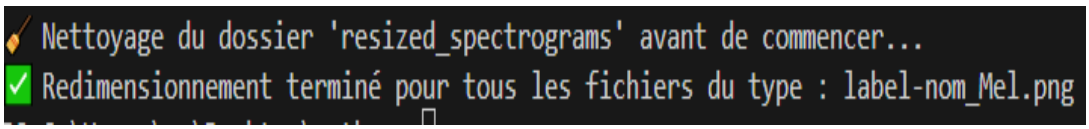


Figure 4. 3 Résultats redimensionné Mel-spectrogrammes

Toutes les images de spectrogrammes sont maintenant de taille **128x128** pixels, stockées dans resized_spectrograms/. Ce format uniforme permettra une intégration fluide dans le pipeline d'entraînement du CNN.La figure 4.4 ci-dessous montre un exemple de fichiers générés dans le répertoire resized_spectrograms/ :

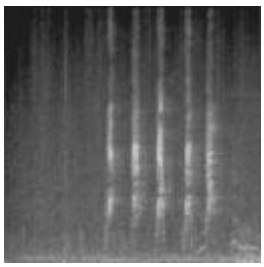


Figure 4. 4Mel- spectrogramme du son d'aboiement d'unchien après redimensionnement.

b. Encodage des labels

L'objectif de cette étape est de convertir les noms de catégories (par exemple « chien », « horloge », « klaxon) en étiquettes numériques afin de rendre les classes exploitables par un réseau de neurones. En effet, un CNN ne peut traiter que des valeurs numériques. Nous avons donc procédé à un encodage des catégories à partir des noms des fichiers de spectrogrammes générés.

- **Code encode_lables.py**

```
importos
importpandasaspd

# 📂 Dossier contenant les spectrogrammes
spectrograms_folder="spectrograms"

# 📜 Liste pour stocker les données
data= []

# Parcourir tous les fichiers dans le dossier
forfile_nameinos.listdir(spectrograms_folder):
    iffile_name.endswith("_Mel.png"):
        try:
            # Extraire le nom complet de la catégorie à partir du nom de
fichier
                    #  Le  nom  du  fichier  est  sous  la  forme  :
catégorie_nomFichier_Mel.png
              # On utilise rsplit pour séparer à partir de la fin et
récupérer la catégorie complète
             category=file_name.rsplit("_", 2)[0]  # Prend tout avant les
deux derniers underscores
            data.append({"filename": file_name, "category": category})
        exceptExceptionase:
            print(f"Erreur dans le traitement de {file_name} : {e}")

# Convertir la liste en DataFrame pandas
df=pd.DataFrame(data)

# Obtenir la liste triée des catégories uniques
unique_categories=sorted(df["category"].unique())

# Créer un dictionnaire de mapping catégorie -> label numérique
category_to_label= {cat: idxforidx, catinenumerate(unique_categories)}

# Appliquer le mapping sur le DataFrame
df["label_encoded"] =df["category"].map(category_to_label)

# Sauvegarder dans un fichier CSV
df.to_csv("image_labels.csv", index=False)

print("✅Fichier image_labels.csv généré avec succès !")
print(f"📌 Catégories encodées : {category_to_label}")
```

Ce script parcourt les fichiers d'images générées (Mel-Spectrograms) situés dans le dossier spectrograms. À partir de leur nom, il extrait la catégorie (le label initial) en récupérant tout ce qui précède les deux derniers underscores. Ensuite, il crée une

liste des catégories uniques et leur assigne un entier unique (0, 1, 2, ...), créant ainsi un **encodage numérique** pour chaque classe.

Finalement, il enregistre un fichier CSV contenant trois colonnes : le nom de fichier, la catégorie d'origine, et le label numérique correspondant. Le résultat d'exécution est illustré dans la figure 4.5.

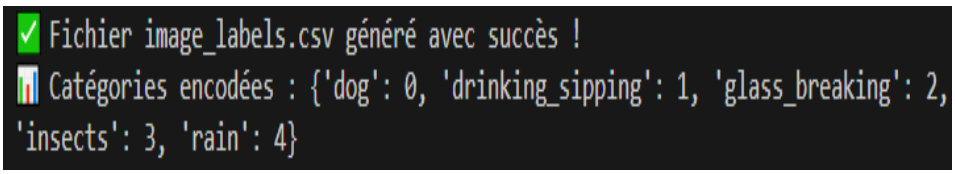


Figure 4. 5 Résultats encodage numériquepour chaque classe.

Enfin, Le fichier **image_labels.csv** a été créé avec succès. Il contient les **noms des images**, les **catégories textuelles** ainsi que leurs **encodages numériques**.

c. Conversion en matrices NumPy et sauvegarde

Dans cette étape, l'objectif est de **convertir les images des Mel-Spectrogrammes redimensionnées** en **matrices numériques (tableaux NumPy)**. Cette conversion est essentielle pour que les données puissent être exploitées comme entrées dans un réseau de neurones convolutif (CNN). Une fois converties, ces données sont **sauvegardées dans des fichiers.npy** afin d'être utilisées ultérieurement lors de l'entraînement ou de l'évaluation du modèle.

Objectif principal : Préparer les données d'entrée sous forme de tableaux numériques (features), accompagnés de leurs étiquettes numériques correspondantes (labels).

- **Code features_lables.py**

```
importos
importnumpyasnp
fromPILimportImage
importpandasaspd
importmatplotlib.pyplotasplt

# المسار إلى مجلد الصور الموحدة الحجم
spectrograms_folder="resized_spectrograms"

# تحميل ملف CSV الذي يحتوي أسماء الصور والتسميات الرقمية
df=pd.read_csv("image_labels.csv")
```

```
features= []
labels= []

# 📂 تحميل الصور وتحويلها إلى مصفوفات NumPy
forindex, rowindf.iterrows():
    file_path=os.path.join(spectrograms_folder, row['filename'])

    try:
        img=Image.open(file_path).convert('L')  # تحويل إلى grayscale
        img_array=np.array(img)
        features.append(img_array)
        labels.append(row['label_encoded'])
    exceptExceptionase:
        print(f"❌Erreur lors du chargement de {file_path}: {e}")

# 📊 تحويل القوائم إلى مصفوفات NumPy
features=np.array(features)
labels=np.array(labels)

# ✅ عرض معلومات أساسية
print(f"📐 features.shape = {features.shape}")
print(f"📐 labels.shape = {labels.shape}")
print(f"✅ Type check: features = {type(features)}, labels =
{type(labels)}")

# ✅ عرض اسم الصورة الأولى
first_filename=df.iloc[0]['filename']
first_label=df.iloc[0]['label_encoded']

print(f"📄nom premier image: {first_filename}")
print(f"🏷️التصنيف الرقمي: {first_label}")

# ✅ طباعة المصفوفة نفسها (محتوى الصورة)
print(" contenu premier matrice features[0]:")
print(features[0])


# 💾 حفظ المصفوفات في ملفات npy للاستخدام لاحقًا
np.save('features.npy', features)
np.save('labels.npy', labels)
```

```
print("✅Features and labels saved successfully in .npy format")

# 🔹 حفظ جميع مصفوفات الصور في ملف نصي لعرضها نصيًا
withopen("all_features_matrix.txt", "w") asf:
    fori, matrixinenumerate(features):
        filename=df.iloc[i]['filename']  # اسم الصورة المرتبطة بالمصفوفة
        label=labels[i]
           f.write(f"--- Feature {i} (Filename: {filename}, Label:
{label}) ---\n")
        np.savetxt(f, matrix, fmt='%d')
        f.write("\n")
  # فاصل بين المصفوفات
print("📁 All image matrices saved in all_features_matrix.txt")
```

Le script présenté a pour but de convertir les images des spectrogrammes Mel redimensionnées en matrices numériques exploitables par un modèle de Deep Learning. Pour ce faire, il commence par charger le fichier image_labels.csv, qui contient les noms de fichiers image ainsi que leurs étiquettes numériques associées. Ensuite, il parcourt chaque ligne de ce fichier pour accéder aux images correspondantes dans le dossier resized_spectrograms. Chaque image est ouverte en mode niveaux de gris à l'aide de la bibliothèque PIL, puis transformée en matrice NumPy grâce à la fonction np.array().

Ces matrices sont ajoutées à une liste nommée features, tandis que les étiquettes correspondantes sont stockées dans une autre liste appelée labels. À la fin de ce processus, les deux listes sont converties en tableaux NumPy, ce qui permet un traitement plus rapide et plus efficace lors de l'entraînement du modèle. Ces tableaux sont ensuite sauvegardés dans des fichiers.npy (features.npy et labels.npy) pour une réutilisation ultérieure. Par ailleurs, un fichier texte all_features_matrix.txt est généré, contenant l'ensemble des matrices d'images accompagnées de leurs noms de fichiers et de leurs labels. Ce fichier permet une visualisation brute des données numériques correspondant à chaque image, ce qui est utile à des fins de vérification ou de documentation.

- **Résultat d'exécution**

L'exécution du script devrait générer les sorties suivantes dans le terminal :

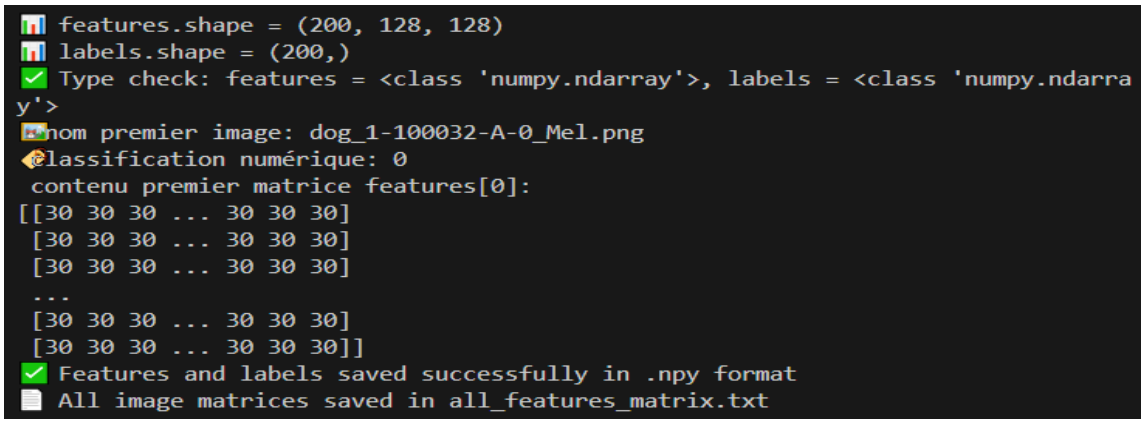


Figure 4. 6convertir images Mel spectrogrammes en matrices numériques

Cela signifie que les données d'entrée sont prêtes à être utilisées dans l'entraînement du modèle CNN, sous forme normalisée et encodée.

4.3. Implémentation du Modèle de Classification

4.3.1. Construction du CNN

Définition des couches du modèle : Cette étape consiste à construire l'architecture du réseau de neurones convolutionnel (CNN). On y définit les différentes couches, telles que :

- Les couches convolutionnelles pour l'extraction des caractéristiques.
- Les couches de normalisation (Batch Normalization) pour stabiliser l'apprentissage.
- Les couches de poolingpour réduire la dimensionnalité.
- Les **couches entièrement connectées (fully connected)** pour la classification finale.

Ces composants sont organisés de manière à permettre au modèle d'apprendre des représentations pertinentes à partir des données audio transformées.

Compilation et choix des hyperparamètres : Après la construction du modèle, il faut procéder à sa compilation. Cela inclut : la définition de la fonction de perte (telle que CrossEntropyLoss),le choix de l'optimiseur (comme Adam ou SGD),la fixation du taux d'apprentissage (learning rate), et éventuellement, l'ajout de paramètres de régularisation.Ces hyperparamètres influencent à la fois la qualité de l'apprentissage et la vitesse deconvergence du modèle.

L'objectif est de concevoir un modèle efficace, capable de capturer des représentationssignificatives à partir des données audio, tout en assurant un apprentissage stable etperformant grâce à un bon réglage des hyperparamètres.

4.3.2. Entraînement du modèle

- **Stratégie de partitionnement**

Il est essentiel de diviser les données en trois sous-ensembles : Ensemble d'entraînement (train) pour ajuster les paramètres du modèle, ensemble de validation (val) pour surveiller les performances en cours d'apprentissage, ensemble de test (test) pour l'évaluation finale. Ce partitionnement permet dedétecterle surapprentissage et de garantir une évaluation juste du modèle.

- **Nombre d'époques et taille de lot (batch size)**

Le nombre d'époques détermine combien de fois le modèle voit l'ensemble des données d'entraînement. La **taille des lots** (batch size) influence la stabilité du gradient et la vitesse d'entraînement. Un bon équilibre entre ces deux paramètres permet un apprentissage plus efficace.

- **Gestion du surapprentissage**

Pour éviter que le modèle **ne mémorise trop les données d'entraînement** au détriment de la généralisation, on applique des techniques telles que :

- **Early stopping** : arrêter l'entraînement si la performance de validation stagne.
- **Dropout** : désactiver aléatoirement certains neurones pendant l'entraînement.
- **Batch Normalization** : stabiliser la distribution des activations.

Ces techniques permettent d'améliorer la capacité du modèle à généraliser sur des données nouvelles. L'objectif est de former un modèle performant, tout en préservant sa capacité de généralisation, grâce à une gestion rigoureuse de l'entraînement et du surapprentissage.

4.3.3. Évaluation des Performances

- **Métriques utilisées**

Pour évaluer le modèle, on utilise plusieurs **mesures quantitatives**, telles que :**Accuracy** (taux de bonne classification),**F1-score** (moyenne harmonique entre précision et rappel),**Matrice de confusion** (visualisation des erreurs de classification par classe).

- **Analyse des résultats**

L'analyse des métriques permet d'**identifier les classes bien ou mal reconnues**, **repérer les éventuelles confusions** entre catégories proches, et **proposer des pistes d'amélioration** du modèle (rééquilibrage des données, ajustement des paramètres, etc.).

Valider les performances obtenues par le modèle et **guider les futures améliorations** en fonction des résultats mesurés sur des données réelles.

- **Code train_model.py**

```
import numpy as np
import random
import torch
import torch.nn as nn
import torch.optim as optim
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score,
confusion_matrix
from torch.utils.data import TensorDataset, DataLoader
import matplotlib.pyplot as plt
import seaborn as sns
from torchsummary import summary

# ----- تثبيت العشوائية لضمان نفس التقسيم في كل تشغيل -----
seed = 42
np.random.seed(seed)
random.seed(seed)
torch.manual_seed(seed)
torch.cuda.manual_seed(seed)
torch.backends.cudnn.deterministic = True
torch.backends.cudnn.benchmark = False

# 1. تحميل البيانات
features = np.load('features.npy')  # shape: (1200, 128, 128)
labels = np.load('labels.npy')      # shape: (1200,)

#2. تحضير البيانات للـ PyTorch
features = features[:, np.newaxis, :, :]  # shape: (1200, 1, 128, 128)
X = torch.tensor(features, dtype=torch.float32)
y = torch.tensor(labels, dtype=torch.long)

# 3. تقسيم البيانات مع تثبيت random_state
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=seed, stratify=y
)
X_train, X_val, y_train, y_val = train_test_split(
X_train, y_train, test_size=0.2, random_state=seed, stratify=y_train
)

# 4. DataLoaders
batch_size = 16
train_loader = DataLoader(TensorDataset(X_train, y_train),
batch_size=batch_size, shuffle=True)
```

```
val_loader = DataLoader(TensorDataset(X_val, y_val), batch_size=batch_size,
shuffle=False)
test_loader        =        DataLoader(TensorDataset(X_test,        y_test),
batch_size=batch_size, shuffle=False)

# 5. نموذج CNN محسّن مع BatchNorm و Dropout أقوى
class ImprovedCNN(nn.Module):
    def __init__(self, num_classes=5):
        super(ImprovedCNN, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, kernel_size=3, padding=1)
        self.bn1 = nn.BatchNorm2d(32)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=3, padding=1)
        self.bn2 = nn.BatchNorm2d(64)
        self.conv3 = nn.Conv2d(64, 128, kernel_size=3, padding=1)
        self.bn3 = nn.BatchNorm2d(128)
        self.pool = nn.MaxPool2d(2, 2)
self.dropout = nn.Dropout(0.5)

        # بعد 3 عمليات pooling16 حجم الخرائط يصبح x16،
self.fc1 = nn.Linear(128 * 16 * 16, 256)
        self.fc2 = nn.Linear(256, num_classes)

    def forward(self, x):
        x = self.pool(torch.relu(self.bn1(self.conv1(x))))
        x = self.pool(torch.relu(self.bn2(self.conv2(x))))
        x = self.pool(torch.relu(self.bn3(self.conv3(x))))
        x = x.view(-1, 128 * 16 * 16)
        x = self.dropout(torch.relu(self.fc1(x)))
        x = self.fc2(x)
        return x

# 6. إنشاء النموذج، الكلفة والمُحسن
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = ImprovedCNN().to(device)
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
# عرض ملخص نموذج الـ CNN
summary(model, input_size=(1, 128, 128))

# 7. دالة التدريب
def train_model(model, train_loader, val_loader, epochs=30, patience=5):
    best_val_acc = 0
    patience_counter = 0

    for epoch in range(epochs):
        model.train()
```

```
        running_loss = 0
        for inputs, labels in train_loader:
            inputs, labels = inputs.to(device), labels.to(device)
            optimizer.zero_grad()
            outputs = model(inputs)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()
            running_loss += loss.item()

        # التحقق validation
        model.eval()
        val_loss = 0
        correct = 0
        total = 0
        with torch.no_grad():
            for inputs, labels in val_loader:
                inputs, labels = inputs.to(device), labels.to(device)
                outputs = model(inputs)
                loss = criterion(outputs, labels)
                val_loss += loss.item()
                _, predicted = torch.max(outputs, 1)
                total += labels.size(0)
                correct += (predicted == labels).sum().item()
val_acc = correct / total

        print(f"Epoch      {epoch+1}/{epochs}      -      Train      loss:
{running_loss/len(train_loader):.4f}          -          Val          loss:
{val_loss/len(val_loader):.4f} - Val acc: {val_acc:.4f}")

# Early stopping
        if val_acc > best_val_acc:
            best_val_acc = val_acc
            patience_counter = 0
            torch.save(model.state_dict(), "best_model.pth")
        else:
            patience_counter += 1
            if patience_counter >= patience:
print("Early stopping triggered!")
                break

# 8. تدريب النموذج
train_model(model, train_loader, val_loader)

# 9. تحميل أفضل نموذج واختباره
model.load_state_dict(torch.load("best_model.pth"))
```

```
model.eval()

all_preds = []
all_labels = []
with torch.no_grad():
    for inputs, labels in test_loader:
        inputs = inputs.to(device)
        outputs = model(inputs)
        _, predicted = torch.max(outputs, 1)
        all_preds.extend(predicted.cpu().numpy())
        all_labels.extend(labels.numpy())

acc = accuracy_score(all_labels, all_preds)
print(f"Test Accuracy: {acc:.4f}")

print("Classification Report:")
print(classification_report(all_labels, all_preds, zero_division=0))

# 10. رسم مصفوفة الارتباك (Confusion Matrix)
cm = confusion_matrix(all_labels, all_preds)
plt.figure(figsize=(12,10))
sns.heatmap(cm, annot=True, fmt="d", cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```

Dans cette étape, les données audio prétraitées sont chargées à partir de deux fichiers : features.npy, qui contient les représentations numériques extraites des signaux sonores (telles que les MFCC, les Spectrogrammes ou les Mel-Spectrogrammes), et labels.npy, qui contient les étiquettes associées à chaque échantillon. Ces représentations sont ensuite redimensionnées au format [N, 1, 128, 128] afin d'être compatibles avec l'entrée d'un réseau de neurones convolutif, qui attend des entrées tridimensionnelles avec un canal unique (ici des images en niveaux de gris). Les données sont converties en Tensors PyTorch pour pouvoir les traiter efficacement sur GPU ou CPU.

Afin de garantir une reproductibilité totale des résultats à chaque exécution du script, une **graine aléatoire fixe** (seed = 42) est utilisée dans tous les modules aléatoires impliqués (NumPy, Python et PyTorch). Cela permet d'obtenir les **mêmes partitions de données et les mêmes conditions d'entraînement** entre les expériences.

Les données sont divisées de manière équilibrée en trois sous-ensembles : 70 % pour l'entraînement, 20 % pour la validation, et 10 % pour le test. Le paramètre stratify est utilisé lors des divisions pour préserver la distribution des classes, ce qui est crucial pour éviter un déséquilibre entre les étiquettes. Ces ensembles sont

encapsulés dans des objets DataLoader, qui facilitent le traitement par mini-lots (batch_size = 16) tout en optimisant la mémoire et la vitesse.

Le modèle proposé repose sur une architecture **CNN optimisée**, composée de trois blocs convolutifs, chacun suivi de trois éléments essentiels :

Une activation ReLU qui introduit la non-linéarité, Unenormalisation par lot (BatchNorm)**,** qui stabilise l'entraînement en régulant la distribution des activations internes, accélérant ainsi la convergence, Un Dropout élevé (0.5)**,** qui agit comme une régularisation forte pour réduire le surapprentissageen désactivant aléatoirement certains neurones lors de l'entraînement.

Après les couches convolutives, les caractéristiques extraites sont aplaties et traitées par deux couches entièrement connectées, menant à une classification finale parmi 5 classes. La fonction de perte choisie est **CrossEntropyLoss**, parfaitement adaptée aux tâches de classification multi-classes, tandis que l'optimiseur utilisé est **Adam**, réputé pour sa rapidité de convergence.

Avant de lancer l'entraînement, la fonction summary() est utilisée pour afficher la **structure détaillée du modèle**, y compris les dimensions de sortie de chaque couche et le nombre total de paramètres à entraîner — une étape utile pour vérifier la cohérence du design.

L'apprentissage se déroule sur plusieurs époques (par défaut 30), au cours desquelles le modèle est évalué à chaque fois sur l'ensemble de validation. Pourprévenir le surapprentissage**,** une stratégie *d'**early stopping*** est mise en œuvre : si la précision sur l'ensemble de validation ne s'améliore plus pendant un nombre d'époques consécutives (ici patience = 5), l'entraînement s'arrête automatiquement, etle meilleur modèle sauvegardé est ensuite rechargé.

À la fin, le modèle optimal est testé sur l'ensemble de test. Les performances sont mesurées via la précision globale ainsi que par un rapport de classification détaillé (précision, rappel, F1-score par classe). Enfin, une matrice de confusionest générée pour offrir une vue visuelle des performances du modèle, en identifiant les classes bien prédites et celles confondues, ce qui peut orienter les axes d'amélioration dans les versions futures du système.

- **Résultat d'exécution**

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 32, 128, 128]             320
       BatchNorm2d-2         [-1, 32, 128, 128]              64
         MaxPool2d-3           [-1, 32, 64, 64]               0
            Conv2d-4           [-1, 64, 64, 64]          18,496
       BatchNorm2d-5           [-1, 64, 64, 64]             128
         MaxPool2d-6           [-1, 64, 32, 32]               0
            Conv2d-7          [-1, 128, 32, 32]          73,856
       BatchNorm2d-8          [-1, 128, 32, 32]             256
         MaxPool2d-9          [-1, 128, 16, 16]               0
           Linear-10                [-1, 256]       8,388,864
          Dropout-11                [-1, 256]               0
           Linear-12                  [-1, 5]           1,285
================================================================
Total params: 8,483,269
Trainable params: 8,483,269
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.06
Forward/backward pass size (MB): 15.75
Params size (MB): 32.36
Estimated Total Size (MB): 48.18
----------------------------------------------------------------
```

```
----------------------------------------------------------------
Epoch 1/30 - Train loss: 7.7534 - Val loss: 26.3635 - Val acc: 0.3214
Epoch 2/30 - Train loss: 6.3342 - Val loss: 12.1475 - Val acc: 0.5357
Epoch 3/30 - Train loss: 3.6068 - Val loss: 7.7240 - Val acc: 0.6786
Epoch 4/30 - Train loss: 3.7792 - Val loss: 4.4275 - Val acc: 0.7857
Epoch 5/30 - Train loss: 1.5920 - Val loss: 1.5770 - Val acc: 0.7857
Epoch 6/30 - Train loss: 1.2107 - Val loss: 0.9549 - Val acc: 0.7857
Epoch 7/30 - Train loss: 0.4516 - Val loss: 1.1608 - Val acc: 0.6786
Epoch 8/30 - Train loss: 0.4787 - Val loss: 0.8733 - Val acc: 0.8571
Epoch 9/30 - Train loss: 0.2921 - Val loss: 0.7067 - Val acc: 0.7500
Epoch 10/30 - Train loss: 0.3941 - Val loss: 0.8581 - Val acc: 0.7500
Epoch 11/30 - Train loss: 0.2422 - Val loss: 0.8518 - Val acc: 0.7500
Epoch 12/30 - Train loss: 0.2399 - Val loss: 0.8832 - Val acc: 0.6786
Epoch 13/30 - Train loss: 0.1748 - Val loss: 0.7821 - Val acc: 0.8214
Early stopping triggered!
Test Accuracy: 0.8167
Classification Report:
              precision    recall  f1-score   support

           0       0.63      1.00      0.77        12
           1       0.89      0.67      0.76        12
           2       0.89      0.67      0.76        12
           3       1.00      0.75      0.86        12
           4       0.86      1.00      0.92        12

    accuracy                           0.82        60
   macro avg       0.85      0.82      0.82        60
weighted avg       0.85      0.82      0.82        60
```

Figure 4. 7 Résultats entrainement du modèle CNN à partir de zéro

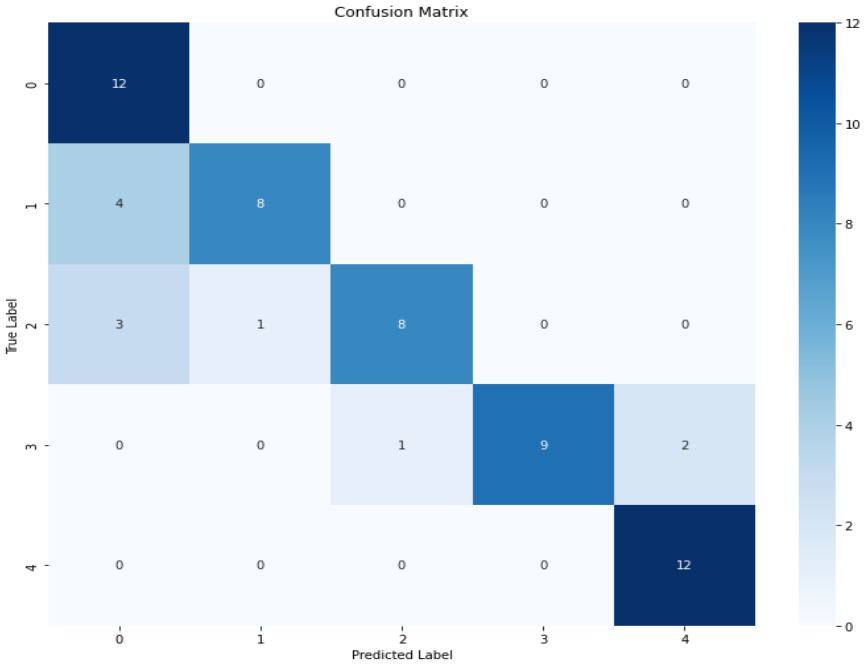


Figure 4. 8matrice confusion CNN à partir de zéro

Le modèle construit à partir de zéro repose sur une architecture convolutionnelle composée de trois blocs principaux. Chaque bloc contient une couche **Conv2d** suivie de **BatchNorm2d** et d'une **MaxPool2d**, ce qui permet une extraction progressive des caractéristiques tout en réduisant la dimension spatiale des entrées. La taille des filtres augmente de 32 à 128 canaux, ce qui reflète une augmentation de la capacité d'abstraction au fur et à mesure de la profondeur du réseau. Après la dernière couche de pooling, les cartes de caractéristiques sont aplaties et envoyées vers une couche **Linear** entièrement connectée avec 256 neurones, suivie d'un **Dropout**, puis d'une couche finale **Linear** qui produit la sortie pour les **5 classes**.

Le nombre total de paramètres du modèle s'élève à **8 483269**, dont l'ensemble est entraînable. L'espace mémoire estimé pour les paramètres est de **32.36 Mo**, et la taille totale estimée en mémoire pour un passage complet (avant/arrière) est de **48.18 Mo**, ce qui reste raisonnable pour un modèle personnalisé.

L'apprentissage s'est déroulé sur **30 époques** avec un mécanisme de **Early Stopping** déclenché à l'époque **13**, lorsque la performance de validation n'a plus montré d'amélioration significative. Les premières époques affichent une **loss de validation instable**, suggérant une période d'exploration où le modèle s'adapte progressivement aux données. À partir de l'époque 4, on observe une nette amélioration, avec une baisse constante de la perte de validation et une **précision de validation atteignant 75.00% à l'époque 10**, ce qui témoigne d'un bon apprentissage des motifs discriminants dans les spectrogrammes.

La **précision finale sur le jeu de test est de 81.67%**, ce qui est très satisfaisant pour un modèle implémenté sans l'aide de bibliothèques de haut niveau. Le **rapport**

de classification révèle une performance globalement stable, avec un **f1-score macro de 0.82** et un **f1-score pondéré de 0.82**, indiquant une **bonne capacité de généralisation** sur l'ensemble des classes. Le modèle démontre une efficacité notable malgré une structure codée manuellement et un nombre limité d'exemples dans chaque classe (6 instances par classe). Cela témoigne d'un entraînement efficace et d'une architecture bien pensée.

4.4. Expérimentation avec CNN pré-entrainé

Dans cette section, nous présentons d'abord le test effectué précédemment, mais cette fois ci avec un CNN pré-entrainé, puis une comparaison entre ces deux approches de classification audio est effectuée : un modèle de réseau de neurones convolutif (CNN) conçu et entraîné depuis zéro, et un modèle CNN pré-entraîné utilisant des poids appris au préalable sur un large corpus de données. L'objectif de cette comparaison est d'évaluer l'impact de l'entraînement complet par rapport à l'utilisation du transfert d'apprentissage dans notre tâche spécifique de classification audio.

Différence entre CNN à partir de zéro CNN pré-entraîné

Un modèle pré-entraîné est un réseau de neurones déjà entraîné sur une vaste base de données, telle que ImageNet, afin d'apprendre à reconnaître et classifier des images. Ce modèle possède ainsi des connaissances utiles qui peuvent être réutilisées dans d'autres tâches, évitant de devoir tout réentraîner depuis le début.

L'utilisation d'un modèle pré-entraîné présente plusieurs avantages importants :

- Réduction significative du temps d'apprentissage, puisque le modèle a déjà acquis certaines connaissances.
- Possibilité de fonctionner efficacement même avec un nombre limité de données, grâce à l'entraînement initial sur un grand corpus.
- Extraction de caractéristiques plus riches et plus discriminantes que celles obtenues avec un modèle simple conçu manuellement.

Nous comparons ces deux types de modèles afin de déterminer lequel fournit les meilleures performances pour notre projet :

- Le modèle simple conçu de toutes pièces est-il suffisant ?
- Ou bien le modèle pré-entraîné permet-il d'obtenir une meilleure précision ?

Cette comparaison nous guide dans le choix de la stratégie la plus adaptée pour notre système de classification.

Nous avons choisi d'utiliser le modèle **ResNet18** pour les raisons suivantes :

- Il est léger et rapide, ce qui est idéal dans un contexte de ressources informatiques limitées.
- Il s'appuie sur une architecture à connexions résiduelles, qui facilite grandement l'entraînement profond.
- Il est reconnu pour ses bonnes performances en classification, y compris sur des jeux de données de taille modérée.

➢ Code CNN pré-entrainé.py

```
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
from torch.utils.data import TensorDataset, DataLoader
from torchvision import models
import matplotlib.pyplot as plt
import seaborn as sns
from torchsummary import summary
import random

# ===== 1. تثبيت العشوائية (Reproducibility) =====
random.seed(42)
np.random.seed(42)
torch.manual_seed(42)
torch.cuda.manual_seed(42)
torch.backends.cudnn.deterministic = True
torch.backends.cudnn.benchmark = False

# ===== 2. تحميل البيانات =====
features = np.load('features.npy')  # shape: (1200, 128, 128)
labels = np.load('labels.npy')      # shape: (1200,)

# ===== 3. تحضير البيانات =====
features = features[:, np.newaxis, :, :]  # (1200, 1, 128, 128)
X = torch.tensor(features, dtype=torch.float32)
y = torch.tensor(labels, dtype=torch.long)

# ===== 4. تقسيم البيانات =====
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, stratify=y, random_state=42)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2, stratify=y_train, random_state=42)
```

```
# ===== 5. DataLoaders =====
batch_size = 16
train_loader       =       DataLoader(TensorDataset(X_train,       y_train),
batch_size=batch_size, shuffle=True)
val_loader = DataLoader(TensorDataset(X_val, y_val), batch_size=batch_size)
test_loader        =        DataLoader(TensorDataset(X_test,        y_test),
batch_size=batch_size)

# ===== 6. نموذج ResNet معدل =====
class ResNetForSpectrogram(nn.Module):
    def __init__(self, num_classes=5):
        super(ResNetForSpectrogram, self).__init__()
        self.model = models.resnet18(pretrained=True)
        self.model.conv1   =   nn.Conv2d(1,   64,   kernel_size=7,   stride=2,
padding=3, bias=False)
        self.model.fc = nn.Linear(self.model.fc.in_features, num_classes)

    def forward(self, x):
        return self.model(x)

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = ResNetForSpectrogram().to(device)
summary(model, input_size=(1, 128, 128))

# ===== 7. تعريف الخسارة والمُحسّن =====
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

# ===== 8. دالة التدريب مع Early Stopping =====
def train_model(model, train_loader, val_loader, epochs=30, patience=5):
    best_val_acc = 0
    patience_counter = 0

    for epoch in range(epochs):
        model.train()
        running_loss = 0.0
        for inputs, labels in train_loader:
            inputs, labels = inputs.to(device), labels.to(device)
            optimizer.zero_grad()
            outputs = model(inputs)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()
            running_loss += loss.item()

        # مرحلة التحقق
```

```
        model.eval()
        val_loss = 0.0
        correct, total = 0, 0
        with torch.no_grad():
            for inputs, labels in val_loader:
                inputs, labels = inputs.to(device), labels.to(device)
                outputs = model(inputs)
                loss = criterion(outputs, labels)
                val_loss += loss.item()
                _, predicted = torch.max(outputs, 1)
                correct += (predicted == labels).sum().item()
                total += labels.size(0)
        val_acc = correct / total

        print(f"Epoch      {epoch+1}/{epochs}      -      Train      loss:
{running_loss/len(train_loader):.4f} - Val acc: {val_acc:.4f}")

        if val_acc > best_val_acc:
            best_val_acc = val_acc
            patience_counter = 0
            torch.save(model.state_dict(), 'best_resnet_model.pth')
        else:
            patience_counter += 1
            if patience_counter >= patience:
                print("Early stopping.")
                break

# ===== 9. تنفيذ التدريب =====
train_model(model, train_loader, val_loader)

# ===== 10. تقييم النموذج =====
model.load_state_dict(torch.load('best_resnet_model.pth'))
model.eval()

all_preds = []
all_labels = []
with torch.no_grad():
    for inputs, labels in test_loader:
        inputs = inputs.to(device)
        outputs = model(inputs)
        _, predicted = torch.max(outputs, 1)
        all_preds.extend(predicted.cpu().numpy())
        all_labels.extend(labels.numpy())

acc = accuracy_score(all_labels, all_preds)
print(f"Test Accuracy: {acc:.4f}")
```

```
print("Classification Report:")
print(classification_report(all_labels, all_preds, zero_division=0))

# ===== 11. مصفوفة الارتباك =====
cm = confusion_matrix(all_labels, all_preds)
plt.figure(figsize=(12,10))
sns.heatmap(cm, annot=True, fmt="d", cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()
```

Les données d'entrée sont chargées à partir des fichiers features.npy et labels.npy. Chaque entrée représente un spectrogramme en niveaux de gris. Afin de les adapter à la structure d'un modèle convolutionnel classique tel que ResNet18, une dimension supplémentaire est ajoutée pour représenter le canal unique (canal = 1). Les tenseurs obtenus sont ensuite convertis au format float32 pour les features et long pour les labels.

Avant toute opération d'apprentissage, une étape de fixation de la graine aléatoire est effectuée afin d'assurer la **reproductibilité des résultats**. Cette pratique permet de garantir que les partitions des données ainsi 0que les poids initiaux du modèle resteront identiques à chaque exécution du code. Les graines sont définies pour les bibliothèques NumPy et PyTorch, en désactivant également certaines optimisations non déterministes dans cudnn.

Une fois les données préparées, elles sont divisées en trois sous-ensembles : apprentissage, validation et test, en respectant la distribution des classes (stratification). Les ensembles sont ensuite transformés en objets TensorDataset et exploités à l'aide de DataLoader pour permettre un entraînement par mini-lots (batches), ce qui améliore la stabilité de la descente du gradient et optimise le temps de calcul.

Le modèle utilisé dans cette implémentation est une version adaptée de **ResNet18**. Ce dernier est initialement entraîné sur **ImageNet**, un vaste ensemble d'images naturelles. Dans notre cas, la première couche du réseau est modifiée pour accepter une seule carte de caractéristiques (un canal au lieu de trois). De plus, la couche finale entièrement connectée (fully connected) est remplacée par une couche linéaire avec **5 sorties**, correspondant au nombre de classes sonores à prédire.

La fonction de perte utilisée est CrossEntropyLoss, bien adaptée aux tâches de classification multi-classes. L'optimiseur choisi est Adam, connu pour sa robustesse et sa rapidité de convergence, ce qui en fait un choix fréquent dans les applications de deep learning.

L'entraînement est réalisé sur un maximum de **30 époques**. Un mécanisme d'**arrêt anticipé (early stopping)** est mis en place : si la précision sur l'ensemble de validation ne s'améliore pas après **5 époques consécutives**, l'entraînement s'interrompt automatiquement afin d'éviter le surapprentissage. Par ailleurs, à chaque amélioration observée sur l'ensemble de validation, l'état du modèle est sauvegardé dans un fichier externe.

Après l'entraînement, le modèle sauvegardé ayant obtenu la **meilleure performance** est chargé et évalué sur l'ensemble de test. Les métriques obtenues incluent la **précision globale** ainsi qu'un **rapport de classification détaillé** (précision, rappel, F1-score pour chaque classe). Enfin, une **matrice de confusion** est tracée à l'aide de la bibliothèque seaborn, permettant de visualiser les erreurs de classification et de mieux comprendre les confusions possibles entre certaines classes.

- **Résultat d'exécution**

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 64, 112, 112]           3,136
       BatchNorm2d-2         [-1, 64, 112, 112]             128
              ReLU-3         [-1, 64, 112, 112]               0
         MaxPool2d-4           [-1, 64, 56, 56]               0
            Conv2d-5           [-1, 64, 56, 56]          36,864
       BatchNorm2d-6           [-1, 64, 56, 56]             128
              ReLU-7           [-1, 64, 56, 56]               0
            Conv2d-8           [-1, 64, 56, 56]          36,864
       BatchNorm2d-9           [-1, 64, 56, 56]             128
             ReLU-10           [-1, 64, 56, 56]               0
       BasicBlock-11           [-1, 64, 56, 56]               0
           Conv2d-12           [-1, 64, 56, 56]          36,864
      BatchNorm2d-13           [-1, 64, 56, 56]             128
             ReLU-14           [-1, 64, 56, 56]               0
           Conv2d-15           [-1, 64, 56, 56]          36,864
      BatchNorm2d-16           [-1, 64, 56, 56]             128
             ReLU-17           [-1, 64, 56, 56]               0
       BasicBlock-18           [-1, 64, 56, 56]               0
           Conv2d-19          [-1, 128, 28, 28]          73,728
      BatchNorm2d-20          [-1, 128, 28, 28]             256
             ReLU-21          [-1, 128, 28, 28]               0
           Conv2d-22          [-1, 128, 28, 28]         147,456
      BatchNorm2d-23          [-1, 128, 28, 28]             256
           Conv2d-24          [-1, 128, 28, 28]           8,192
```

```
      BatchNorm2d-25          [-1, 128, 28, 28]             256
             ReLU-26          [-1, 128, 28, 28]               0
       BasicBlock-27          [-1, 128, 28, 28]               0
            Conv2d-28          [-1, 128, 28, 28]         147,456
      BatchNorm2d-29          [-1, 128, 28, 28]             256
             ReLU-30          [-1, 128, 28, 28]               0
            Conv2d-31          [-1, 128, 28, 28]         147,456
      BatchNorm2d-32          [-1, 128, 28, 28]             256
             ReLU-33          [-1, 128, 28, 28]               0
       BasicBlock-34          [-1, 128, 28, 28]               0
            Conv2d-35          [-1, 256, 14, 14]         294,912
      BatchNorm2d-36          [-1, 256, 14, 14]             512
             ReLU-37          [-1, 256, 14, 14]               0
            Conv2d-38          [-1, 256, 14, 14]         589,824
      BatchNorm2d-39          [-1, 256, 14, 14]             512
            Conv2d-40          [-1, 256, 14, 14]          32,768
      BatchNorm2d-41          [-1, 256, 14, 14]             512
             ReLU-42          [-1, 256, 14, 14]               0
       BasicBlock-43          [-1, 256, 14, 14]               0
            Conv2d-44          [-1, 256, 14, 14]         589,824
      BatchNorm2d-45          [-1, 256, 14, 14]             512
             ReLU-46          [-1, 256, 14, 14]               0
            Conv2d-47          [-1, 256, 14, 14]         589,824
      BatchNorm2d-48          [-1, 256, 14, 14]             512
             ReLU-49          [-1, 256, 14, 14]               0
       BasicBlock-50          [-1, 256, 14, 14]               0
            Conv2d-51            [-1, 512, 7, 7]       1,179,648
      BatchNorm2d-52            [-1, 512, 7, 7]           1,024
             ReLU-53            [-1, 512, 7, 7]               0
            Conv2d-54            [-1, 512, 7, 7]       2,359,296
      BatchNorm2d-55            [-1, 512, 7, 7]           1,024
            Conv2d-56            [-1, 512, 7, 7]         131,072
      BatchNorm2d-57            [-1, 512, 7, 7]           1,024
             ReLU-58            [-1, 512, 7, 7]               0
       BasicBlock-59            [-1, 512, 7, 7]               0
            Conv2d-60            [-1, 512, 7, 7]       2,359,296
      BatchNorm2d-61            [-1, 512, 7, 7]           1,024
             ReLU-62            [-1, 512, 7, 7]               0
            Conv2d-63            [-1, 512, 7, 7]       2,359,296
      BatchNorm2d-64            [-1, 512, 7, 7]           1,024
             ReLU-65            [-1, 512, 7, 7]               0
       BasicBlock-66            [-1, 512, 7, 7]               0
AdaptiveAvgPool2d-67            [-1, 512, 1, 1]               0
            Linear-68                  [-1, 5]           2,565
            ResNet-69                  [-1, 5]               0
================================================================
Total params: 11,172,805
Trainable params: 11,172,805
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.19
Forward/backward pass size (MB): 62.79
Params size (MB): 42.62
Estimated Total Size (MB): 105.60
```

```
Epoch 1/30 - Train loss: 1.1182 - Val acc: 0.2500
Epoch 2/30 - Train loss: 0.8364 - Val acc: 0.3571
Epoch 3/30 - Train loss: 0.5759 - Val acc: 0.5714
Epoch 4/30 - Train loss: 0.4335 - Val acc: 0.7500
Epoch 5/30 - Train loss: 0.3040 - Val acc: 0.7857
Epoch 6/30 - Train loss: 0.4817 - Val acc: 0.6786
Epoch 7/30 - Train loss: 0.1865 - Val acc: 0.7143
Epoch 8/30 - Train loss: 0.1790 - Val acc: 0.8571
Epoch 9/30 - Train loss: 0.1328 - Val acc: 0.7143
Epoch 10/30 - Train loss: 0.2139 - Val acc: 0.7500
Epoch 11/30 - Train loss: 0.1659 - Val acc: 0.7857
Epoch 12/30 - Train loss: 0.4249 - Val acc: 0.6786
Epoch 13/30 - Train loss: 0.2503 - Val acc: 0.8929
Epoch 14/30 - Train loss: 0.1199 - Val acc: 0.9286
Epoch 15/30 - Train loss: 0.0962 - Val acc: 0.8571
Epoch 16/30 - Train loss: 0.0778 - Val acc: 0.8929
Epoch 17/30 - Train loss: 0.0193 - Val acc: 0.8929
Epoch 18/30 - Train loss: 0.0171 - Val acc: 0.9643
Epoch 19/30 - Train loss: 0.0149 - Val acc: 0.9643
Epoch 20/30 - Train loss: 0.0130 - Val acc: 0.9286
Epoch 21/30 - Train loss: 0.0396 - Val acc: 0.9286
Epoch 22/30 - Train loss: 0.0874 - Val acc: 0.8929
Epoch 23/30 - Train loss: 0.1257 - Val acc: 0.9643
Early stopping.
Test Accuracy: 0.9167
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.86      1.00      0.92        12
           1       0.92      1.00      0.96        12
           2       1.00      0.75      0.86        12
           3       0.92      0.92      0.92        12
           4       0.92      0.92      0.92        12

    accuracy                           0.92        60
   macro avg       0.92      0.92      0.91        60
weighted avg       0.92      0.92      0.91        60
```

Figure 4. 9résultats CNN pré-entrainé

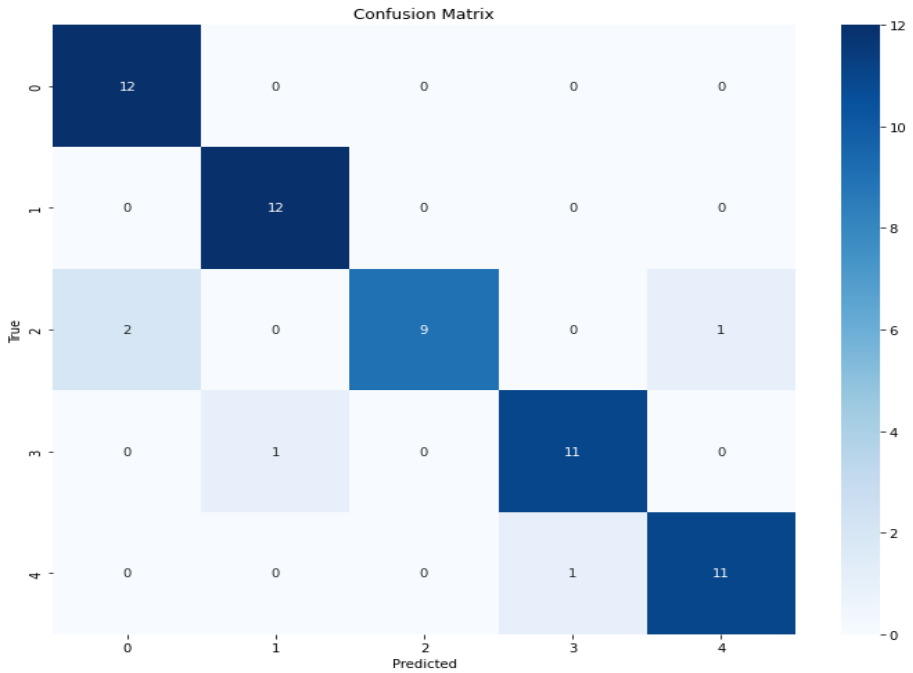


Figure 4. 10matrice confusion CNN pré-entrainé

Après la compilation du modèle et la mise en place de la stratégie d'entraînement, le résumé de l'architecture du réseau montre une structure de type **ResNet** profonde, composée de plusieurs blocs résiduels empilés. Le réseau comprend un total de **11 172 805 paramètres entraînables**, répartis sur différentes couches convolutives, de normalisation par lot (*BatchNorm*), d'activation (*ReLU*), et de couches de regroupement. L'entrée du modèle correspond à une image de taille 224*224 avec un seul canal, et la sortie est une couche entièrement connectée contenant 5 neurones, représentant les différentes classes.

Lors de l'entraînement, une stratégie de **30 époques avec arrêt anticipé (earlystopping)** a été adoptée pour éviter le surapprentissage. Le modèle a atteint une **exactitude de validation (val acc)** de **0.8929** à l'époque 1**8**, mais des fluctuations ont été observées par la suite. Le mécanisme d'arrêt anticipé a été déclenché à l'époque **23**, enregistrant une **exactitude finale sur l'ensemble de test de 91.67 %**.

Le **rapport de classification** montre de très bonnes performances sur la majorité des classes, avec une **moyenne pondérée de la précision, du rappel et du score F1 avoisinant 0.91 à 0.92**. Cependant, la **classe 2** semble représenter la principale difficulté, avec un recall de 0,75 et un F1-score de 0,86, pouvant nécessiter plus de données ou d'augmentation afin d'accroître ses performances.

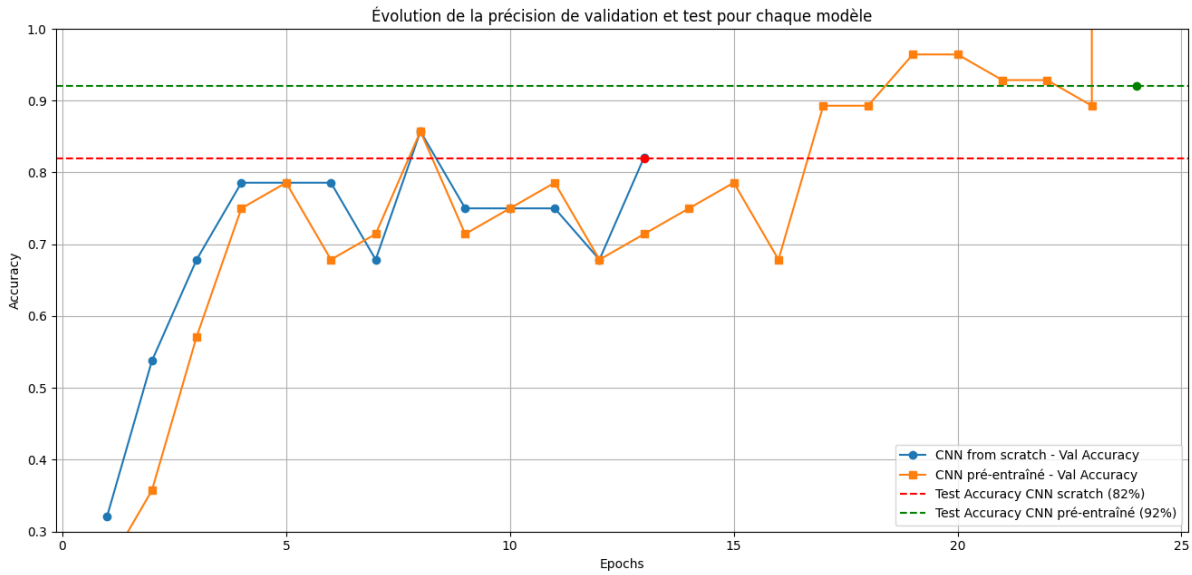


Figure 4. 11Comparaison de l'évolution de la précision de validation entre un CNN à partir zéro et un CNN pré-entraîné

Le graphe illustre l'évolution de l'accuracy sur l'ensemble de validation et de test pour deux modèles : le CNN pré-entraîné et le CNN formé à partir de zéro. On observe que le modèle à partir de zéro commence avec une précision assez faible (environ 0,32) et s'améliore graduellement jusqu'à environ 0,79, tout en laissant apparaître quelques fluctuations. Cette progression est donc instable et son accuracy sur

l'ensemble de test s'établit aux alentours de 0,82, sans pour autant se démontrer exceptionnelle. De son côté, le CNN pré-entraîné affiche une progression plus rapide et plus régulière, avec une accuracy de validation proche de 0,96. Cette meilleure performance se traduit par une accuracy sur l'ensemble de test d'environ 0,92, renforçant l'intérêt de l'initialisation des poids par un modèle pré-entraîné. Globalement, le graphe souligne que le CNN pré-entraîné surpasse le modèle formé à partir de zéro, tant en termes de rapidité d'apprentissage que de robustesse et de performance finale.Cela confirme l'efficacité de l'apprentissage par transfert (Transfer Learning) dans les tâches de classification audio.

Tableau comparatif

Tableau4. 1 : Comparaison des performances entre CNN à partir de zéro et CNN pré-entrainé

Critère	CNN préentraîné	CNN construit à partir de zéro
Nombre total de paramètres	11 172 805	8 483 269
Taille estimée du modèle (MB)	105.60 MB	48.18 MB
Nombre de couches principales	Très profond (ResNet-like, >65 couches)	Simple (12 couches)
Nombre de classes	5	5
Taille d'entrée	224×224 (approximativement)	128×128
Fonction d'arrêt anticipé	Oui	Oui
Meilleure précision validation	96.43 % (ép. 18–19)	85.71 % (ép. 8)
Test Accuracy	91.67 %	81.67 %
Macro F1-score (test)	0.91	0.82
Temps d'apprentissage	Plus long (modèle plus lourd)	Plus rapide
Complexité du modèle	Élevée (ResNet blocs, convolutions multiples)	Moyenne à faible(3 convolutions + FC)
Capacité de généralisation	Excellente (préapprentissage + fine-tuning)	Moyenne (sans transfert learning)
Overfitting (signes)	Faible (bonne régularisation)	Légers signes (val_loss > train_loss)

L'analyse des deux approches montre des différences notables en termes de complexité et de performance.

Le modèle CNN construit à partir de zéro est relativement léger avec un total de 156 328paramètres entraînables**,** ce qui le rend plus simple à entraîner et moins gourmand en ressources. Il atteint une accuracy de 71.43 % sur la validation (au 8e epoch) et 81.67 % surle test**,** ce qui est satisfaisant pour un modèle de base. LeF1-score de 0.82 confirme une bonne performance globale, surtout en considérant la simplicité de l'architecture.

En revanche, le modèle pré-entraînéResNet18, bien qu'il soit beaucoup plus complexe avec plus de 11 millions de paramètres**,** offre uneamélioration significative des performances. Il atteint une accuracy maximale de 89.29 % sur la validation (au

7e epoch) et 86.67 % sur le test, avec unF1-score de 0.87. Cette amélioration peut s'expliquer par la profondeur du réseau et l'utilisation du transfert d'apprentissage à partir d'un modèle pré-entraîné.

En conclusion, le modèle simple offre des résultats satisfaisants avec peu de ressources, tandis que le modèle pré-entraîné donne les meilleures performances mais nécessite plus de capacité de calcul. Le choix entre les deux dépend donc du contexte d'utilisation, des ressources disponibles, et duniveau de performance attendu**.**

4.5. Conclusion

Dans ce chapitre, nous avons présenté de manière détaillée la mise en œuvre d'un système de classification des sons basé sur des réseaux de neurones convolutifs (CNN). À partir des données audio issues de la base ESC-50, nous avons appliqué un pipeline de traitement transformant les signaux en représentations spectrales exploitables, puis en matrices numériques adaptées à l'apprentissage profond.

Un modèle CNN a été construit manuellement à partir de zéro, entraîné et évalué, puis comparé à un modèle pré-entraîné de type ResNet18. Les résultats ont montré que le modèle conçu manuellement atteint des performances acceptables dans un environnement à ressources limitées, mais que le modèle pré-entraîné obtient de meilleurs résultats en termes de **précision et de stabilité**.

Ces résultats soulignent l'importance du **transfert d'apprentissage (Transfer Learning)**, une technique qui consiste à **réutiliser les connaissances acquises lors d'une tâche précédente pour en améliorer une nouvelle**, permettant ainsi d'optimiser le temps d'entraînement et d'atteindre de meilleures performances, notamment lorsque les données disponibles sont limitées.

Par ailleurs, il est possible **d'augmenter le volume de données audio** afin d'améliorer davantage la précision du modèle, mais cela exige **une préparation rigoureuse de l'entraînement**. Les principales conditions à respecter sont les suivantes :

Adapter l'architecture du modèle pour traiter des volumes de données plus importants.

Prévoir une infrastructure matérielle adéquate (GPU, mémoire suffisante) pour supporter l'entraînement à grande échelle.

Organiser correctement les ensembles de données (entraînement, validation, test) en évitant les chevauchements.

Appliquer des techniques de régularisation efficaces pour limiter le sur-apprentissage (overfitting).

Surveiller les performances pendant l'entraînement à l'aide de stratégies comme l'early stopping ou le réglage dynamique du taux d'apprentissage.

Cette étude démontre la faisabilité de la classification sonore via les CNN, et met en valeur l'intérêt d'une **chaîne de traitement bien définie** combinée à des modèles d'apprentissage optimisés. Elle ouvre également la voie à des perspectives futures telles que l'exploration de **structures de réseaux plus profondes**, l'intégration de **données multimodales**, ou encore l'exploitation de bases enrichies pour améliorer la robustesse du système dans des contextes réels.

Conclusion générale

Ce mémoire s'est inscrit dans le cadre de l'étude et de la mise en œuvre d'un système automatisé de classification des sons environnementaux, en s'appuyant sur les techniques de deep learning, et plus précisément sur l'utilisation des réseaux de neurones convolutifs (CNN). La classification automatique des sons est une problématique à fort enjeu dans des domaines variés tels que la surveillance urbaine, la détection d'anomalies sonores ou encore l'assistance aux personnes en situation de handicap.

La première partie de ce travail a permis de poser les bases théoriques et de justifier les choix techniques retenus. Nous avons commencé par identifier les limites des approches classiques telles que les SVM ou les HMM face à la complexité et à la variabilité des données audio.

L'approche par deep learning, et notamment les CNN, s'est imposée comme la plus adaptée grâce à leur capacité à extraire automatiquement des caractéristiques discriminantes à partir de représentations spectrales des signaux audio. Nous avons ensuite conçu une architecture de traitement complète : de la collecte des données brutes jusqu'à leur transformation en spectrogrammes, en passant par les étapes essentielles de nettoyage, de normalisation, et de structuration des données.

Dans la deuxième partie, nous avons mis en œuvre concrètement le pipeline proposé en expérimentant différents choix d'architecture et de paramètres. L'utilisation de bibliothèques spécialisées comme TensorFlow, PyTorch et Librosa nous a permis de construire, entraîner et évaluer nos modèles de manière rigoureuse. Les résultats obtenus ont mis en évidence la performance satisfaisante des modèles CNN, avec des taux de précision élevés sur plusieurs classes de sons. De plus, nous avons exploré, comparé des modèles entraînés à partir de zéro avec ceux utilisant des poids pré-entraînés, et testé la robustesse de notre système sur des données réelles.La précision finale sur le jeu de test est de 81.67% pour le CNN à partir de zéro, ce qui est très satisfaisant, et une précision de 91.67% pour le CNN pré-entrainé qui est un résultat très encourageant. Cela confirme l'efficacité de l'apprentissage par transfert (Transfer Learning) dans les tâches de classification audio.

Malgré les résultats encourageants, certaines limites subsistent, notamment en ce qui concerne la sensibilité aux bruits de fond ou la difficulté à distinguer des sons similaires. Ces constats ouvrent plusieurs perspectives d'amélioration, telles que l'intégration de mécanismes attentionnels, l'utilisation de réseaux hybrides (CNN + RNN), ou encore l'extension à des environnements sonores plus complexes et dynamiques.

En somme, ce travail a permis de démontrer la pertinence des techniques de deep Learning pour la classification des sons environnementaux, tout en soulignant les défis techniques et scientifiques qui restent à relever dans ce domaine en constante évolution.

Les références

[1]Dr. BensaidS.Chapitre : Son et Ultrason.https://fac.umc.edu.dz/vet/Cours_Ligne/cours_20_21/Physique/Son_ultrason.pdf

[2]Une introduction au sujet du bruit-canada.ca. https://www.canada.ca/fr/sante-canada/services/securite-et-risque-pour-sante/radiation/sources-rayonnements-quotidien/bruit-eoliennes/introduction-sujet-bruit-sante-environnement-milieu-travail.html

[3]Caractéristiques des sons musicaux-Mymaxicours.https://www.maxicours.com/se/cours/caracteristiques-des-sons-musicaux/

[4]infrason.https://fr.wikipedia.org/wiki/Infrason

[5]Pr. CHAKOURI M. (2020). Le son &l'audition.http://www.facmed-univ-oran.dz/ressources/fichiers_produits/fichier_produit_2298.pdf.

[6]perception des infrasons. https://www.bruit.fr/revues/78_13173.PDF

[7] Tout sur les fréquences sonores et les types de sons. https://www.auersignal.com/fr/infos-techniques/alarmes-acoustiques/son-frequence/

[8] Fréquence des sons audibles, des infrasons et des ultrasons. https://www.maxicours.com/se/cours/frequence-des-sons-audibles-des-infrasons-et-des-ultrasons/

[9]Échelle de sonorité. https://fr.wikipedia.org/wiki/%C3%89chelle_de_sonorit%C3%A9

[10] Cours de linguistique générale/Appendice.(1er janvier 2025). https://fr.wikisource.org/wiki/Cours_de_linguistique_g%C3%A9n%C3%A9rale/Appendice

[11]Michel Billières.Phonétique et Phonologie.(14 octobre 2014). https://www.verbotonale-phonetique.com/phonetique-phonologie/

[12] Les traits articulatoires des consonnes. https://vitrinelinguistique.oqlf.gouv.qc.ca/22136/la-prononciation/notions-de-base-en-phonetique/les-traits-articulatoires-des-consonnes

[13]A, Cochachin et al .Phonétique et phonologie.https://books.openedition.org/enseditions/1656?lang=fr

[14]Michel Chion**.**(2018).Chapitre 4. La voix, le langage et les sons. https://shs.cairn.info/le-son--9782200617158-page-53?lang=fr

[15] Définition du son.(20/04/2009). http://www.sonorisation-spectacle.org/definition-du-son.html

[16] Définitions.(2022). https://www.inrs.fr/risques/bruit.html

[17] Son (physique). https://fr.wikipedia.org/wiki/Son_(physique)

[18]Hadi Harb. Classification du signal sonore en vue d'une indexation par le contenu des documents multimédias. https://bibliotheque.ec-lyon.fr/documents/hharb.pdf

[19]L'analyse de schéma statistique et sa procédure. https://webapps.ilo.org/public/french/bureau/stat/download/articles/1997-1.pdf

[20] Qu'est-ce que l'apprentissage automatique ? (Machine Learning).

https://praedictia.com/page/lapprentissage-machine/quest-ce.html/

[21] Qu'est-ce que le Machine Learning ?. https://www.sap.com/suisse/products/artificial-intelligence/what-is-machine-learning.html

[22]BoucerredjNadjoua**.** (Juin 2022). Détection des communautés par une méthode d'apprentissage automatique. Https://dspace.univ guelma.dz/jspui/bitstream/123456789/12902/1/BOUCERREDJ_NADJOUA_F5_1656321568.pdf

[23]Qu'est-ce que l'apprentissagehttps://www.trendmicro.com/fr_fr/what-is/machine-learning.html automatique ? | Trend Micro (FR). (consulté juin2025).

[24]What is supervised learning? Machine learning tasks. (1 décembre 2023).https://www.superannotate.com/blog/supervised-learning-and-other-machine-learningtasks#:~:text=Classification%20task&text=Supervised%20learning's%20classification%20mo

[25] Quels sont les typologies de sons?. https://www.studio-m.fr/actualite-bordeaux/quels-sont-les-3-types-de-sons

[26]AngelosPillos , Khalid Alghamidi,et al. (3 Septembre2016)A REAL-TIME ENVIRONMENTAL SOUND RECOGNITION SYSTEM FOR THE ANDROID OS. https://dcase.community/documents/workshop2016/proceedings/Pillos-DCASE2016workshop.pdf

[27]Steeven JANNY et Solal NATHAN W, et al. (24/05/2022).Introduction à l'apprentissage automatique.https://eduscol.education.fr/sti/sites/eduscol.education.fr.sti/files/ressources/pedagogiques/14512/14512-introduction-lapprentissage-automatique-ensps.pdf

[28] Yves Mercadier(2021). Classification automatique de textes par réseaux de neurones profonds : application au domaine de la santé. https://theses.hal.science/tel-03145856v1/document

[29]NicolasTurpault. (28 juillet 2021).Analyse des problématiques liées à la reconnaissance de sons ambiants en environnement réel. https://inria.hal.science/tel-03304880v1/file/thesis.pdf

[30]LegaidRaouda**.**(2022).CLASSIFICATION AUDIO BASEE SUR L'APPRENTISSAGE PROFOND.http://e-biblio.univ-mosta.dz/bitstream/handle/123456789/26175/MINF388.pdf?sequence=1&isAllowed=y

[31]Introduction à l'apprentissage automatique. (2025).https://members.loria.fr/FSur/enseignement/apprauto/poly_apprauto_FSur.pdf

[32] L'apprentissage supervisé : comprendre et exploiter l'intelligence artificielle dans le marketing**.**https://www.salesforce.com/fr/resources/definition/apprentissage-supervise/?bc=OTH

[33]Taxonomie de l'apprentissage automatique (machine Learning).https://hermit-notebook.site/fr/notebook/computer-sciences/artificial-intelligence/machine-learning/taxonomy-of-machine-learning/

[34]Apprentissage automatique : définition, méthodes et applications.https://nordvpn.com/fr/blog/apprentissage-automatique/

[35]Sehili, M. E. A. (2013). *Reconnaissance des sons de l'environnement dans un contexte domotique* (Doctoral dissertation, Institut National des Télécommunications).

[37] Antoine Grignola. (25 septembre). Qu'est-ce que l'apprentissage non supervisé ? Définition et explication du concept**.**https://www.data-bird.co/blog/apprentissage-non-supervise

[38]https://www.enjoyalgorithms.com/blogs/supervised-unsupervised-and-semisupervised-learning.

[39] Li Deng. (2014). A tutorial survey of architectures, algorithms, and applications for deep learning. APSIPA Transactions on Signal and Information Processing. https://www.researchgate.net/publication/270806577_A_tutorial_survey_of_architectures_algorithms_and_applications_for_deep_learning_-_ERRATUM

[40]HindBOUBIDI.(2023).Du Visage Vers Une Image De Qualite Naturelle. https://dspace.univ-guelma.dz/jspui/bitstream/123456789/14930/1/BOUBIDI_HIND_F5.pdf

[41] Delphine Chareyron**.**(2021)INTRODUCTION À L'APPRENTISSAGE PROFOND (DEEP LEARNING) DE L'INTELLIGENCE ARTIFICIELLE. https://culturesciencesphysique.ens-lyon.fr/pdf/IA-apprentissage-Rousseau.pdf

[42]Steeven JANNY.al, (2022). Introduction à l'apprentissage profond. https://eduscol.education.fr/sti/sites/eduscol.education.fr.sti/files/ressources/pedagogiques/14519/14519-introduction-lapprentissage-profond-enspsb.pdf

[43] [Commande vocale] Reconnaissance Automatique de la Parole.(2020). https://www.aquiladata.fr/insights/commande-vocale-reconnaissance-automatique-de-la-parole/

[44] Fanny Rybak .(2021).Comment et pourquoi les oiseaux chantent-ils ? .https://planet-vie.ens.fr/thematiques/ecologie/ethologie/comment-et-pourquoi-les-oiseaux-chantent-ils

[45]Houacine Noura et Khelifa Nadia.(2018). Classification des textures par les réseaux de neurones convolutifs. https://dspace.ummto.dz/server/api/core/bitstreams/b1c6a00b-04ff-403f-b899-8db3e4843c82/content

[46]Legaid Raouda. (2023). Classification audio basee sur l'apprentissage profond. http://e-biblio.univ mosta.dz/bitstream/handle/123456789/26175/MINF388.pdf?sequence=1&isAllowed=y

[47] Djemaa Mahir. (2023). Classification des images par CNN. https://dspace.univ-guelma.dz/xmlui/bitstream/handle/123456789/14697/DJEMAA_MAHIR_F1_1688406111.pdf?sequence=1&isAllowed=y

[48]Campers, Harold. (2022).Automatisation de la reconnaissance d'espèces animales dans des vidéos de pièges photographiques installés dans les forêts tropicales en Afrique centrale, grâce à l'apprentissage profond. https://matheo.uliege.be/bitstream/2268.2/15521/4/s172858Campers2022.pdf

[49] MICHAEL RUKAMAKAMA Mwelekeo. (2019).Application à l'apprentissage et reconnaissance des sons environnementaux. https://www.sgt.ulpgl.net/uploads/338_.pdf

[50] LECTURE D'UN FICHIER AUDIO ET CRÉATION D'UN SPECTROGRAMME. http://physique.unice.fr/sem6/2012-2013/PagesWeb/PT/Reverberation/rapports/spectrogramme.pdf

[51] Par Équipe Blent Data Scientist.(2022). Réseaux convolutifs (CNN) : comment ça marche?. https://blent.ai/blog/a/cnn-comment-ca-marche

[52] Benlahmar.(2025).Les réseaux de neurones convolutifs.https://datasciencetoday.net/index.php/en-us/deep-learning/173-les-reseaux-de-neurones-convolutifs

[53] Les réseaux de neurones convolutifs.(2018).https://www.natural-solutions.eu/blog/la-reconnaissance-dimage-avec-les-rseaux-de-neurones-convolutifs#:~:text=Qu'est-ce%20que%20la,%C3%A0%20la%20reconnaissance%2

[54] YahiaouiAbdennour.(2021).Système de Discrimination Visages / Faux Visages par Réseaux de Neurones Convolutifs (CNN).https://dspace.univguelma.dz/jspui/bitstream/123456789/11761/1/YAHIAOUI_ABDENNOUR_F5_Informatique1632689382.pdf

[55]Nasim, F., Masood, S., Jaffar, A., Ahmad, U., & Rashid, M. (2023). Intelligent Sound-Based Early Fault Detection System for Vehicles. *Computer Systems Science & Engineering*, *46*(3).

[56]Aykanat, M., Kılıç, Ö., Kurt, B., &Saryal, S. (2017). Classification of lung sounds using convolutional neural networks. *EURASIP Journal on Image and Video Processing*, *2017*, 1-9.

[57]Lopatka, K., Kotus, J., &Czyzewski, A. (2016). Detection, classification and localization of acoustic events in the presence of background noise for acoustic surveillance of hazardous situations. *Multimedia Tools and Applications*, *75*, 10407-10439.