Université de BLIDA 1

Faculté des Sciences

Département d'Informatique

MASTER THESIS

Option : Ingénierie des Systèmes Informatiques et Réseaux

ENCODER-DECODER-BASED NEURAL NETWORK ARCHITECTURES FOR AUTOMATIC AUDIO CAPTIONING

By :

Gharouba Hadil

Ben doumia kaouther

In front of a jury composed of:

Dr. Hadjer YKHLEF	Supervisor
Dr Kamech Abdallah Hicham	President
Dr Chettat Merouane	Examiner

2024-2025

Université de BLIDA 1

Faculté des Sciences

Département d'Informatique

Mémoire de Master

Option : Ingénierie des Systèmes Informatiques et Réseaux

ARCHITECTURES DE RÉSEAUX DE NEURONES BASÉES SUR LE MODÈLE ENCODEUR-DÉCODEUR POUR L'AUTOMATIC AUDIO CAPTIONING

Présenté par :

Gharouba Hadil

Ben doumia kaouther

Devant le jury composé de :

Dr. Hadjer YKHLEF	Promoteur
Dr Kamech Abdallah Hicham	Président
Dr Chettat Merouane	Examinateur

2024-2025

RÉSUMÉ

L’objectif principal de notre projet est de développer un système efficace d’Automated Audio Captioning (AAC), une tâche qui consiste à décrire les sons ambiants présents dans un extrait audio à l’aide d’une phrase en langage naturel, comblant ainsi le fossé entre la perception auditive et l’expression linguistique. Ces dernières années, l’AAC a suscité un intérêt considérable et a connu des progrès considérables. Malgré ces avancées, le domaine fait encore face à de nombreux défis.

Afin de réaliser cette tâche, notre approche repose sur un modèle encodeur-décodeur basé sur des techniques d’apprentissage profond. Plus précisément, nous utilisons une nouvelle architecture entièrement basée sur des transformateurs, construite autour du modèle BART, qui surmonte les limites des approches traditionnelles basées sur les RNN et les CNN dans AAC. Le mécanisme d’attention du modèle BART permet une meilleure modélisation des dépendances locales et globales dans les signaux audio. Notre modèle intègre VGGish pour extraire les embeddings audio à partir de spectrogrammes log-Mel, et le transformer BART, combinant un encodeur bidirectionnel et un décodeur autoregressif, pour générer les descriptions textuelles. Word embeddings sont produites à l'aide d'un tokeniseur BPE, qui est adapté au vocabulaire unique du dataset d’entraînement, ce qui permet de répondre aux exigences générales de la tâche de génération de captions. Dans le but d’améliorer la qualité des captions audio générées, nous avons réalisé plusieurs expériences en utilisant le dataset Clotho. Les résultats indiquent que notre modèle produit des descriptions plus précises et plus diversifiées que les approches de l’état de l’art existantes.

Mots-clés :

Automated Audio Captioning, Encodeur-Décodeur, Apprentissage Profond, Transformer, BART, VGGish, tokeniseur BPE, Clotho.

ABSTRACT

The main objective of our project is to develop an effective system for Automated Audio Captioning (AAC), a task that involves describing ambient sounds within an audio clip using a natural language sentence, effectively bridging the gap between auditory perception and linguistic expression. In recent years, AAC has gained significant attention and has seen considerable progress. Despite these advancements, the field still faces many challenges.

To achieve this task, our approach follows an encoder-decoder model based on deep learning techniques. Specifically, we employ a novel, fully transformer-based architecture built around BART, which overcomes the limitations of traditional RNN and CNN approaches in AAC. The self-attention mechanism in BART facilitates better modeling of both local and global dependencies in audio signals. Our model integrates VGGish to extract audio embeddings from log-Mel spectrograms, and a BART transformer combining a bidirectional encoder and an autoregressive decoder for generating captions. Word embeddings are produced using a BPE tokenizer, which is adapted to the unique vocabulary of the training dataset, thereby aligning it with the general requirements of the captioning task. In order to improve the quality of the generated audio captions, we performed multiple experiments using the Clotho dataset. The results indicate that our model produces more accurate and diverse descriptions than existing state-of-the-art approaches.

Keywords:

Automated Audio Captioning, Encoder-Decoder, Deep Learning, Transformer, BART, VGGish, BPE tokenizer, Clotho.

ملخص

الهدف الرئيسي من مشروعنا هو تطوير نظام فعّال للتعليق الصوتي الآلي(AAC) ، وهي مهمة تتمثل في وصف الأصوات المحيطة داخل مقطع صوتي باستخدام جملة بلغة طبيعية، مما يساهم في سد الفجوة بين الإدراك السمعي والتعبير اللغوي. في السنوات الأخيرة، حظي هذا المجال باهتمام كبير وحقق تقدمًا ملحوظًا. وعلى الرغم من هذه التطورات، لا يزال يواجه العديد من التحديات.

لتحقيق هذه المهمة، يعتمد نهجنا على نموذج مُشفِّر-فكِّ تشفير قائم على تقنيات التعلم العميق. وتحديدا، نستخدم بنية جديدةً قائمة بالكامل على المحولات، مبنيةً على تقنية BART، والتي تتغلب على قيود مناهج RNN وCNN التقليدية في الترجمة الصوتية الآلية. تُسهِّل آلية الانتباه الذاتي في BART نمذجة أفضل للتبعيات المحلية والعالمية في الإشارات الصوتية. يدمج نموذجنا VGGish لاستخراج التضمينات الصوتية من مخططات الطيف log-Mel، ومحول BART، الذي يجمع بين مشفر ثنائي الاتجاه وفك تشفير تلقائي الانحدار، لتوليد أوصاف نصية. يتم إنتاج تضمينات الكلمات باستخدام مجزئ BPE، والذي تم تكييفه مع المفردات الفريدة لمجموعة البيانات المستخدمة في التدريب، مما يجعله متوافقًا مع المتطلبات العامة لمهمة توليد التعليقات التوضيحية. من أجل تحسين جودة التعليقات الصوتية المُولدة، أجرينا العديد من التجارب باستخدام مجموعة بيانات Clotho.وتشير النتائج إلى أن نموذجنا ينتج أوصافا أكثر دقة وتنوعا مقارنة بأحدث الأساليب المتوفرة.

الكلمات المفتاحية

التعليق الصوتي الآلي، مُشفِّر-فكِّ تشفير، التعلم العميق، المحول، BART، VGGish، مجزئ BPE، Clotho.

Remerciement

Nous remercions avant tout Dieu Allah le tout puissant de nous avoir donné la patience et la force pour atteindre ce stade du savoir et pour mener à bien ce modeste travail.

Notre gratitude va à notre promotrice, Madame YKHLEF H., pour sa patience, sa disponibilité constante et surtout ses judicieux conseils, qui ont contribué à alimenter notre réflexion.

Nous tenons à remercier les membres du jury pour le temps qu'ils ont consacré à l'évaluation attentive de ce travail.

Un remerciement tout particulier à nos familles, pour leur soutien, leurs encouragements, et leur patience face à nos longues heures de travail.

Merci à tous.

Dédicace

Je dédie ce travail

À ma tendre mère Souad et à mon très cher père Ali, pour leur amour, leurs sacrifices et leur soutien inconditionnel. Je souhaite qu'ils trouvent en moi la source de leur fierté.

À mes tantes, Hayat et Bachira, pour leur bienveillance et leurs encouragements.
À mes sœurs, Fatima et Marwa, et à mes frères, Abderrahim et Haron, pour leur présence et leur appui moral.

J'exprime également toute ma gratitude à mes familles Gharouba et Dahmani pour leur sollicitude et leur générosité.

À mes meilleurs amis, Kawther, Sara et Ahlem, pour leur soutien indéfectible, leur écoute bienveillante et les moments inoubliables partagés tout au long de ce parcours.

Et enfin, à tous mes professeurs, pour leur enseignement passionné tout au long de mes études.

Gharouba Hadil

Dédicace

J'offre ce projet

À ma mère et à mon père, qui ont toujours été à mes côtés par leurs prières, leur soutien et leur affection.

À mon frère Aboubakar, à mes sœurs Mariam, Inaam et Ikhlas, ainsi qu'aux

enfants de ma sœur : Gaith, Jawad et Abderrahmane — ma joie ne serait pas complète sans vous.

J'adresse également toute ma gratitude à la famille Ben Doumia et à la famille Baghdad.

À mes amis que j'ai rencontrés tout au long de mon parcours scolaire, et à Hadil, qui a toujours été là pour moi et m'a soutenu à chaque instant. Ce projet est pour vous, merci d'être présents dans ma vie.

Ben Doumia kaouther

Table des matières

Table des figures

Liste des tableaux

Liste des acronymes et abréviations

AAC	Automated Audio Captioning
ADC	Convertisseur Analogique-Numérique
DAC	Convertisseur Numérique-Analogique
FT	Transformée de fourier
STFT	Transformée de Fourier à Court Terme
DFT	Transformée de Fourier Discréte
NLP	Traitement du Langage Naturel
AI	Intelligence Artificielle
BPE	Byte Pair Encoding
CBOW	Continuous Bag-Of-Words
SOTA	State Of The Art
GPT	Generative Pre-trained Transformer
BERT	Bidirectional Encoder Representations from Transformers
BART	Bidirectional and Auto-Regressive Transformers
ML	Apprentissage Automatique
ANN	Réseau de Neurones Artificiels
CNN	Réseau de Neurones Convolutif
DNN	Deep Neural Network
ReLU	Unité Linéaire Rectifiée
FFN	Réseau de Neurones Feed-Forward
MLP	Perceptron Multicouche
RNN	Réseau de Neurones Récurrent
VGG	Visual Geometry Group
OOV	Out-Of-Vocabulary

Introduction

Contexte et Problématique

Le son est considéré comme le deuxième sens le plus important après la vue, étant capable de transmettre des informations essentielles sur l'environnement. Il offre une perception riche de notre entourage, allant des paysages acoustiques globaux, comme le bourdonnement du trafic urbain ou la pluie tombant dans une forêt, jusqu'aux événements sonores individuels tels que la fermeture d'une porte, les pleurs d'un bébé ou le retentissement d'une sirène au loin. Chez les êtres humains, la capacité à percevoir et interpréter ces signaux auditifs est une compétence innée. En revanche, permettre à une machine d'extraire et d'interpréter automatiquement des informations significatives à partir du son demeure un défi complexe [1]. Avec le développement rapide de l'apprentissage automatique, le domaine de l'écoute automatique a connu des avancées sans précédent ces dernières années, notamment dans des tâches comme l'audio tagging [2], la détection d'événements sonores [3] et la classification de scènes acoustiques [4]. Ces tâches se concentrent généralement sur l'identification et l'étiquetage d'événements sonores individuels ou d'environnements acoustiques globaux. Cependant, elles peinent souvent à capturer les relations sémantiques et contextuelles riches présentes dans les scènes audio réelles.

Cette limite a conduit à l'émergence d'une tâche plus avancée et plus globale, appelée l'Automatic Audio Captioning (AAC). Contrairement aux approches traditionnelles qui se limitent à classifier ou détecter des éléments isolés, l'AAC vise à générer des descriptions textuelles libres qui reflètent à la fois le contenu et le contexte d'un signal audio. Par exemple, plutôt que de simplement identifier « barking dog » et « passing car », un système AAC peut produire une description comme « a dog is barking while a car passes by on a busy street », capturant ainsi non seulement les événements sonores mais aussi leurs relations temporelles et spatiales. Cette capacité ouvre de nouvelles perspectives pour la compréhension intelligente de l'audio dans divers domaines d'application, allant des technologies d'assistance à l'indexation multimédia et à la surveillance.

Cependant, l'AAC soulève plusieurs défis majeurs. Il s'agit notamment de la reconnaissance précise des événements sonores dans des environnements souvent bruyants et comportant plusieurs sources sonores simultanées, de la modélisation des interactions complexes entre ces événements, ainsi que de la génération d'un langage naturel fluide, pertinent et contextuellement cohérent. À cela s'ajoutent la grande diversité des scènes sonores, leur variabilité temporelle, en plus de la subjectivité inhérente à l'interprétation des sons. Par conséquent, la problématique centrale de l'AAC réside dans la capacité à doter les systèmes informatiques d'une compréhension sémantique fine de l'environnement sonore et à produire des descriptions textuelles précises, informatives et adaptées à différents contextes d'usage.

Objectifs et Contributions

Ce travail vise à traiter les principaux défis mentionnés précédemment, dans le but de contribuer au développement du domaine de l'AAC. Notre recherche est guidée par les objectifs spécifiques suivants :

- Améliorer les performances des systèmes d'AAC en générant des descriptions textuelles de haute qualité, sémantiquement précises, reflétant fidèlement le contenu des extraits audio.
- Explorer la génération de descriptions audio diversifiées et contextuellement riches, afin d'accroître la variabilité des sorties du modèle et de mieux représenter la diversité d'interprétation des signaux audio.
- Réaliser plusieurs expériences et mener une étude comparative avec d'autres modèles encodeurs-décodeurs afin d'identifier le choix le plus pertinent pour la tâche d'AAC.

Les contributions de ce travail sont systématiquement alignées avec les objectifs énoncés précédemment et se résument comme suit :

- Nous adoptons une architecture de type encodeur-décodeur reposant sur le modèle Transformer BART.
- Nous exploitons le modèle VGGish préentraîné afin d'extraire des embeddings audio pertinentes pour améliorer l'efficacité du système AAC.
- Nous utilisons la technique BPE, qui permet une prise en charge robuste des entrées audio variées ainsi que des mots rares.

- Nous avons réalisé diverses expérimentations liées à la configuration de BART, en variant le nombre de couches et le type de décodage, sur le dataset Clotho afin de trouver un équilibre entre efficacité computationnelle et qualité des descriptions générées.
- Nous développons une application web d'AAC pour générer des descriptions textuelles des événements sonores dans l'audio.

Organisation de Mémoire

Ce mémoire se compose de deux parties principales. La première partie couvre les notions de pointe nécessaires à la compréhension des idées développées dans ce mémoire. Le **chapitre 1** est également divisé en deux parties : la première donne un aperçu des caractéristiques acoustiques utilisées pour représenter les signaux audio. Plus précisément, nous présentons les principales techniques d'extraction de caractéristiques couramment utilisées. Quant à la deuxième partie de ce chapitre, elle s'intéresse au traitement du langage naturel (NLP) et aux techniques associées. Dans le **chapitre 2**, nous présentons l'architecture de certains modèles d'apprentissage profond fondamentaux, ainsi que celle des modèles séquence à séquence. La seconde moitié de ce mémoire est consacrée au modèle proposé pour la tâche d'AAC**.** Le **chapitre 3** fournit une description détaillée du protocole expérimental, incluant le prétraitement, l'extraction des caractéristiques et l'architecture encodeur-décodeur utilisée. Dans le **chapitre 4**, nous présentons les résultats des expériences obtenues à travers des tableaux de performance et des analyses comparatives. Enfin, nous concluons en résumant les contributions de ce mémoire, en soulignant ses limites, et en proposant des pistes pour des travaux futurs.

PARTIE I : FONDAMENTAUX DE L'AUDIO CAPTIONING

Dans cette partie, nous présentons les concepts fondamentaux nécessaires à la compréhension des idées développées dans ce mémoire. Elle est structurée en deux chapitres principaux. Le Chapitre 1 se divise en deux parties. La première est consacrée aux bases du traitement du signal audio. Nous y introduisons le concept d'Automated Audio Captioning (AAC), ainsi que les caractéristiques physiques du son. Nous mettons également en lumière l'importance de l'extraction de caractéristiques pour convertir le signal en une représentation exploitable. La seconde partie se concentre sur le traitement du langage naturel (NLP) et ses différentes techniques, essentielles à la génération automatique de descriptions textuelles. Nous abordons également les défis spécifiques liés au NLP. Le Chapitre 2 est consacré à l'apprentissage profond, avec un accent particulier sur les architectures utilisées dans notre travail, notamment le modèle séquence-à-séquence (Seq2Seq), largement adopté pour l'AAC. Nous discutons également des observations empiriques et théoriques concernant les différences entre ces architectures. Enfin, nous abordons les défis spécifiques liés à la recherche en AAC. Cette partie établit ainsi les bases théoriques indispensables pour aborder les développements ultérieurs de ce mémoire.

Chapitre 1 : Généralités sur l'Audio Captioning

1.1 Introduction

Le son constitue un vecteur fondamental d'information, jouant un rôle central dans notre perception de l'environnement. L'analyse automatique du son est devenue un enjeu majeur en intelligence artificielle, notamment avec le développement de l'audio captioning, dont l'objectif est de générer automatiquement des descriptions textuelles à partir d'extraits audio. Cette tâche repose sur la combinaison du traitement du signal audio, pour l'extraction des caractéristiques pertinentes, et du traitement du langage naturel (NLP), qui les transforme en phrases compréhensibles.

Ce chapitre est structuré en deux parties complémentaires. La première partie est consacrée au traitement du signal audio. Elle commence par une définition de l'audio captioning, suivie d'une présentation du son et de ses composantes. Elle aborde ensuite la numérisation et la restitution du signal analogique, puis examine différentes représentations temporelles et fréquentielles utilisées pour l'extraction des caractéristiques acoustiques. La seconde partie est dédiée au traitement du langage naturel. Elle explore les techniques de prétraitement textuel, les méthodes de représentation vectorielle des mots, ainsi que les modèles de langage modernes, tout en mettant en lumière les défis linguistiques spécifiques au NLP, qui conditionnent la qualité et la cohérence des descriptions produites.

1.2 Automated Audio Captioning

L'Automated Audio Captioning (AAC) est une tâche intermodale qui établit un lien entre le traitement du signal audio et le traitement du langage naturel**.** L'objectif est de générer une description écrite en langage naturel, appelée caption pour un enregistrement audio donné [5]. Contrairement à la reconnaissance vocale automatique (ASR), qui se limite à transcrire la parole contenue dans un enregistrement, l'AAC va plus loin en produisant un résumé descriptif en une seule phrase, mettant en évidence les événements sonores dominants et les scénarios présents dans les échantillons audio [6]. Ces captions doivent être informatives, concises et

grammaticalement correctes. Elles décrivent non seulement les sons présents par exemple, « des oiseaux chantent » ou « un chien aboie », mais aussi le contexte ou les interactions possibles comme « une foule applaudit après un discours ». Cela requiert une compréhension approfondie des caractéristiques acoustiques, ainsi qu'une capacité à traduire ces informations en langage humain compréhensible. Un système AAC basique est présenté dans la Figure 1.1.

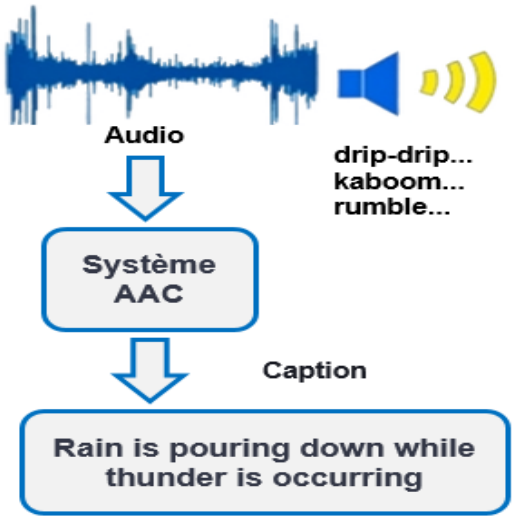

Figure 1-1: Visualisation fondamentale d'un système AAC [7].

L'AAC peut être utilisée dans divers scénarios, tels que l'amélioration de l'accessibilité pour les personnes malentendantes, en générant automatiquement des descriptions audio pour les vidéos en ligne ou les annonces sonores dans les transports publics, la surveillance de sécurité dans des environnements urbains innovants avec la détection d'événements sonores inhabituels tels que des bris de verre, des alarmes ou des cris, ainsi que la facilitation de l'organisation efficace des contenus audio grâce à l'indexation utilisée dans les bases de données multimédias, les archives sonores ou les plateformes de streaming.

1.3 Le Son

Le son est une vibration mécanique qui se propage dans un milieu matériel comme l'air, l'eau ou les solides sous forme d'ondes longitudinales. Sa transmission dépend de l'existence d'un milieu physique, car il ne peut pas se propager dans le vide.

Le son possède des caractéristiques telles que la fréquence, l'intensité et la longueur d'onde, qui déterminent sa nature et ses propriétés [8].

- **La fréquence**

  La fréquence est le nombre d'oscillations ou de cycles qu'une onde effectue par unité de temps. Elle se mesure en hertz (Hz). La fréquence détermine si un son est aigu ou grave : les sons à haute fréquence sont perçus comme aigus, tandis que ceux à basse fréquence sont perçus comme graves [9].

- **L'intensité sonore**

  L'intensité sonore est la quantité d'énergie transportée par une onde sonore par unité de surface. Elle est mesurée en watts par mètre carré (W/m^2) et est souvent exprimée en décibels (dB). L'intensité détermine la perception du volume sonore: plus l'intensité est élevée, plus le son est fort [10].

- **La longueur d'onde**

  La longueur d'onde est la distance entre deux points consécutifs d'une onde qui sont en phase, par exemple, entre deux crêtes successives [11]. Elle est mesurée en mètres (m) et est liée à la fréquence $\boldsymbol{f}$ et à la vitesse $\boldsymbol{v}$ du son par la relation :

$$\boldsymbol{\lambda = \frac{v}{f}} \qquad (1.1)$$

1.4 La Numérisation et la Restitution du Signal Analogique

La conversion du son entre le signal analogique et numérique est réalisée par le ADC pour son traitement numérique, puis par le DAC pour sa restitution sous forme de signal analogique audible, comme illustré dans la figure 1.2.

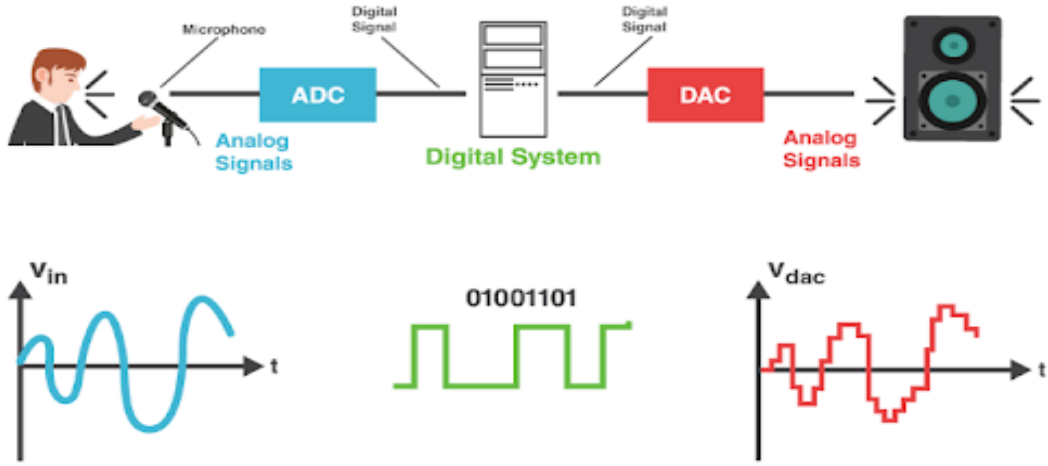

Figure 1-2: Le traitement du son [12].

1.4.1 Signal Analogique

Un signal analogique est un signal continu qui varie de manière fluide dans le temps et peut prendre une infinité de valeurs dans une plage donnée (voir figure 1-4a). Il est généralement représenté par des grandeurs physiques comme la tension, le courant ou la pression acoustique. Les signaux analogiques sont utilisés dans les systèmes audio, vidéo et de communication, mais leur sensibilité aux interférences et au bruit limite leur fiabilité dans certaines applications [13].

1.4.2 Signal Numérique

Un signal numérique est un signal discret qui prend un ensemble fini de valeurs, généralement représenté sous forme binaire (0 et 1). Contrairement au signal analogique, le signal numérique est robuste face au bruit et aux interférences, ce qui le rend idéal pour le stockage et la transmission d'informations dans les systèmes informatiques et électroniques modernes (voir figure 1-3) [14].

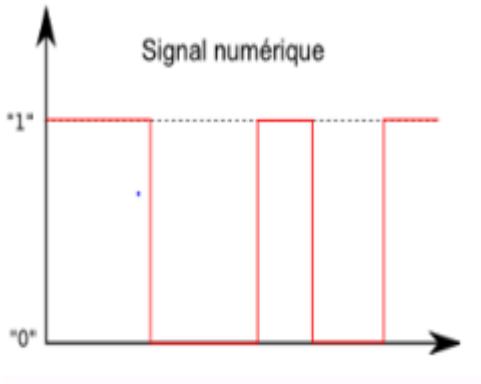

Figure 1-3: Exemple d'un signal numérique [15].

1.4.3 Le Convertisseur Analogique-Numérique (ADC)

ADC est un circuit électronique qui transforme un signal analogique en un signal numérique. Ce processus se déroule en deux étapes : l'échantillonnage du signal analogique à une certaine fréquence (taux d'échantillonnage), suivi de la quantification en valeurs numériques discrètes (voir figure 1-4). La qualité de la conversion dépend du taux d'échantillonnage (SR) et de la profondeur de bits (Bit Depth) [14].

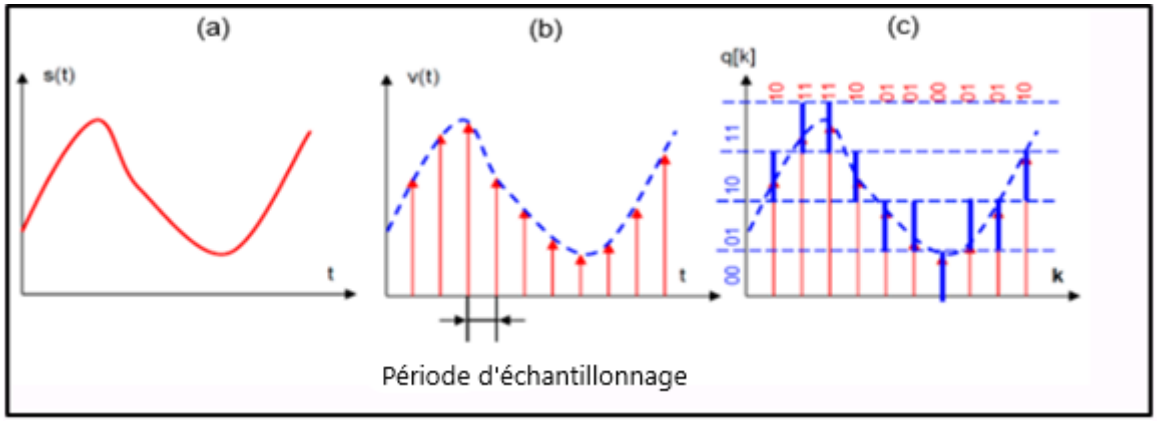

Figure 1-4: (a) signal analogique (b) signal échantillonné (c) puis quantifié [16].

1.4.4 Le Convertisseur Numérique-Analogique (DAC)

DAC est un circuit électronique qui effectue l'opération inverse de l'ADC. Il transforme les données numériques stockées ou traitées par l'ordinateur en un signal analogique pouvant être reproduit par des haut-parleurs ou des écouteurs. Ce processus consiste à reconstruire l'onde sonore analogique à partir des échantillons numériques, en appliquant des techniques de filtrage pour réduire les distorsions [13].

1.4.5 Étapes de la Conversion Analogique-Numérique (ADC) et Numérique-Analogique (DAC)

Tableau 1-1: Les étapes de ADC et DAC [8], [9].

Processus	ADC	DAC
Étape 1	**L'échantillonnage :** Est le processus de prise de mesures périodiques d'un signal analogique à des intervalles de temps réguliers afin de le convertir en une suite de valeurs discrètes (voir figure 1-4b). La fréquence d'échantillonnage, exprimée en hertz (Hz), détermine le nombre d'échantillons prélevés par seconde et influence la fidélité du signal numérisé.	**Déquantification :** Consiste à reconstruire un signal continu à partir de valeurs discrètes obtenues après la quantification. Ce processus implique l'attribution d'une valeur analogique approximative à chaque niveau de quantification, ce qui peut entraîner une légère perte d'information due au bruit de quantification.

Étape 2	**Quantification :** Consiste à arrondir les valeurs échantillonnées d'un signal analogique à l'un des niveaux discrets prédéfinis (voir figure 1-4c). Cela permet de représenter le signal sous forme numérique avec un nombre limité de bits. La précision de cette conversion dépend du nombre de bits utilisé, ce qui influence la qualité du signal reconstitué et le bruit de quantification.	**Filtrage :** Modifier ou améliorer un signal en supprimant certaines fréquences indésirables ou en amplifiant des composantes spécifiques. En traitement du signal, les filtres peuvent être analogiques ou numériques et sont classés en plusieurs types, tels que les filtres passe-bas, passe-haut, passe-bande et coupe-bande, selon les fréquences qu'ils laissent passer ou atténuent.

1.5 Représentation du Signal Audio

La représentation du signal dans le domaine temporel est utilisée depuis des années, car elle correspond à la perception humaine des sons. Cependant, cette approche ne permet pas d'examiner toutes les caractéristiques du signal [17]**.** Il devient donc nécessaire de le transformer en domaine fréquentiel à l'aide de la Transformée de Fourier (FT). Toutefois, lors de l'analyse d'un signal temporel fini, des phénomènes de fuite spectrale (en anglais : spectral leakage) peuvent survenir, car les signaux ayant des fréquences autres que celles de l'ensemble de base ne sont pas périodiques sur la fenêtre d'observation. L'extension périodique d'un signal dont la période naturelle n'est pas commensurable avec la fenêtre d'observation introduit des discontinuités aux bords de la fenêtre, qui sont responsables des fuites spectrales [18].

1.5.1 Fenêtrage (windowing)

Est la méthode la plus couramment utilisée pour réduire la fuite spectrale. Pour ce faire, on force l'amplitude de la séquence temporelle d'entrée à tendre progressivement vers une valeur commune au début et à la fin de l'intervalle d'échantillonnage[19].

Le processus de fenêtrage consiste à multiplier chaque trame par une fonction fenêtre (voir figure 1-5), ce qui permet d'atténuer le signal près des bords et de mettre en valeur la partie

centrale. Parmi les fonctions les plus couramment utilisées, on trouve les fenêtres de Hamming, Hann et Blackman.

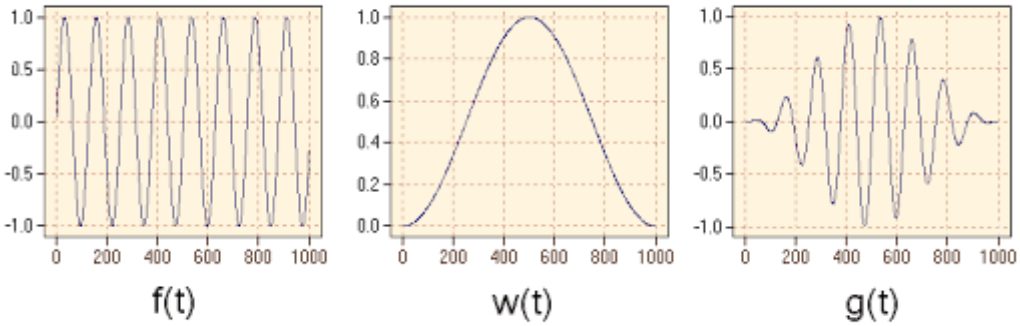

Figure 1-5: Fonction de fenêtrage : le signal original f(t) est multiplié par une fenêtre de Hann w(t), donne la fonction g(t)

1.5.2 Transformée de Fourier

La transformée de Fourier (FT) permet de passer d'une représentation du signal dans le domaine temporel à une représentation dans le domaine fréquentiel. Au lieu d'observer l'évolution du signal au cours du temps (voir Figure 1-6).

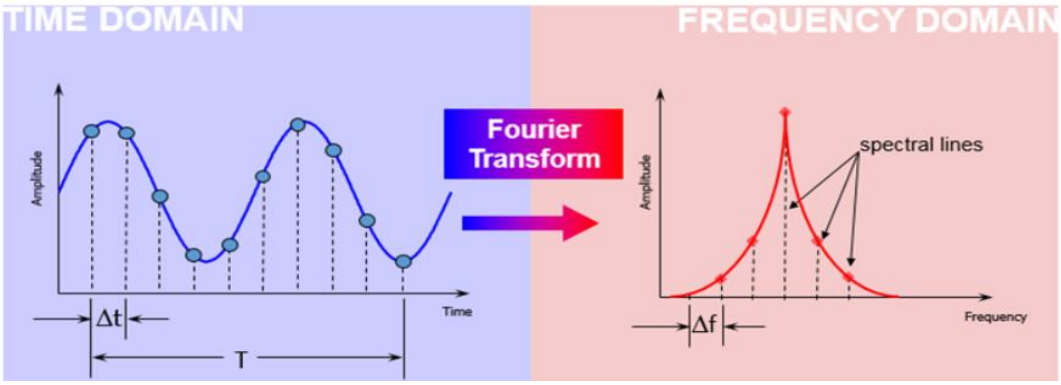

Figure 1-6: Passage du domaine temporel au domaine fréquentiel à l'aide de la FT [20].

L'idée principale de l'analyse de Fourier est de comparer le signal avec des sinusoïdes de différentes fréquences (mesurées en Hz). Chaque sinusoïde ou ton pur peut être considéré comme une oscillation de référence. Pour chaque fréquence considérée on obtient un coefficient d'amplitude, ainsi qu'un coefficient de phase. Si le coefficient est grand, cela signifie qu'il y a une forte similitude entre le signal et la sinusoïde de fréquence, indiquant que le signal contient une oscillation périodique à cette fréquence (voir figure 1-7b). À l'inverse, si

est petit, le signal ne contient pas de composante périodique à cette fréquence (voir figure 1-7c). De plus, comme l'illustre la figure 1-7d, on observe une forte similarité entre le signal et la sinusoïde dont la fréquence est égale à 523 Hz.[21]

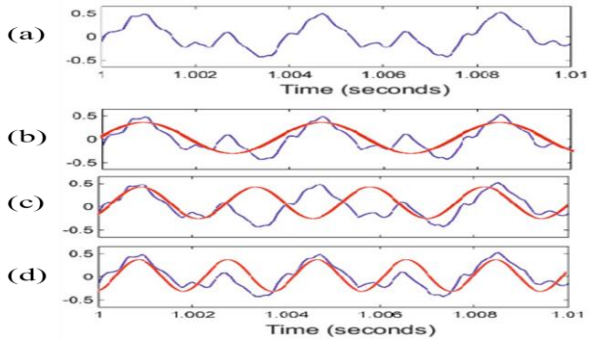

Figure 1-7: (a) Zoom sur une section de 10 millisecondes d'une forme d'onde (b-d) Comparaison de la forme d'onde avec des sinusoïdes de différentes fréquences [22].

La définition mathématique de la Transformation de Fourier en temps continu est :

$$\mathrm{X}(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} x(t)\, e^{-i\omega t} dt \tag{1.2}$$

Où x(t) représente le signal original, X(ω) est sa représentation dans le domaine fréquentiel, i est l'unité imaginaire, ω est la fréquence angulaire et t est l'indice temporel. Dans le cas d'un signal discret, une autre forme de FT est utilisée, appelée la Transformée de Fourier Discréte (DFT), définie comme suit :

$$\mathrm{X(K)} = \sum_{n=0}^{N-1} x(n) e^{-2\pi i k n / N} \quad \text{pour k} = 0, 1, \ldots . \mathrm{N} - 1 \tag{1.3}$$

1.5.3 Transformée de Fourier à Court Terme

D'après les équations ci-dessus (si l'on considère un signal en temps continu), on observe que la Transformée de Fourier suppose que le signal est analysé sur toute la durée, c'est-à-dire une durée infinie. Cela implique qu'il n'existe aucune notion de temps dans le domaine fréquentiel, tout comme il n'y a aucune notion de variation de fréquence dans le temps. Les deux domaines ne peuvent pas être mélangés ; ils sont orthogonaux l'un par rapport à

l'autre[23] . La FT fournit des informations sur les fréquences présentes dans un signal, mais elle ne précise pas le moment où ces fréquences apparaissent dans le temps. Pour retrouver la relation entre la fréquence et le temps, il est nécessaire d'utiliser la Transformée de Fourier à Court Terme (STFT).

L'idée principale de STFT est de considérer uniquement une petite section du signal. Pour cela, on utilise une fonction fenêtre uniquement sur une courte période de temps. Le signal d'origine est ensuite multiplié par cette fonction fenêtre, ce qui donne un signal fenêtré (comme celles mentionnées précédemment). Pour obtenir des informations de fréquence à différents instants, on décale la fonction de fenêtre dans le temps et on calcule une transformée de Fourier pour chacun des signaux fenêtrés résultants, comme illustré dans la figure 1-8 [21].

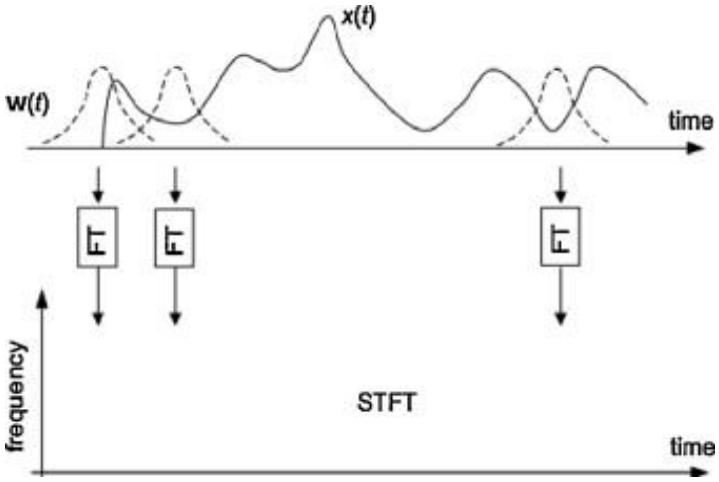

Figure 1-8: Transformée de Fourier à court terme STFT [24].

1.6 Méthodes d'Extraction de Caractéristiques Audio

L'extraction de caractéristiques (en anglais : Feature Extraction) est un processus qui consiste à mettre en évidence les propriétés les plus dominantes et discriminantes d'un signal. Une caractéristique appropriée reflète les propriétés du signal de manière compacte et significative. L'évolution des caractéristiques audio peut être subdivisée en trois principaux domaines du signal : le domaine temporel, le domaine fréquentiel et le domaine temps-fréquence conjoint, chacun offrant des perspectives uniques sur les propriétés du signal [25].

- **Domaine temporel :** Ces caractéristiques sont directement extraites de la forme d'onde audio brute, telles que le zero-crossing rate et l'enveloppe d'amplitude. Les

premières et les plus simples caractéristiques audio appartiennent à ce domaine. Les caractéristiques du domaine temporel ont évolué jusqu'à la fin des années 1950 et jouent encore un rôle essentiel dans l'analyse et la classification audio en raison de leur simplicité et de leur efficacité computationnelle.

- **Domaine fréquentiel :** Ces caractéristiques sont extraites après l'application de transformations telles que la transformée de Fourier. Elles fournissent des informations sur le contenu spectral du signal et incluent des caractéristiques comme le centroïde spectral et la dispersion spectrale (spectral spread). Des caractéristiques telles que la hauteur tonale (pitch) et les formants sont des exemples de caractéristiques du domaine fréquentiel qui sont largement utilisées dans diverses applications depuis les années 1950 et 1960.
- **Domaine temps-fréquence :** Ce domaine combine à la fois les informations temporelles et fréquentielles, offrant une vue plus détaillée de l'évolution des propriétés spectrales au cours du temps. Parmi les exemples, on trouve les spectrogrammes, les spectrogrammes Mel et les spectrogrammes log-Mel. Les algorithmes d'extraction de caractéristiques conjointes temps-fréquence ont été développés à la fin des années 1960 et sont depuis devenus des standards dans de nombreux algorithmes de traitement du signal audio.

1.6.1 Spectrogramme

La STFT est souvent utilisée pour l'analyse spectrographique des signaux vocaux. Le spectrogramme est une représentation graphique du spectre de puissance de la parole en fonction du temps, donnée par la formule suivante :

$$S(n, \omega) = |X(n, \omega)|^2 \tag{1.4}$$

Où X (n, ω) désigne STFT du signal x(n) [26]. Dans un spectrogramme, on distingue essentiellement deux axes : L'axe horizontal représente le temps, L'axe vertical représente la fréquence. La luminosité ou la couleur de chaque point dans l'image constitue une représentation tridimensionnelle (3D) de l'amplitude d'une fréquence donnée à un instant précis [27] (voir figure 1-9). Un spectrogramme est une excellente mesure du spectre d'un signal. Il représente la distribution spectrale du signal, utilisée pour former une image, et fournit des informations sur la variation de la densité spectrale au cours du temps [28].

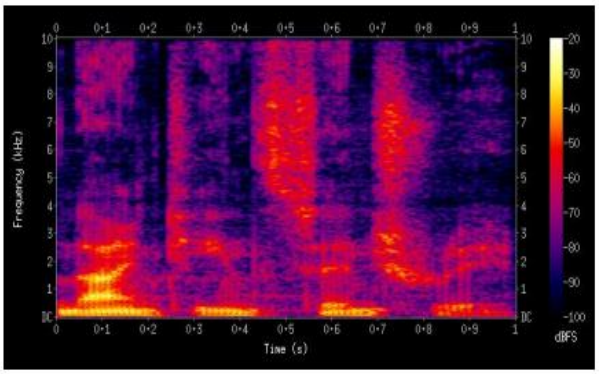

Figure 1-9: Exemple d'un spectrogramme.

1.6.2 Spectrogramme Mel

Un spectrogramme est une représentation visuelle de la composition fréquentielle d'un signal audio au fil du temps. Cependant, la perception humaine du son est non linéaire : nous distinguons mieux les variations de fréquence dans les basses fréquences que dans les hautes fréquences. Pour mieux refléter cette perception, l'échelle de Mel est utilisée. Elle fournit une échelle linéaire adaptée au système auditif humain et est liée aux Hertz [29]. Par la formule suivante, où m représente les Mels et f représente les Hertz :

$$m(f) = 2595\ log_{10}\ (1 + \frac{f}{700}) \tag{1.5}$$

La transformée inverse :

$$f\ (m)\ =\ 700\ (10^{m/2595}\ -\ 1) \tag{1.6}$$

Le spectrogramme Mel est une matrice composée de vecteurs de caractéristiques d'énergie des bandes Mel, concaténés pour des trames temporelles successives. Il est obtenu en appliquant la banque de filtres Mel à chaque trame temporelle sur le spectrogramme de magnitude. La banque de filtres Mel utilise l'échelle Mel et se compose de filtres triangulaires (voir figure 1-10) dont la largeur de bande s'élargit avec l'augmentation des fréquences centrales des filtres. Cela permet d'obtenir une meilleure résolution fréquentielle dans les basses fréquences et, inversement, une résolution plus faible dans les hautes fréquences [30].

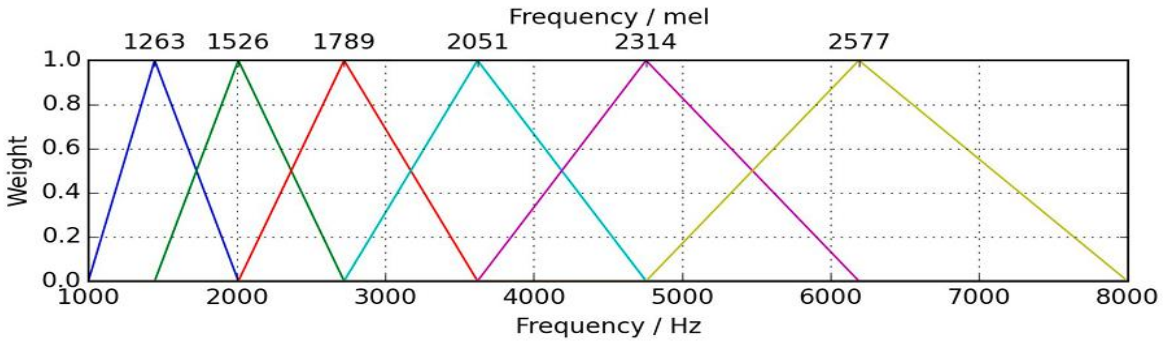

Figure 1-10: Exemple de filtres triangulaires utilisés dans la conversion du spectre fréquentiel en Mel.

1.6.3 Spectrogrammes Log-Mel

Le spectrogrammes log-Mel est un spectrogramme Mel sur lequel une transformation logarithmique est appliquée sur l'amplitude [31]. Cette opération permet de compresser la plage dynamique du signal, d'améliorer la robustesse au bruit, et de rapprocher davantage la représentation de la perception auditive humaine. Pour ces raisons, le spectrogrammes log-Mel est largement utilisé comme entrée dans les modèles d'apprentissage profond pour diverses tâches de traitement audio, telles que la reconnaissance vocale, la classification de sons ou l'analyse musicale. La figure 1-11 illustre un exemple de spectrogramme Mel et spectrogrammes log-Mel**.**

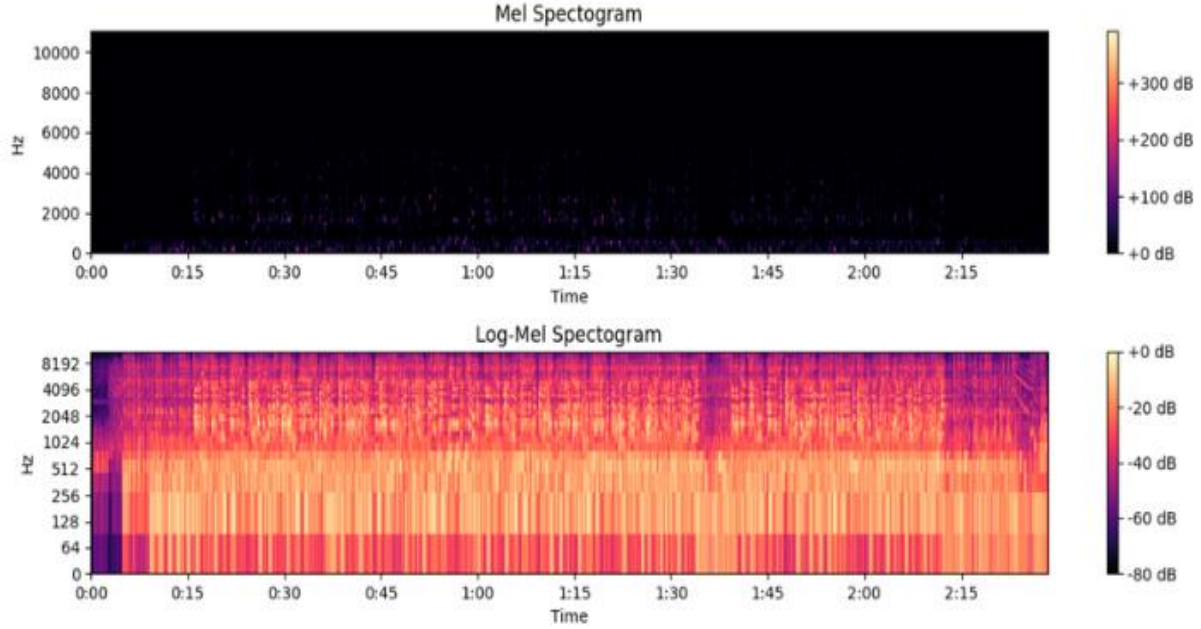

Figure 1-11: Exemple de Mel spectrogramme et Log-Mel Spectrogramme [31].

1.7 Le Traitement du Langage Naturel (NLP)

Le NLP est un sous-domaine de l'Intelligence Artificielle [32] qui permet d'effectuer des tâches telles que la traduction automatique, l'analyse des sentiments, la reconnaissance vocale, etc. L'objectif principal du NLP est de donner aux machines la capacité de comprendre, traiter et analyser des textes rédigés en langues humaines. [33]. Afin qu'une machine puisse exploiter ce type de données, il est nécessaire de prétraiter les données textuelles, en les nettoyant et en les structurant, ce qui facilite leur exploitation.

Après ce prétraitement, le texte est converti en une représentation sous forme de vecteurs numériques, permettant ainsi aux algorithmes d'apprentissage automatique de les comprendre et de les manipuler efficacement.

Le NLP est essentiel en AAC pour générer des captions cohérentes, correctes et riches de sens à partir du signal audio.

1.8 **Les Techniques de Prétraitement du Texte**

Le prétraitement du texte consiste à convertir les données brutes en une structure bien définie, dans laquelle les mots ne contribuant pas au sens contextuel de la phrase sont éliminés. Il existe différentes méthodes pour prétraiter un texte. Voici quelques-unes des approches les plus couramment utilisées.

1.8.1 Tokenisation

La tokenisation est une méthode de prétraitement qui divise un flux de texte en mots, phrases, symboles ou autres éléments significatifs appelés tokens [34]. Il existe de nombreux techniques de tokenisation, notamment la tokenisation par mot consistent à découper le texte en mots individuels, tandis que la tokenisation par caractère représente le texte comme une séquence de caractères individuels. La tokenisation par sous-mots découpe le texte en unités plus petites que des mots entiers, mais plus grandes que des caractères isolés (voir la figure 1-12) [35]. Elle permet au modèle de traiter des mots inconnus en les décomposant en sous-mots familiers.

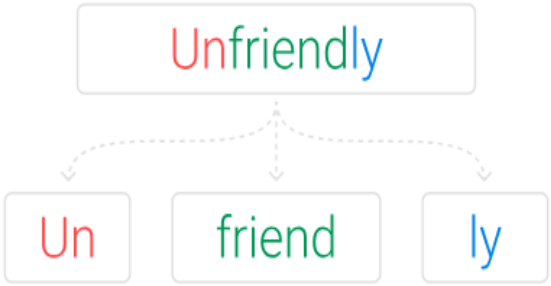

Figure 1-12: Exemple de tokenisation en sous-mots [36].

Les algorithmes les plus connus pour la tokenisation par sous-mots sont :

- **BPE (Byte Pair Encoding) :** Commence avec un vocabulaire initial contenant tous les symboles présents dans le dataset. Ensuite, il forme de nouveaux symboles en fusionnant de manière répétée les deux symboles les plus fréquents, jusqu'à atteindre la taille de vocabulaire souhaitée. Le BPE a été utilisé dans le modèle OpenAI GPT.
- **WordPiece :** Est similaire à BPE, mais au lieu de fusionner les symboles en fonction de leur fréquence, il les fusionne en fonction de la likelihood des données d'entraînement après l'ajout du nouveau symbole au vocabulaire. Il été utilisé dans de nombreux modèles de langage importants tels que BERT.
- **Unigram :** Commence avec un grand nombre de tokens et élimine progressivement certains d'entre eux pour obtenir un vocabulaire plus réduit. L'algorithme Unigram calcule une perte sur les données d'entraînement en utilisant le vocabulaire courant et un modèle de langage Unigram.
- **SentencePiece :** Traite l'entrée comme un flux brut de texte et inclut l'espace dans l'ensemble des caractères à utiliser. Ensuite, il applique les algorithmes BPE ou Unigram pour créer un vocabulaire adapté [37].

1.8.2 Suppression des Mots Vides

Le NLP consiste à extraire des mots-clés liés à un sujet particulier, en fonction du cas d'usage. Ainsi, pour des tâches telles que la classification de texte ou d'autres problématiques similaires, des mots comme « le », « un », « une » etc., ne sont pas importants et sont fréquemment éliminés. Ces mots sont appelés mots vides (stopwords) et doivent être

identifiés de manière aussi efficace que possible. La tokenisation du texte facilite grandement l'identification de ces mots sans difficulté majeure [38].

1.8.3 Stemming

Les algorithmes de stemming suppriment les suffixes ainsi que les flexions, afin que les variantes d'un mot puissent être regroupées sous leur radical commun [39]. Par exemple, si l'on considère les mots « rapidement, rapidité, rapide » le radical obtenu sera « rapid »

1.8.4 Lemmatisation

La lemmatisation est une transformation morphologique qui consiste à convertir un mot tel qu'il apparaît dans un texte en sa forme de base ou sa forme canonique, appelée lemme, en supprimant les terminaisons flexionnelles [40]. Ce processus est similaire au stemming, mais il ajoute une dimension sémantique en tenant compte du sens des mots. En termes simples, la lemmatisation permet de relier des mots ayant un sens similaire à une forme unique. Par exemple pour les mots « étaient, été, est » devient « être ».

1.9 La Représentation Vectorielle des Mots

1.9.1 One-Hot Encoding

La manière la plus simple de représenter un mot sous une forme lisible par un ordinateur est le vecteur one-hot. Ce vecteur a une dimension égale à la taille du vocabulaire et attribue la valeur 1 à l'index correspondant au mot représenté, tandis que toutes les autres positions sont remplies avec 0 (voir figure 1-13). Il est évident que les vecteurs one-hot contiennent très peu d'informations sémantiques sur les mots, à part le fait de les différencier les uns des autres [41].

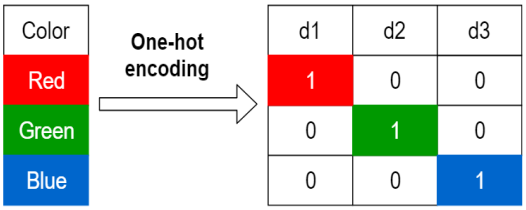

Figure 1-13: One-Hot Encoding [42].

1.9.2 Word Embeddings

Jurafsky et Martin (2000) définissent le word embedding comme la représentation des mots pour l'analyse de texte, généralement sous la forme d'un vecteur de nombres réels qui encode le sens du mot. Ainsi, les mots situés plus près les uns des autres dans l'espace vectoriel sont censés avoir des significations similaires [43]. Parmi les méthodes de word embedding les plus connues figurent Word2Vec, GloVe et FastText.

1.9.2.1 Word2Vec

Word2vec, ou "word-to-vector" est l'un des modèles les plus largement adoptés. Word2vec propose une gamme de modèles permettant de représenter les mots dans un espace n-dimensionnel, de manière à ce que les mots similaires ou ayant des significations proches soient placés à proximité les uns des autres. Cela justifie l'objectif global de représenter les mots dans un nouvel espace vectoriel. Comme illustré dans la figure 1-14 où les word embeddings sont représentés graphiquement, les mots ayant des significations similaires sont plus proches dans l'espace, ce qui indique leur similarité sémantique.

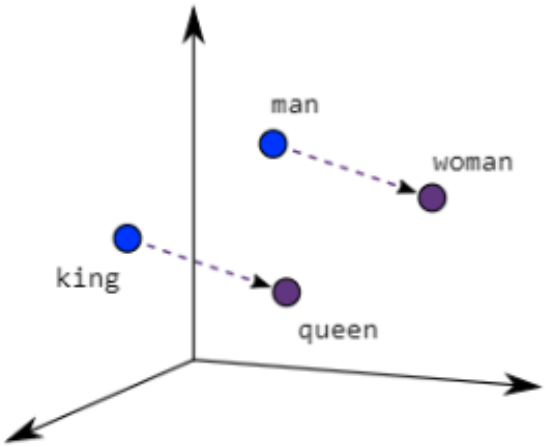

Figure 1-14: Représentation vectorielle sémantique des mots dans Word2Vec [44].

En comparaison avec One-Hot Encoding, Word2Vec permet de réduire la taille de l'espace d'encodage et compresse la représentation des mots à la longueur souhaitée pour le vecteur [45]. Parmi les modèles les plus couramment utilisés, Continuous Bag-Of-Words (CBOW) et Skip-Gram, Les deux modèles sont similaires sur le plan algorithmique, la différence résidant uniquement dans leur façon d'effectuer la prédiction.

- **CBOW :** Ce modèle utilise les mots de contexte comme entrée et tente de prédire le mot central correspondant. Elle prédit donc les mots centraux à partir de leur contexte (voir figure 1-15a)**.**
- **Skip-Gram :** À l'inverse, ce modèle utilise le mot central pour prédire les mots de contexte**.** Concrètement, chaque mot courant est fourni en entrée à un classificateur log-linéaire avec une couche de projection continue, et le modèle cherche à prédire les mots apparaissant dans une plage définie autour de ce mot (voir figure 1-15b). Les recherches ont montré que l'augmentation de cette plage améliore la qualité des vecteurs de mots obtenus, mais cela entraîne également une augmentation de la complexité computationnelle [46].

Chacun de ces modèles présente ses propres avantages et inconvénients. Selon Mikolov, le modèle Skip-Gram fonctionne bien avec de petites quantités de données et s'avère efficace pour représenter les mots rares. D'un autre côté, le modèle CBOW est plus rapide et offre de meilleures représentations pour les mots fréquents [47].

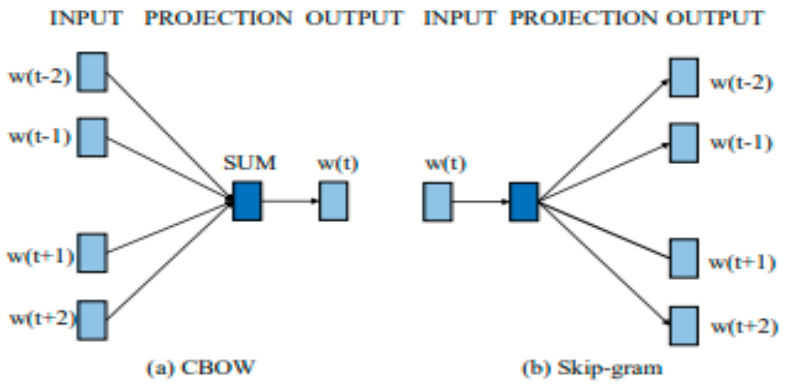

Figure 1-15: L'architecture CBOW et Skip-Gram de Word2Vec [48].

1.9.2.2 FastText

Ce modèle Apprend des informations morphologiques détaillées en représentant chaque mot sous forme de n-grammes de caractères plutôt que d'apprendre directement un vecteur unique par mot. Il exploite ainsi les informations sous-mot et la structure interne des mots pour améliorer la qualité des word embeddings [49]. Donc, FastText est similaire à Word2Vec**,** mais il est conçu pour gérer les informations sous-mot. Le principal avantage de fastText est

qu'il génère de meilleurs word embeddings pour les mots rares et inconnus pendant l'entraînement [50].

1.9.2.3 GLOVE

L'approche est très similaire à celle de Word2Vec, où chaque mot est représenté par un vecteur de grande dimension et est entraîné en fonction des mots environnants sur un vaste corpus [34]. GloVe construit un modèle en s'appuyant à la fois sur les avantages de la factorisation matricielle globale et sur l'utilisation de fenêtres de contexte locales. La factorisation matricielle globale consiste à réduire une grande matrice de fréquences d'apparition des termes en utilisant des méthodes de factorisation issues de l'algèbre linéaire. Cette matrice globale indique si des mots apparaissent ensemble dans un texte. Il est entraîné à partir d'une matrice de cooccurrence mot–mot globale et accumulée [50].

1.10 Les Modèles de Langage Modernes

Les modèles classiques tels que Word2Vec ou FastText associent un unique vecteur à chaque mot, sans prendre en compte son contexte. Ils sont donc limités face aux mots polysémiques et aux dépendances longues dans une phrase. Pour dépasser ces contraintes, les modèles de langage modernes utilisent des représentations contextuelles basées sur l'architecture Transformer, présentée en section 2.7 du chapitre 2. Ces modèles préentraînés sur de vastes corpus textuels peuvent générer, compléter ou comprendre le langage avec une bien meilleure précision. Ils constituent aujourd'hui l'état de l'art (SOTA) en traitement du langage naturel. Parmi les modèles de langage modernes les plus utilisés, on retrouve :

1.10.1 GPT (Generative Pre-trained Transformer)

Avant GPT, les modèles classiques de NLP étaient entraînés sur de grandes quantités de données annotées, spécifiques à une tâche précise. Cela représentait un inconvénient majeur, car il était difficile d'accéder à la quantité de données étiquetées nécessaire pour entraîner le modèle avec précision. Ces modèles classiques étaient incapables d'effectuer des tâches en dehors de leur training set, car ils étaient limités à un data set particulier. Pour contourner ces limitations, OpenAI a proposé un modèle de langage génératif appelé GPT-1 puis GPT-2 et GPT-3. Le modèle GPT produit de grandes quantités de texte pertinent et complexe généré par machine à partir d'une petite quantité de texte en entrée. Les modèles GPT peuvent être

considérés comme des modèles de langage qui imitent le texte humain en utilisant des techniques d'apprentissage profond. L'architecture de GPT repose sur le décodeur du Transformer. Le processus de pré-entraînement suit une approche autorégressive, dans laquelle chaque prédiction dépend des valeurs précédentes. Ce pré-entraînement permet au modèle d'apprendre des représentations riches du langage naturel, qui peuvent ensuite être fine-tuning pour accomplir des tâches spécifiques [51].

1.10.2 BERT (Bidirectional Encoder Representations from Transformers)

La principale limitation de ces modèles de langage classiques (word2v, FastText) est que les mots ayant plusieurs significations selon le contexte sont représentés par un seul vecteur**.** BERT résout ce problème en intégrant les informations contextuelles dans les deux directions (gauche et droite) lors de l'apprentissage des représentations des mots [52]. L'architecture du modèle BERT est un encodeur Transformer. L'amélioration essentielle par rapport à GPT est que BERT propose une solution pour rendre les Transformers bidirectionnels. Cette modification permet à BERT d'effectuer un conditionnement conjoint sur les contextes gauche et droit dans toutes les couches du modèle [53]. La modélisation du langage masqué (MLM) et la prédiction de la prochaine phrase (NSP) sont les deux principales tâches de pré-entraînement de BERT.

- **MLM :** Dans le MLM, un échantillon aléatoire de tokens dans la séquence d'entrée est sélectionné et remplacé par le token spécial [MASK] (voir figure 1-16). L'objectif est d'amener le modèle à prédire ces tokens masqués en utilisant le contexte des mots.

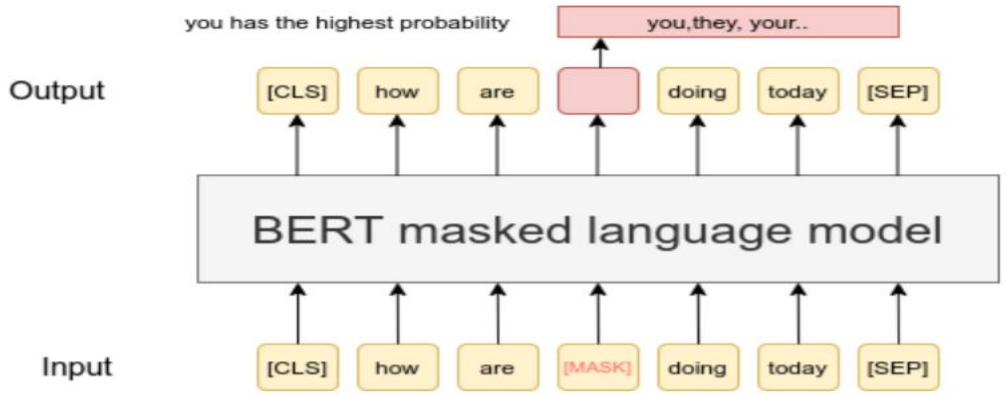

Figure 1-16: BERT – Phrase originale 'how are you doing today' [54].

- **NSP :** Le NSP, quant à lui, est une tâche de classification binaire où BERT doit déterminer si deux segments de texte se suivent dans le texte original ou s'ils proviennent de documents différents [55](voir l'exemple dans la figure 1-17). Ces deux mécanismes permettent à BERT d'acquérir une compréhension approfondie du langage et d'améliorer ses performances sur diverses tâches de NLP.

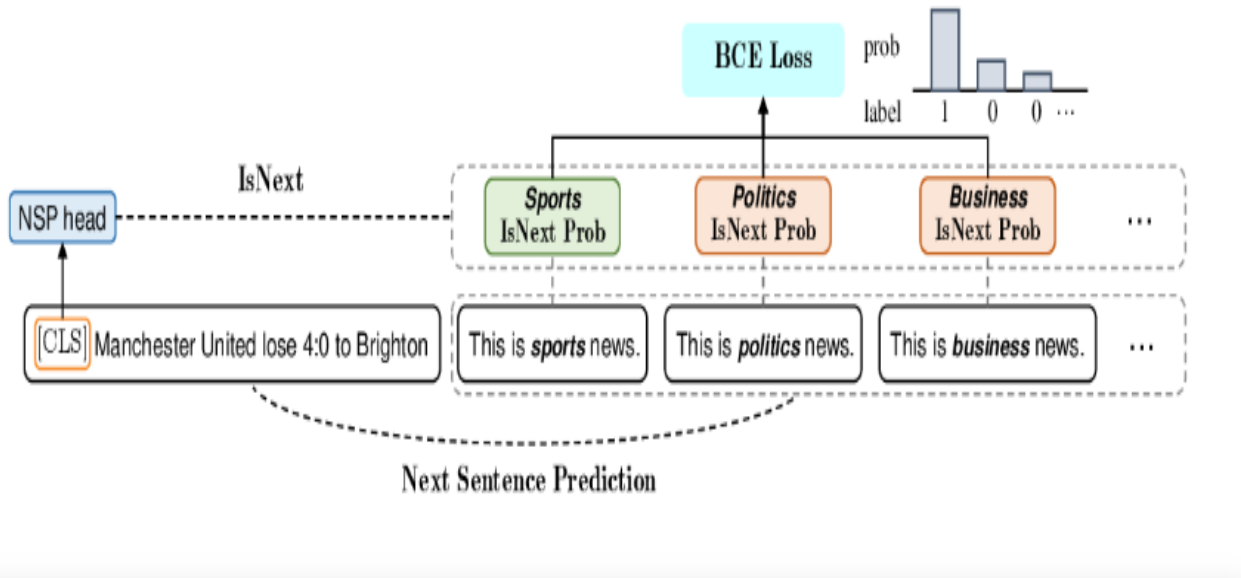

Figure 1-17: Optimisation NSP pour la classification des phrases individuelles. « Manchester United perd 4-0 contre Brighton » est l'entrée d'origine, l'étiquette dorée étant Sports. Les instances négatives sont générées avec des étiquettes erronées : Politique, Affai [56].

1.10.3 BART (Bidirectional and Auto-Regressive Transformers)

Le modèle BART combine les approches de BERT et de GPT. Il est constitué d'un encodeur bidirectionnel et d'un décodeur autorégressif (voir figure 1-18). Lors du pré-entraînement, des transformations bruitées sont appliquées à la séquence d'entrée, qui est alors corrompue à l'aide de symboles de masquage. Cette séquence altérée est ensuite traitée par l'encodeur bidirectionnel. Le décodeur autorégressif est ensuite utilisé pour reconstruire la séquence originale, en calculant la probabilité du document initial. Il n'est pas nécessaire que l'entrée et la sortie soient alignées de manière exacte. Toutefois, lors de la phase de fine-tuning, un document non corrompu est fourni simultanément à l'encodeur et au décodeur, et ce sont les représentations issues du dernier état caché du décodeur qui sont utilisées [57].

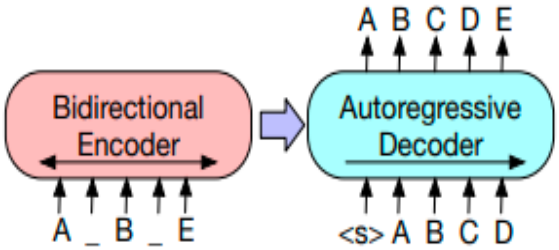

Figure 1-18: L'architecture du modèle BART [58].

1.11 Défis Linguistiques du NLP

D'un point de vue linguistique, les machines sont confrontées à plusieurs défis majeurs dans la compréhension du langage naturel. Contrairement aux humains, qui acquièrent intuitivement les règles de la langue à travers l'expérience, les machines doivent traiter le langage de manière formelle et systématique. Cela rend la modélisation du langage humain particulièrement complexe. Les défis suivants illustrent les principaux obstacles rencontrés dans ce domaine :

- **Diversité des langues** : Les différentes langues parlées à travers les pays et les régions rendent difficile l'adoption d'une approche standard par les machines pour les traiter.

- **Robustesse du langage** : Les machines doivent être capables de maintenir des performances constantes face à différents accents et styles de conversation.
- **Dépendance aux connaissances** : La compréhension du langage repose souvent sur des connaissances pratiques, ce qui constitue un défi commun pour la technologie et la langue.
- **Compréhension contextuelle** : Les machines doivent être capables d'interpréter le langage en fonction de son environnement et du contexte dans lequel il est utilisé [59].

- **AMBIGUÏTÉ**

  Dans l'ambiguïté structurelle, une phrase peut avoir plusieurs structures syntaxiques possibles. Dans l'ambiguïté syntaxique, une erreur de construction grammaticale dans une partie de la phrase entraîne une ambiguïté grammaticale dans l'ensemble de la phrase. L'ambiguïté lexicale, quant à elle, résulte d'un mot ayant deux significations différentes ou de deux mots ayant la même forme. Traiter le problème de l'ambiguïté

est essentiel dans l'analyse des retours (feedback). Dans une étude mentionnée dans [60], la désambiguïsation lexicale a été abordée en personnalisant BERT, un modèle de représentation du langage, et en sélectionnant les meilleures paires contexte-définition parmi un ensemble de paires similaires [61].

1.12 Conclusion

Ce premier chapitre a posé les bases fondamentales nécessaires à la compréhension de l'Automated Audio Captioning en introduisant le traitement du signal audio et le traitement du langage naturel. Nous avons exploré les composantes physiques du son, les processus de conversion analogique-numérique, ainsi que les représentations temporelles et fréquentielles permettant d'extraire les caractéristiques pertinentes du signal. Nous avons abordé les techniques de prétraitement textuel, les méthodes de représentation vectorielle des mots, ainsi que les modèles de langage modernes, tout en mettant en lumière les défis linguistiques liés au traitement du langage naturel. Cette double approche, combinant traitement acoustique et modélisation linguistique, est essentielle pour permettre aux machines de comprendre et de décrire le contenu sonore de manière cohérente et pertinente.

Chapitre 2 : Apprentissage Profond pour l'Audio Captioning

2.1 Introduction

L'Automated audio captioning s'appuie sur les progrès récents de l'intelligence artificielle (AI), en particulier du l'apprentissage automatique (ML), qui permet aux machines d'apprendre automatiquement à partir de grandes quantités de données audio. Parmi les sous-domaines du ML, on retrouve les réseaux de neurones profonds, notamment les CNN et les Transformers, capables de modéliser des relations complexes dans les données. Ces architectures sont souvent utilisées dans des modèles séquence à séquence basés sur une structure encodeur-décodeur, où une séquence audio en entrée est transformée en une séquence textuelle fluide et contextualisée.

Ce chapitre s'attache à explorer ces fondements en présentant, tout d'abord, les concepts clés du ML et des réseaux de neurones, puis les techniques d'entraînement et de régularisation. De plus, nous fournissons une brève description des architectures CNN et transformer, ainsi que du modèle séquence à séquence. Enfin, nous aborderons les défis spécifiques à l'AAC.

2.2 Apprentissage Automatique

L'apprentissage automatique (ML) est au cœur de l'intelligence artificielle, qui se préoccupe de la modélisation des données [62]. L'objectif est de développer des algorithmes qui peuvent apprendre des données, s'adapter aux nouvelles informations et prendre des décisions ou faire des prédictions en fonction des motifs détectés [63]. ML couvre un large éventail d'applications, notamment la reconnaissance vocale et d'images, les véhicules autonomes, le traitement du langage naturel et les systèmes de recommandation. Il occupe une place essentielle dans la résolution de problèmes complexes fondés sur l'analyse de données.

Le ML peut être classé en trois branches principales : l'apprentissage supervisé, où les modèles sont entraînés sur des données étiquetées afin d'établir une relation entre les entrées et les sorties, avec des applications telles que l'automated audio captioning [5], la classification d'images [64] ou l'analyse de sentiments [65]; l'apprentissage non supervisé vise à identifier des structures et des motifs cachés dans des données non étiquetées, illustré par des techniques comme le regroupement (clustering) [66] et la compression d'images [67]; et enfin, l'apprentissage par renforcement repose sur un système de récompenses et de pénalités afin d'optimiser la prise de décision [68], et il est appliqué, par exemple, aux jeux [69] ou aux véhicules autonomes [70].

2.3 Réseaux de Neurones Artificiels

Un réseau de neurones artificiels (ANN) est une méthode d'apprentissage automatique inspirée de la structure du cerveau humain [71]. Chaque neurone est considéré comme une unité de calcul qui effectue une tâche simple à partir des données en entrée. En interconnectant ces neurones au sein de différentes couches, le réseau devient capable de modéliser des relations complexes et d'apprendre à partir de données d'entraînement. Les composants les plus courants des réseaux neuronaux sont les suivants :

2.3.1 Neurone Artificiel

Le neurone artificiel est un élément de base des réseaux de neurones. Sa conception est inspirée de l'observation du neurone biologique, comme le montre la figure 2-1.

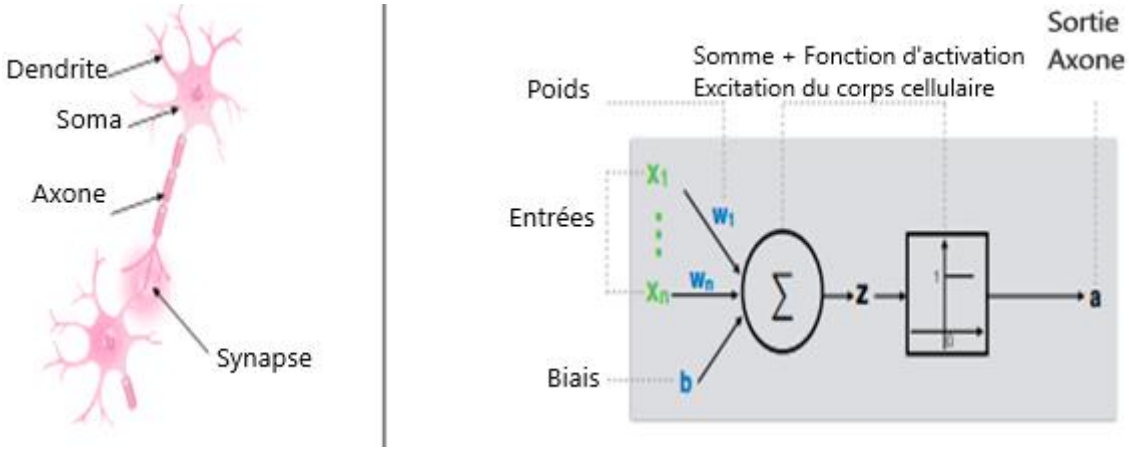

Figure 2-1: Conception de neurones biologiques et artificiels [71] .

Dans un neurone biologique, l'information est reçue par les dendrites, traitée par le soma, puis transmise par l'axone. De même, un neurone artificiel reçoit des entrées pondérées, chaque entrée multipliée individuellement par un poids w, Le corps du neurone artificiel effectue ensuite la somme des entrées pondérées, ajoute un biais b, puis traite cette somme à l'aide d'une fonction d'activation. Enfin, transmet le résultat en sortie[72].

2.3.2 Couches de Réseaux Neuronaux

Un réseau de neurones artificiels (ANN) est formé par l'interconnexion de plusieurs neurones artificiels [72]. Ses paramètres essentiels sont les poids w et les biais b, qui influencent le fonctionnement des neurones. Comme illustré à la figure 2-2, un ANN est structuré en couches : une couche d'entrée (input layers), une ou plusieurs couches cachées (hidden layers), et une couche de sortie (output layers) [73]. La couche d'entrée de chaque neurone pointe vers des caractéristiques d'entrée qui sont transmises aux couches cachées est introduite pour apprendre des relations complexes. Lorsque le réseau comporte plusieurs couches cachées, on parle d'apprentissage profond (DNN).

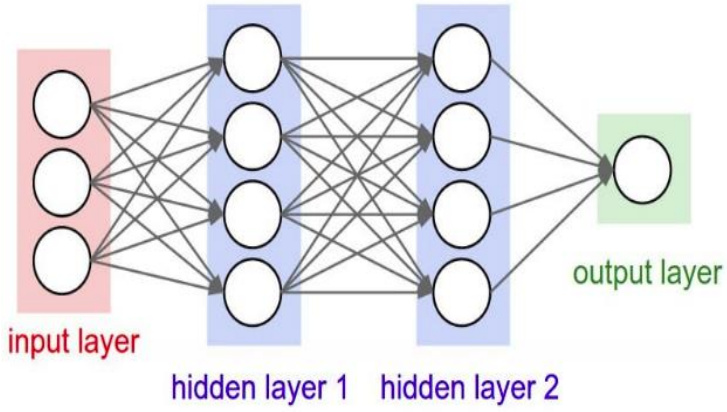

Figure 2-2: Les couches de réseaux neuronaux [74].

2.3.3 Fonctions d'Activation

Les fonctions d'activation sont essentielles car elles permettent aux réseaux de neurones de modéliser des relations complexes dans des données variées. En leur absence, le réseau se limite à une fonction linéaire incapable d'apprendre des structures sophistiquées telles que images, vidéos, audio ou texte [75]. Le tableau 2-1 présente les principales fonctions d'activation, leurs caractéristiques et applications.

Tableau 2-1: Les Fonctions d'activation [75].

Utilisé En Couche	Fonction d'activation	Formule	Caractéristiques
cachée/ sortie	Sigmoïde	$f(x) = \frac{1}{1+e^{-x}}$ Plage de sortie [0,1]	-Symétrie non centrée (toutes les sorties ont des signes identiques). -Limité par l'écrasement du gradient pour des valeurs très grandes ou petites de x.
Cachée	Tanh	$f(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$ Plage de sortie [-1,1]	-Les gradients sont plus raides et centrés autour de zéro.
Cachée	ReLU	$f(x) = max(0, x)$ Plage de sortie [0, *x*]	-Très efficace car la majorité des neurones sont inactifs. - Problèmes de "décès de neurones" si le gradient devient nul.
Sortie	Softmax	$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$ Plage de sortie [0, 1]	-Utilisée pour la classification multi-classes, où chaque sortie est une probabilité.

2.4 Entraînement des Réseaux de Neurones

Le processus d'entraînement d'un réseau de neurones est un processus itératif au cours duquel les calculs sont effectués dans le sens aller (forward) et retour (backward) à travers chaque couche du réseau jusqu'à ce que la fonction de perte soit minimisée. La figure 2-3 illustre que le processus d'entraînement peut être divisé en trois parties principales :

- **Forward Propagation :** consiste à faire circuler les données d'entrée à travers les différentes couches du réseau depuis la couche d'entrée jusqu'à la couche de sortie dans une seule direction[76].
- **La fonction de perte :** Pour évaluer les performances du modèle, une fonction de perte est utilisée. Elle mesure l'erreur entre l'étiquette réelle y_i et la sortie prédite $\hat{y}_i$. Parmi

les fonctions de perte couramment utilisées, la cross-entropy est souvent employée pour les tâches de classification et se définit comme suit :

$$\text{Loss}_{XE} = -\sum_{i=1}^{n-1} y_i \log(\hat{y}_i) \tag{2.1}$$

- **Backpropagation :** est un processus d'ajustement des paramètres du réseau visant à minimiser l'erreur entre la sortie prédite et la valeur réelle. Cette optimisation utilise la fonction de perte et la descente de gradient en rétropropagant l'erreur de la couche de sortie vers les couches précédentes [77]. La description mathématique de l'algorithme de Backpropagation peut être trouvée dans [78].

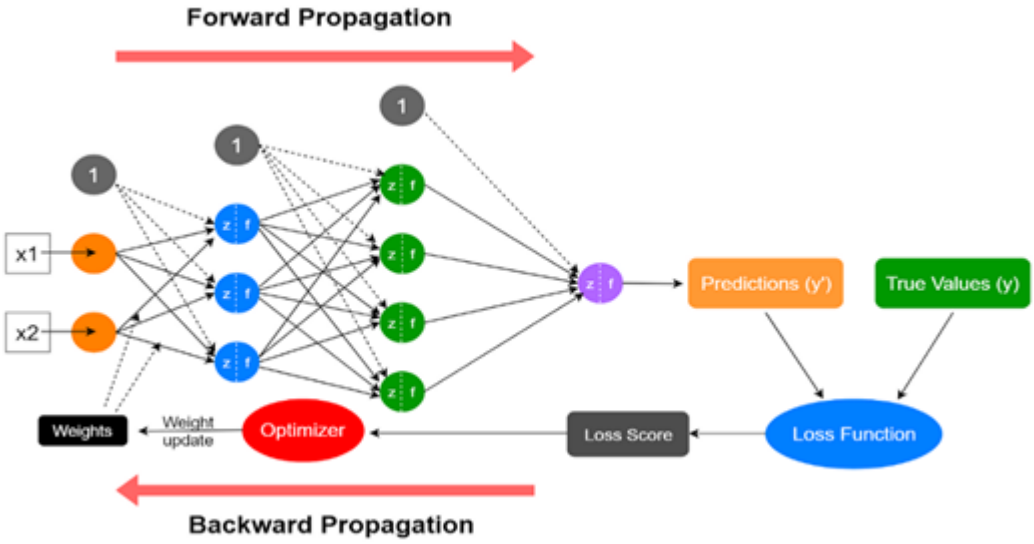

Figure 2-3: Processus d'entraînement d'un réseau de neurones [79].

2.5 Régularisation du Réseau de Neurones

Les modèles de réseaux neuronaux (NN) sont bien adaptés aux domaines où de grands ensembles de données étiquetées sont disponibles, car leur capacité peut être facilement augmentée en ajoutant plus de couches ou plus d'unités dans chaque couche. Cependant, les grands réseaux avec des millions ou des milliards de paramètres peuvent facilement overfit [80].

Par ailleurs, L'entraînement des NN est compliqué par le fait que la distribution des entrées de chaque couche change pendant l'entraînement, à mesure que les paramètres des couches

précédentes sont modifiés. Cela ralentit l'apprentissage, car il faut utiliser de taux d'apprentissage plus faibles et d'une initialisation des paramètres plus rigoureuse [81].

Pour remédier à ces problèmes, plusieurs techniques de régularisation ont été proposées. Celles-ci visent à améliorer la précision du modèle, à renforcer sa capacité de généralisation sur des données non vues, et à accélérer la convergence lors de l'entraînement. Dans ce qui suit, Deux techniques spécifiquement conçues pour les NN sont présentées .

2.5.1 Dropout

Le Dropout est une technique de régularisation utilisée pour prévenir le overfitting dans les réseaux de neurones. Elle consiste à désactiver aléatoirement certaines unités du réseau pendant l'entraînement ainsi que leurs connexions ce qui empêche le modèle de devenir trop dépendant de certaines caractéristiques comme illustré dans la figure 2-4. Chaque unité est conservée avec une probabilité fixe, souvent choisie autour de 0,5, sauf pour les unités d'entrée où elle est généralement plus élevée [82]. Cela rend le réseau plus robuste et améliore sa capacité de généralisation.

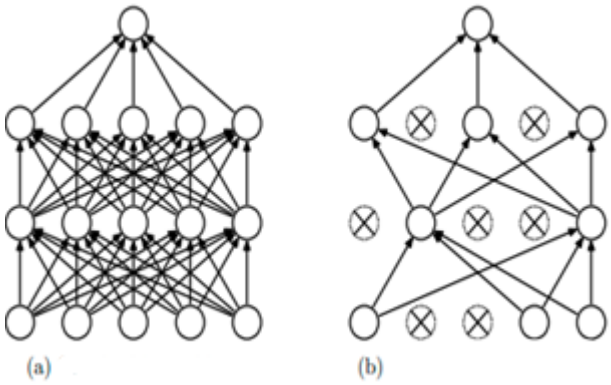

Figure 2-4: mécanisme de Dropout dans ANN, (a) Réseau standard, (b) Réseau avec neurones désactivés [82].

2.5.2 Batch Normalization

La Batch Normalization évite l'explosion des activations en corrigeant de manière répétée toutes les activations pour qu'elles aient une moyenne nulle et un écart-type unitaire. Cela permet de faire des pas de gradient plus grands, conduit à une convergence plus rapide, et peut aider à éviter les minima locaux trop abrupts. Avec cette sorte de "précaution de sécurité", il devient possible d'entraîner des réseaux avec des taux d'apprentissage élevés, car

les activations ne peuvent pas croître de manière incontrôlée étant donné que leurs moyennes et variances sont normalisées [83]. Dans certains cas, cela peut même éliminer le besoin d'utiliser le Dropout [81].

2.6 Métriques d'Évaluation des Performances

Après l'entraînement d'un réseau de neurones, il est indispensable d'évaluer ses performances à l'aide de métriques appropriées. Ces métriques d'évaluation permettent de quantifier la qualité des prédictions réalisées par le modèle. De nombreuses métriques ont été proposées dans la littérature, chacune mettant en évidence un aspect particulier de la performance. Le choix des métriques dépend du type de problème traité[84]. Dans les problèmes de classification, les métriques dérivées de la matrice de confusion sont parmi les plus couramment utilisées. Elles fournissent des informations précieuses sur la capacité du modèle à distinguer correctement les différentes classes, et permettent une analyse comparative entre différents algorithmes.

2.6.1 Matrice de Confusion

La matrice de confusion est un outil d'analyse prédictive en apprentissage automatique, elle est utilisée pour évaluer les performances d'un modèle d'apprentissage supervisé basé sur la classification. En effet, une matrice de confusion est une matrice de taille $n * n$, où n représente le nombre de classes cibles, et elle permet ainsi d'évaluer la performance d'un modèle de classification. Dans cette matrice, d'une part, les colonnes représentent les valeurs prédites, et d'autre part, les lignes représentent les valeurs réelles (ou observées) du modèle [85]. Le tableau 2-2 illustre une représentation de la matrice de confusion.

Tableau 2-2: La matrice de confusion.

Prédite / Réelle	Oui	Non	Total
Oui	TP	FN	P
Non	FP	TN	N
Total	P'	N'	P+N

La matrice de confusion divise les échantillons de test en quatre catégories, en fonction de leurs valeurs réelles et prédites :

- Vrais positifs (TP) : échantillons pour lesquels les valeurs réelle et prédite sont toutes les deux égales à 1.
  Exemple : le patient a un cancer (1) et le modèle classe cet échantillon comme cancéreux (1).
- Vrais négatifs (TN) : échantillons pour lesquels les valeurs réelle et prédite sont toutes les deux égales à 0.
  Exemple : le patient n'a pas de cancer (0) et le modèle classe cet échantillon comme non cancéreux (0).
- Faux positifs (FP) : échantillons pour lesquels la valeur réelle est 0 et la valeur prédite est 1.
  Exemple : le patient n'a pas de cancer (0) et le modèle classe cet échantillon comme cancéreux (1).
- Faux négatifs (FN**)** : échantillons pour lesquels la valeur réelle est 1 et la valeur prédite est 0.
  Exemple : le patient a un cancer (1) et le modèle classe cet échantillon comme non cancéreux (0) [86].

À partir de cette matrice, on définit deux valeurs fondamentales : P, représentant le nombre total de cas positifs réels (soit la somme de TP et FN), et N, représentant le nombre total de cas négatifs réels (somme de TN et FP). De même, P' désigne le nombre total de cas que le modèle a prédits comme positifs (TP + FP), tandis que N' désigne les cas prédits comme négatifs (TN + FN).

2.6.2 Métrique d'Évaluation

Il existe plusieurs métriques d'évaluation qui peuvent être déduites à partir de la matrice de confusion. Le tableau ci-dessous présente quelques-unes des métriques les plus couramment utilisées :

Tableau 2-3: Métriques d'évaluation pour la classification [87].

Métrique	Formule
Accuracy	$\frac{TP+TN}{P+N}$
Error Rate	$\frac{FP+FN}{P+N}$
Sensitivity (Recall)	$\frac{TP}{P}$
Specificity	$\frac{TN}{N}$
Precesion	$\frac{TP}{TP+FP}$
F1-score	$2\times\frac{\text{precesion}\times\text{recall}}{\text{precesion}+\text{recall}}$

2.7 Les Réseaux de Neurones Convolutifs

Les réseaux de neurones convolutifs (CNN) sont une catégorie des réseaux de neurones artificiels appartenant au DDN, Ils sont principalement utilisés pour le traitement et l'analyse des données visuelles, telles que les images et les vidéos. Ces réseaux exploitent des opérations de convolution pour extraire automatiquement des caractéristiques spatiales des données d'entrée[88]. L'architecture de base d'un CNN est une combinaison de différentes couches telles que la couche d'entrée, la couche de convolution, la couche de pooling, la couche Fully Connected et la couche de sortie, comme illustré à la figure 2-5.

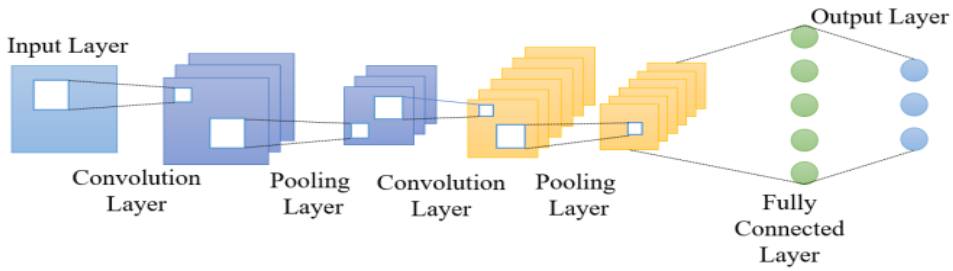

Figure 2-5: Architecture générale d'un CNN [89].

- **Couche de Convolution :** Applique des filtres (kernels) aux données d'entrée pour extraire des caractéristiques importantes, telles que les contours et les motifs, grâce à des opérations de multiplication et de l'addition (voir figure 2-6) [90].

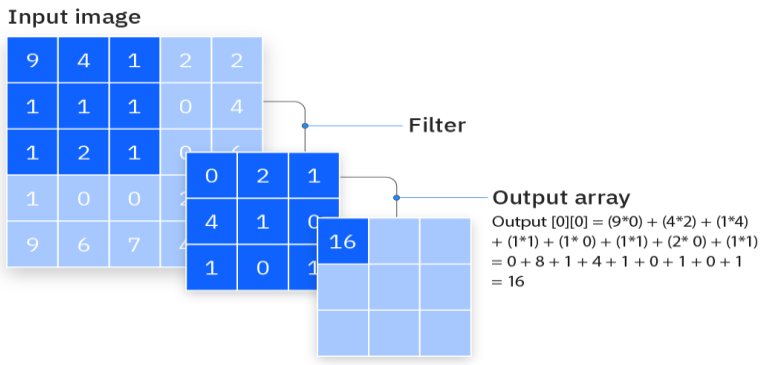

Figure 2-6: Illustration de l'opération de convolution avec un filtre 3x3 [91].

- **Couche d'activation :** Une fonction non linéaire comme ReLU est appliquée aux caractéristiques extraites de la couche de convolution, ce qui introduit de la non-linéarité dans le modèle et améliore sa capacité à apprendre des relations complexes [92].
- **Couche de Pooling :** Cette couche réduit la dimension des cartes de caractéristiques extraites, ce qui permet de diminuer le nombre de calculs et de rendre le modèle plus robuste aux variations mineures des données [93] comme illustré dans la figure 2-7.

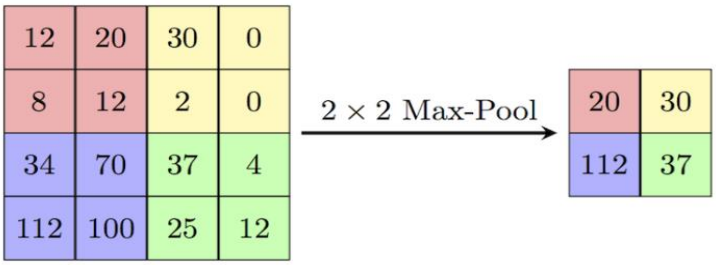

Figure 2-7: Application du Max-Pooling avec une fenêtre 2×2 [94].

- **Couche Fully Connected :** Elle connecte tous les neurones entre eux et est utilisée pour prendre les décisions finales en fonction des caractéristiques extraites par les couches précédentes [95].
- **Couche de sortie** : le nombre de neurones dans cette couche est défini en fonction des besoins. En cas de classification, ce nombre correspond généralement au nombre de catégories à classer [89]. La fonction Softmax pour la classification multiclasse est utilisée juste avant la couche de sortie afin d'attribuer une probabilité décimale à chaque classe. Ces probabilités sont comprises entre 0 et 1.

2.8 Transformer

Le Transformer est un type d'architecture de DDN qui a eu un impact considérable sur le NLP et les tâches d'apprentissage automatique après sa première introduction dans l'article "Attention is All You Need" en 2017 [96]. Il est reconnu comme un composant essentiel de nombreux modèles SOTA en NLP (expliqué en section 1.10) et a été efficacement utilisé dans plusieurs tâches de vision par machine, telles que la classification d'images [97], ainsi que dans des tâches de traitement audio comme la détection d'événements sonores [98] et l'audio captioning [99].

Contrairement aux architectures récurrentes (RNN, LSTM, GRU) qui traitent les données de façon séquentielle, le Transformer utilise un mécanisme d'attention qui lui permet de traiter toute la séquence en parallèle, offrant ainsi une meilleure efficacité et une plus grande capacité d'apprentissage des relations à longue distance dans les données.

2.8.1 Les modèles de langage

À partir l'architecture de Transformer, plusieurs modèles de langage ont été développées, chacune avec ses spécificités. Le tableau 2-4 présente les modèles les plus connus :

Tableau 2-4: Les modèles de langage BERT, GPT et BART [100] [51] [57].

Modèles / Caractéristiques	BERT	GPT	BART
Organisation	Google AI	OpenAI	Facebook AI
Type de modèle	Encodeur	Décodeur	Encodeur – Décodeur
Sens de lecture	Bidirectionnel	Unidirectionnel (de gauche à droite)	Bidirectionnel
Couches	12 couches d'encodeur	12 couches décodeur	12 couches encodeur + 12 décodeur
Paramètres	Plus de 100 millions	117 millions (GPT-1), 1.5 billion (GPT-2), 175 billion (GPT-3),	139 millions

2.8.2 Architecture du Transformer

Le modèle Transformer, introduit dans le modèle original de Vaswani et al. (2017), repose sur une architecture encodeur-décodeur (voir figure 2-8), chacun composé d'une pile de N = 6 couches identiques.

L'encodeur traite la séquence d'entrée à travers deux sous-couches : une mécanisme multi-head self-attention, qui permet à chaque token de prêter attention à tous les autres tokens de la séquence, et un réseau de neurones entièrement connecté appliqué position par position (MLP), qui affine les représentations apprises. Chaque sous-couche est entourée d'une connexion résiduelle, suivie d'une normalisation de couche (layer normalization), ce qui donne une sortie de la forme LayerNorm(x + Sublayer(x)), où Sublayer(x) représente la transformation appliquée [96].

Le décodeur suit une structure similaire à celle de l'encodeur, mais ajoute une troisième sous-couche, appelée Masked Self-Attention. La représentation d'entrée et toutes les couches mentionnées précédemment seront expliquées en détail dans ce qui suit.

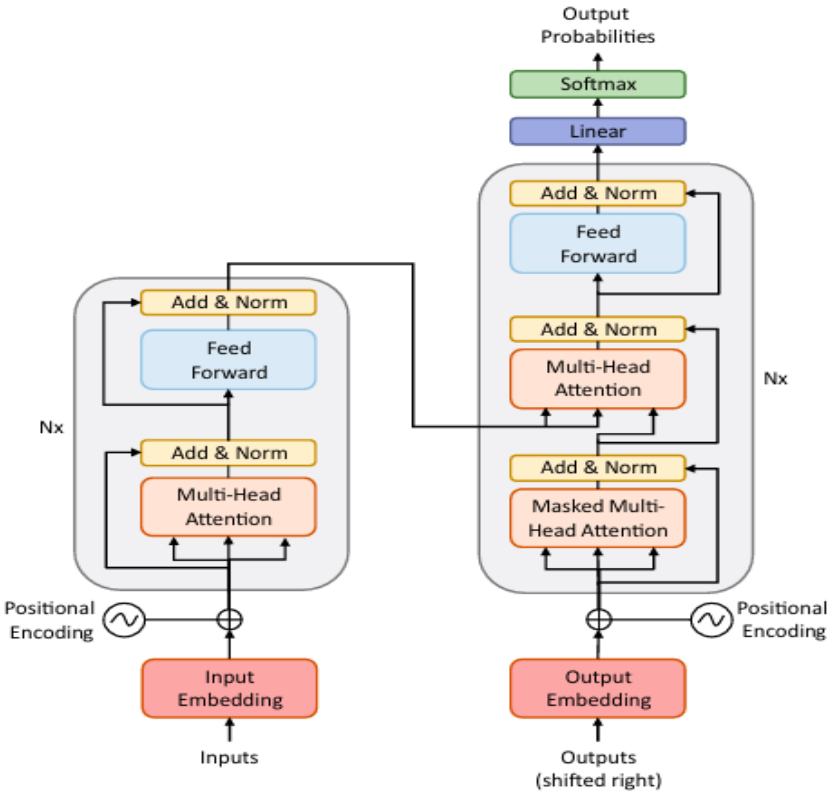

Figure 2-8: L'architecture du modèle Transformer [96].

2.8.2.1 Représentation d'Entrée

Le texte d'entrée est d'abord divisé en tokens (expliqué en section 1.8.1), qui peuvent être des mots ou des sous-mots. Ensuite, chaque token est converti en un vecteur numérique appelé embedding, qui capture son sens sémantique (expliqué en section 1.9.2). Pour conserver l'ordre des mots dans la séquence, un encodage de position est attribué à chaque token. Enfin, les embeddings des tokens et leurs encodages de position sont additionnés pour obtenir l'embedding final utilisé par le modèle [101]. Un exemple illustrant la représentation d'entrée de modèle BERT est présenté dans la figure 2-9.

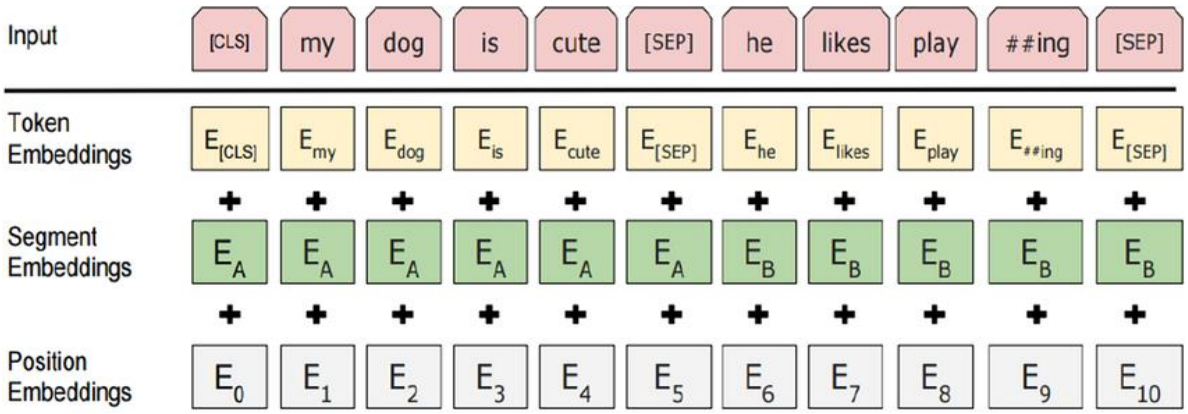

Figure 2-9: La représentation d'entrée de BERT [102].

2.8.2.2 Mécanisme d'Attention

Le Mécanisme d'Attention est une technique qui permet d'attribuer une importance variable à différentes parties de l'entrée lors de la génération de la sortie. Ce mécanisme permet au modèle de se concentrer sur les informations les plus pertinentes et d'ignorer celles qui sont moins importantes, ce qui améliore la performance globale du modèle.

Dans l'AAC, l'attention aide à identifier les segments les plus significatifs d'un signal audio pour générer une description textuelle pertinente. Par exemple, un modèle chargé de générer une caption pour un enregistrement contenant divers sons d'une rue animée, tels que des klaxons, des voix humaines et une ambulance au loin, attribuera un poids plus élevé aux segments contenant la sirène, un élément distinctif, tandis que les voix humaines, étant plus courantes, recevront un poids moindre. Ce processus permet ainsi de générer une caption fidèle qui met en avant les sons les plus caractéristiques.

La fonction d'attention transforme une requête (Q) ainsi qu'un ensemble de paires clé-valeur (K, V) en une sortie, où tous ces éléments sont représentés par des vecteurs [103]. Pour mieux comprendre ce mécanisme, on peut l'assimiler à une recherche sur le Web :

- **Requête (Q)** : C'est le texte de recherche que vous saisissez dans la barre d'un moteur de recherche.
- **Clé (K) :** Ce sont les titres de chaque page web dans la fenêtre des résultats de recherche.
- **Valeur (V) :** Ce sont les contenus réels des pages web affichées.

Une fois que le moteur de recherche a fait correspondre le terme de recherche approprié (requête) avec les résultats pertinents (clés), il souhaite obtenir le contenu (valeur) des pages les plus pertinentes.

2.8.2.3 Self-Attention

La self-attention est un mécanisme d'attention particulier qui permet à un modèle d'évaluer les relations entre les différentes positions d'une même séquence. Elle vise à intégrer le contexte global lors du traitement de chaque élément, en pondérant l'importance relative des autres éléments de la séquence [53].

Le score d'attention est calculé par le produit scalaire entre la requête (Q) et la transposée de la matrice des clés K, suivi d'une normalisation avec Softmax pour obtenir des scores de probabilité. Ces scores servent ensuite à pondérer les valeurs V. La formule de self-attention est donc la suivante :

$$\text{Attention}(Q,K,V) = \text{Softmax}\, Q(K^{T})V \tag{2.2}$$

2.8.2.4 Scaled Dot-Product Attention

La scaled Dot-Product Attention est une amélioration de self-attention. Ce mécanisme est formulé mathématiquement comme suit :

$$\text{Attention}(Q,K,V) = \text{Softmax}\left(\frac{QK^{T}}{\sqrt{d_k}}\right)V \tag{2.3}$$

Ici, Le produit scalaire entre Q et la clé K est divisé par $\sqrt{d_k}$ ($d_k$est la dimension des clés) afin d'éviter d'obtenir des valeurs trop grandes. En effet, lorsque d_k est grand, les produits scalaires peuvent atteindre des valeurs élevées, poussant ainsi la fonction softmax dans des régions où les gradients deviennent extrêmement faibles. Pour contrer cet effet, on normalise les produits scalaires en les divisant par $\sqrt{d_k}$, ce qui stabilise l'apprentissage et améliore les performances du modèle [96].

2.8.2.5 Multi-Head Attention

La Multi-Head Attention est une extension de self-attention, permettant au modèle de capturer plusieurs types de relations contextuelles en parallèle. Au lieu d'utiliser une seule fonction d'attention, les Q, K et V sont projetées h fois. Sur chacune de ces versions projetées la fonction d'attention est appliquée en parallèle. Ensuite, les résultats obtenus sont concaténés et projetés pour produire la sortie finale comme illustré dans la figure 2-10.

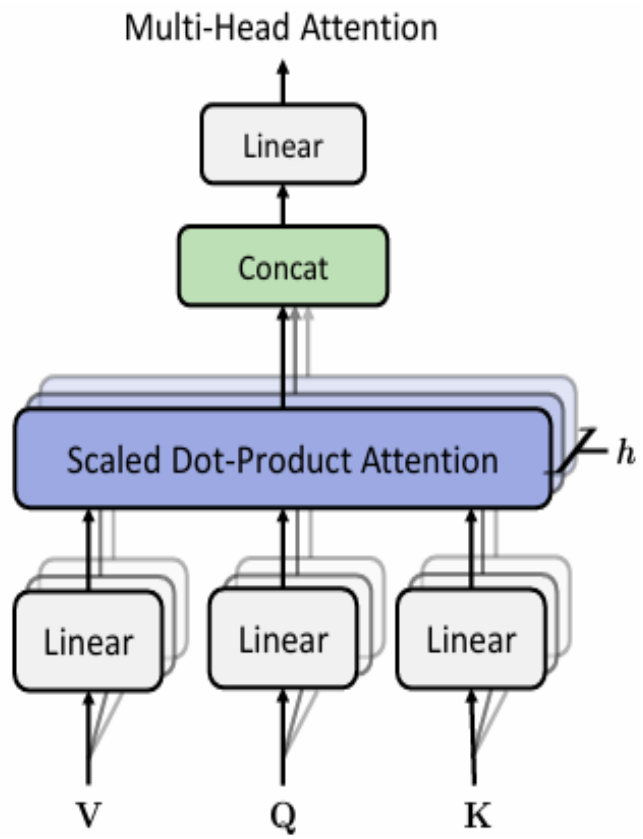

Figure 2-10: **Multi-Head Attention se compose de plusieurs couches d'attention exécutées en parallèle [96].**

2.8.2.6 Masked Self-Attention

Ce mécanisme permet au modèle de générer des séquences en se concentrant sur les parties pertinentes de l'entrée tout en empêchant l'accès aux tokens futurs. On commence par calculer le score d'attention. Il est déterminé par le produit scalaire entre les matrices Q et K, ce qui permet de calculer la relation entre chaque token d'entrée. Ensuite, un masque est appliqué à la matrice d'attention pour empêcher l'accès aux tokens futurs. Après cela, les scores d'attention sont convertis en probabilités grâce à la fonction softmax, ce qui permet de déterminer la pertinence relative de chaque token par rapport aux autres.

2.8.2.7 Feed-Forward

Après le mécanisme de Multi-Head Attention, chaque token passe par un perceptron multicouche (MLP), qui est un réseau de neurones Feed-Forward (FFN) appliquée indépendamment à chaque token. Alors que la couche d'attention a pour objectif de faire circuler l'information entre les tokens en modélisant leurs dépendances, la FFN affine individuellement la représentation de chaque token afin de capturer des interactions complexes et d'améliorer l'expressivité globale du modèle [101]. Mathématiquement, cette couche est composée de deux transformations linéaires séparées par une fonction d'activation non linéaire ReLU :

$$FFN(x) = \max(0 , xW1 + b1)W2 + b2 \quad (2.4)$$

Où W1 et W2 sont des matrices de poids apprises, et b1, b2 des biais.

2.9 Modèles Séquence- à-Séquence

Les modèles Séquence-à-Séquence (Seq2Seq) sont un type de réseau de neurones développés pour traiter une séquence de données en vue de générer une séquence de sortie. En termes simples, l'entrée et la sortie sont toutes deux des séquences. Les modèles Seq2Seq ont été appliqués à un très large éventail d'applications de traitement du langage naturel, telles que la traduction automatique, la reconnaissance vocale et l'AAC.

L'architecture de base des modèles Seq2Seq comprend deux composants principaux : l'encodeur et le décodeur. Le principe fondamental de cette approche repose sur l'utilisation d'un encodeur qui prend une séquence en entrée et en produit une représentation contextualisée, souvent appelée contexte. Cette représentation est ensuite transmise à un décodeur qui génère une séquence de sortie adaptée à la tâche [104](voir figure 2-11).

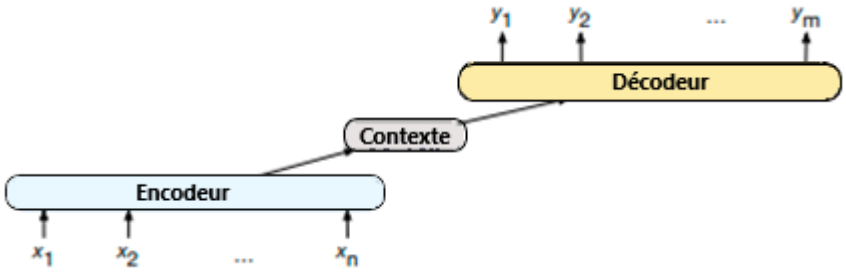

Figure 2-11: Architecture générale d'un modèle séquence-à-séquence [104].

Dans la tâche d'AAC, les caractéristiques acoustiques extraites du signal audio, telles que les spectrogrammes, sont utilisées comme entrée de l'encodeur afin d'être converties en une représentation contextuelle. Cette représentation est ensuite exploitée par le décodeur pour générer une caption. La figure 2-12 illustre cette architecture.

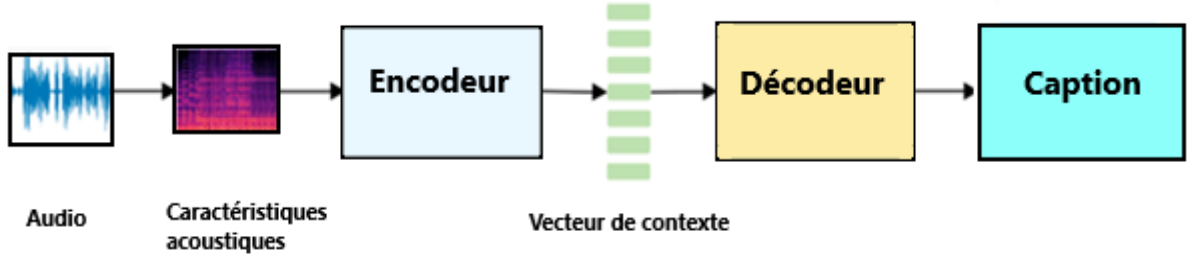

Figure 2-12: L'architecture générale d'un système AAC.

2.9.1 Encodeur

L'analyse du contenu d'un clip audio repose en grande partie sur l'obtention d'une représentation efficace des caractéristiques, ce qui est l'objectif de l'encodeur dans un système AAC [6]. Les formes d'onde dans le domaine temporel sont de longs signaux 1D et il est difficile pour les machines d'identifier directement des événements sonores ou des scènes sonores à partir de ces formes d'onde brutes[105]. Il est donc nécessaire de transformer ces

signaux audio en représentations plus compactes et informatives telles que le spectrogramme afin de faciliter la reconnaissance des événements présents dans le clip.

Dans l'encodeur, ces représentations spectrales sont traitées et converties en représentations vectorielles de longueur fixe, appelées vecteurs de contexte**.** Ces vecteurs capturent les informations pertinentes du signal d'entrée et servent de base à la génération de séquences textuelles en sortie. L'encodeur peut être implémenté à l'aide de réseaux de neurones profonds, tels que CNN, efficaces pour l'extraction des caractéristiques spatiales, ou des architectures récurrentes comme les RNN qui permettent de modéliser la dynamique temporelle du signal, ainsi que les modèles Transformes qui exploitent des mécanismes d'attention pour une meilleure modélisation des relations à long terme dans l'audio.

2.9.1.1 Architectures d'Encodeur

Plusieurs architectures d'encodeur ont été développées, les RNN, CNN et les Transformer étant parmi les plus utilisées, chacune présentant des forces et des limitations spécifiques.

Les RNN sont conçus pour traiter des données séquentielles de longueur variable[106]. L'audio est un signal temporel ; par conséquent, les RNN ont été adoptés comme encodeurs dans les premiers travaux [5]. Les encodeurs basés sur GRU ou LSTM capturent les relations temporelles dans les signaux audio, les états cachés servant de caractéristiques audio extraites. Les principaux avantages de l'utilisation des RNN comme encodeurs sont leur simplicité et leur capacité à traiter des données séquentielles. Cependant, il a été constaté que l'utilisation des RNN seuls comme encodeurs ne permet pas d'obtenir de bonnes performances [107]. Cela s'explique par leur difficulté à capturer les dépendances à long terme dans les séquences étendues. De plus, la compression des états cachés en une unique représentation globale de l'audio peut entraîner une perte d'informations essentielles, compliquant ainsi la génération de descriptions précises et détaillées par le décodeur linguistique.

Face à ces limitations, les CNN ont émergé comme une alternative plus efficace, démontrant des performances exceptionnelles dans les tâches liées à l'audio. Kun Chen et al. [108] ont utilisé un CNN comme encodeur audio et ont considérablement amélioré les

performances des systèmes de génération de caption audio par rapport aux modèles basés sur les RNN [109]. D'une part, parmi les modèles CNN utilisées, on retrouve notamment les réseaux de type VGG et ResNet, ils permettent d'extraire des caractéristiques audio pertinentes et offrent de bonnes performances sur les tâches de traitement du son. D'autre part, Les principaux avantages des CNN sont leur invariance aux décalages temporels et leur capacité à modéliser les dépendances locales au sein des spectrogrammes. Toutefois, ils présentent aussi des limites, notamment des champs réceptifs restreints, cela signifie que chaque neurone dans une couche convolutive ne peut capturer que des informations locales sur une petite partie du spectrogramme audio. Cela limite la capacité du modèle à capturer des relations à long terme dans le signal.

Récemment, les Transformes ont suscité un intérêt croissant en AAC, offrant des performances compétitives dans l'apprentissage de représentations audio robustes, comparables aux encodeurs CNN**.** Contrairement aux RNN et CNN, les Transformers exploitent des mécanismes de self-attention (comme expliqué en section 2.7), permettant une modélisation plus efficace des dépendances globales dans les séquences audio. Toutefois, leur principal désavantage réside dans leur besoin accru en données d'entraînement**,** rendant leur implémentation plus exigeante que celle des CNN [6].

Ainsi, l'évolution des modèles d'encodeurs en AAC est passée des RNN, limités dans le traitement des longues séquences, aux CNN, plus efficaces pour capturer les caractéristiques audio locales, jusqu'aux Transformer, qui offrent une modélisation plus fine des dépendances à long terme mais nécessitent davantage de ressources pour atteindre leur plein potentiel.

2.9.2 Décodeur

L'objectif du décodeur est de générer une caption à partir des vecteurs de contexte fournies par l'encodeur. En d'autres termes, Le décodeur prend en entrée la sortie de l'encodeur et génère la séquence de sortie finale[6]. Cette approche permet au modèle de mieux capturer les relations sémantiques entre les mots et d'améliorer la cohérence des captions. Différentes architectures peuvent être utilisées pour le décodeur, notamment les RNN (LSTM ou GRU) ainsi que les Transformers.

2.9.2.1 Architectures de Décodeur

Il existe plusieurs architectures de décodeur, parmi lesquelles les RNN et les Transformer occupent une place centrale, chacun avec ses forces et ses limitations.

Les phrases sont également des données séquentielles composées de mots discrets. Ainsi, les RNN sont couramment utilisés comme décodeurs de langage. De nombreux travaux ultérieurs ont adopté des RNN monocouches, soit des réseaux GRU, soit des réseaux LSTM [110][111][112] . Cependant, Le principal inconvénient des RNN est qu'ils peuvent avoir du mal à capturer les dépendances à longue portée (long-range) entre les mots générés. Bien que cette limitation soit moins problématique pour AAC, où les légendes sont généralement courtes, elle reste un facteur contraignant.

Avec l'introduction du Transformer par Vaswani et al. [96] en 2017, le mécanisme self-attention, qui est au cœur du Transformer, est rapidement devenu le bloc de construction de base des grands modèles de langage. Les modèles basés sur Transformer, tels que GPT [51] et BART [58] surpassent les RNN dans la modélisation du langage et dominent la plupart des tâches en NLP. Donc, Les décodeurs basés sur Transformer offrent des performances à l'état de l'art en AAC et sont plus efficaces sur le plan computationnel que les décodeurs basés sur RNN lors de l'entraînement[6]. Cette évolution souligne l'intérêt croissant pour les architectures basées sur l'attention, qui s'affranchissent des limitations des RNN en matière de capture des dépendances à longue portée, mais qui peuvent présenter un coût computationnel plus élevé.

2.10 Travaux Connexes

Au cours de notre recherche, nous avons étudié plusieurs architectures proposées pour la détection et la classification des scènes et événements acoustiques, dans la tâche AAC. Parmi les approches les plus courantes, l'architecture séquence à séquence, fondée sur une structure encodeur–décodeur, est largement utilisée. Dans ce schéma, l'encodeur extrait les caractéristiques audio pertinentes, tandis que le décodeur génère une description textuelle correspondante. Ces architectures ont démontré de bonnes performances pour la tâche d'AAC, en capturant efficacement la relation entre les signaux audio et le langage naturel. Parmi les modèles étudiés pour résoudre le problème du captioning audio, trois rapports

présentent chacun une approche spécifique pour l'extraction des caractéristiques audio et la génération de descriptions textuelles.

Article 1: Cet article [113] repose sur une architecture séquence à séquence conçue pour la tâche AAC. L'entrée audio est d'abord transformée en spectrogramme log-Mel, puis un réseau CNN 1D pour l'extraction des embeddings audio. Cette représentation est ensuite traitée par un encodeur MLP-Mixer, un réseau composé uniquement de perceptrons multicouches (MLP) organisés pour mélanger successivement les dimensions temporelles et fréquentielles du signal. Contrairement aux réseaux à attention, le MLP-Mixer utilise uniquement des blocs linéaires, accompagnés de normalisation et de connexions résiduelles, pour capturer les dépendances dans les deux dimensions. Le décodeur est un Transformer classique basé sur le modèle de Vaswani et al., 2017, qui génère les captions audio mot par mot à partir des embeddings.

Article 2 : L'architecture présentée dans cet article [114] repose sur un modèle de type encodeur-décodeur destiné à la génération automatique de descriptions textuelles à partir de données audio. L'encodeur est basé sur une version adaptée du réseau de neurones convolutionnel profond ResNet-18. Ce réseau est constitué de plusieurs blocs convolutifs dotés de connexions résiduelles, ce qui facilite l'apprentissage de représentations complexes tout en atténuant le problème de la dégradation du gradient dans les réseaux profonds. Dans cette architecture, ResNet-18 est utilisé pour extraire les caractéristiques audio à partir de spectrogrammes log-Mel. Ces représentations sont ensuite transformées par une couche linéaire avant d'être transmises au décodeur. Ce dernier, composé d'une couche GRU suivie d'une couche linéaire, génère les séquences textuelles mot par mot. Cette architecture hybride permet une modélisation efficace du contenu sonore en vue de produire des descriptions textuelles cohérentes.

Article 3 : L'architecture proposée dans cet article [115] repose sur un modèle encodeur–décodeur de type séquence-à-séquence**.** L'encodeur est constitué d'un empilement de blocs de convolutions neuronales (CNNs), combinant des convolutions depthwise separable suivies de convolutions 2D classiques, de couches d'activation (Leaky ReLU, ReLU), de normalisation (BatchNorm), de pooling et de Dropout. Ces CNNs permettent d'extraire les caractéristiques audio à partir de spectrogrammes log-Mel. Le décodeur est celui du Transformer, composé de

blocs de multi-head attention et de couches Feed-Forward, permettant de générer des descriptions textuelles.

Le tableau 2-5 illustre une analyse comparative des trois études mentionnées ci-dessus, chacune utilisant des méthodes différentes pour améliorer les performances du système audio captioning.

Tableau 2-5 : Analyse comparative des trois rapports.

Rapport	Encodeur	Décodeur	caractéristiques audio	Dataset	Fonction de perte	Optimiseur
Article 1	MLP-Mixer	Transformer	spectrogramme log-Mel, CNN 1D pour embedding	Clotho v2.1	Cross-entropy	Adam
Article 2	ResNet-18	GRU	spectrogramme log-Mel	Clotho v2.1	Cross-entropy	Adam
Article 3	CNN	Transformer	spectrogramme log-Mel	Clotho	Cross-entropy	Adam

2.11 Défis de l'AAC

De nombreuses méthodes basées sur l'apprentissage profond ont été proposées pour améliorer les systèmes d'AAC, et cette tâche a connu des progrès rapides ces dernières années. Cependant, il existe toujours un écart important entre les performances de ces systèmes et le niveau de performance humaine. Parmi les principaux défis, on peut citer :

La rareté des données : Les datsets disponibles pour l'AAC sont encore limités en taille et en diversité. La collecte de telles données est coûteuse en temps, et il est difficile de garantir la qualité des captions humaines.

La capacité de généralisation : Les datasets existantes ne couvrent pas l'ensemble des situations et environnements réels, ce qui limite la capacité des modèles à s'adapter à des contextes inconnus.

La diversité et la stylisation des captions : Comme discuté dans [116], un bon modèle de génération de captions devrait produire des phrases possédant trois propriétés :

- Fidélité : Les captions générées doivent refléter fidèlement le contenu audio.
- Naturalité : Les captions ne doivent pas être identifiables comme étant générées par une machine.
- Diversité : Les phrases doivent être riches et variées, reflétant la manière dont différentes personnes décriraient un même clip audio de façons différentes.

Cependant, la plupart des approches actuelles ne prennent en compte que la fidélité[6].

La description détaillée : Les systèmes actuels [117], [118] tendent à ne capturer que les événements sonores principaux, en négligeant des détails importants tels que les propriétés des événements ou les relations temporelles entre eux. Pourtant, ces détails sont essentiels pour produire des légendes vraiment informatives [119].

2.12 Conclusion

Dans ce chapitre, nous avons abordé les concepts fondamentaux d'apprentissage automatique et des réseaux de neurones, en mettant l'accent sur les principes clés, les méthodes d'entraînement et les techniques de régularisation nécessaires à la construction de modèles efficaces. Nous avons également présenté un aperçu des principales architectures de réseaux de neurones, telles que les CNN et les Transformers, et nous avons souligné l'importance du modèle Séquence-à-Séquence dans le domaine de l'AAC, ainsi que les architectures encodeur-décodeur, chacune avec ses forces et ses limitations. Ensuite, nous avons présenté les travaux connexes à l'AAC. Enfin, nous avons discuté des défis spécifiques associés à l'AAC, en mettant en évidence la complexité de cette tâche et la nécessité de solutions innovantes pour surmonter ces obstacles. Ces éléments constituent les bases nécessaires pour approfondir la compréhension des sujets avancés abordés dans les chapitres suivants.

PARTIE II : EXPÉRIMENTATIONS

Dans cette partie, nous présentons la méthodologie adoptée pour la conception, la mise en œuvre et l'évaluation de notre système d'AAC. Elle est structurée en deux chapitres. Le premier chapitre décrit l'approche expérimentale mise en place ainsi que l'architecture du système proposé. Le second chapitre est consacré à l'analyse des résultats obtenus à l'issue des expérimentations, accompagnée d'une interprétation des performances observées.

Chapitre 3 : Approche Expérimentale

3.1 Introduction

L'AAC est une tâche de traduction multimodale dans laquelle un système reçoit un signal audio en entrée et l'analyse afin de générer une caption en langage naturel en sortie. Ce chapitre présente notre approche proposée pour l'AAC, utilisée afin de mener nos expériences. Nous présentons tout d'abord un aperçu global de l'architecture proposée dans la section 3.2. Chaque étape de notre modèle est ensuite détaillée : la segmentation audio en section 3.3, l'extraction des embeddings audio à l'aide des techniques et modèles décrits dans les sections 3.4 et 3.5, suivie de leur projection présentée dans la section 3.6, la préparation du texte en section 3.7, et enfin l'implémentation du modèle basé sur l'architecture Transformer en section 3.8.

3.2 Architecture du Modèle d'AAC Proposé

L'architecture proposée de notre modèle repose sur une structure séquence-à-séquence, basée sur un Transformer BART, pour la génération de captions à partir de signaux audio. La figure 3-1 illustre le fonctionnement global du modèle, qui se décompose en deux étapes principales : l'extraction des caractéristiques et la modélisation.

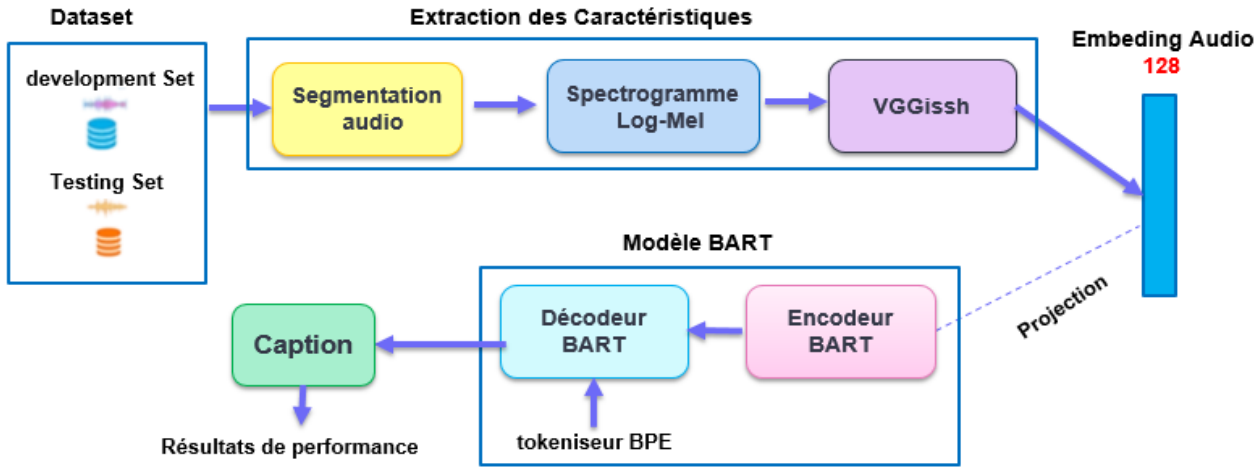

Figure 3-1: Schémas généraux illustrant les étapes principales de la mise en œuvre de notre modèle.

Lors de l'extraction des caractéristiques, les signaux audio issus des ensembles de développement et de test sont d'abord segmentés, puis convertis en spectrogrammes Log-Mel. Ces représentations sont ensuite traitées par un modèle préentraîné, VGGish, afin d'extraire des embeddings audio de dimension 128, qui correspondent à des vecteurs compacts capturant les caractéristiques pertinentes du signal audio. Ces embeddings sont ensuite projetés dans un espace vectoriel de dimension appropriée afin d'être intégrés dans notre modèle BART.

La phase de modélisation repose sur l'utilisation du modèle BART. L'encodeur prend en entrée les embeddings transformés et produit une séquence d'embeddings de même dimension que la séquence d'entrée. Le décodeur génère ensuite la caption de manière auto-régressive, les mots générés précédemment sont d'abord tokenisés à l'aide d'une méthode de type Byte-Pair Encoding (BPE)**,** puis transformés en vecteurs d'entrée. Le décodeur applique des mécanismes d'attention à la fois sur ces tokens et sur les sorties de l'encodeur. La sortie finale est produite par une tête de classification avec une fonction d'activation softmax, qui prédit la probabilité de chaque token du vocabulaire, permettant ainsi la génération mot par mot de la caption audio**.** Les résultats de performance sont évalués à partir de cette sortie.

3.3 Segmentation Audio

La première étape du prétraitement consiste à segmenter les fichiers audio en extraits de 1 seconde, comme illustré à la figure 3-2**.** Chaque audio du dataset Clotho, qui dure entre 15 et 30 secondes, est divisé en segments fixes de 1 seconde sans recouvrement. Ce choix de durée permet d'obtenir une granularité temporelle adéquate pour capturer des événements sonores courts tout en conservant une structure temporelle gérable pour le modèle. Cette segmentation facilite également le traitement par batchs et l'extraction systématique de caractéristiques audio homogènes. En divisant les fichiers audio en trames d'une seconde, on obtient une série de représentations temporelles normalisées, prêtes à être transformées en représentations spectrales log-Mel, puis en embeddings VGGish**.** Cette approche garantit que chaque segment audio contient suffisamment d'informations acoustiques pour permettre au modèle d'extraire des éléments pertinents en vue de générer des descriptions sémantiques précises.

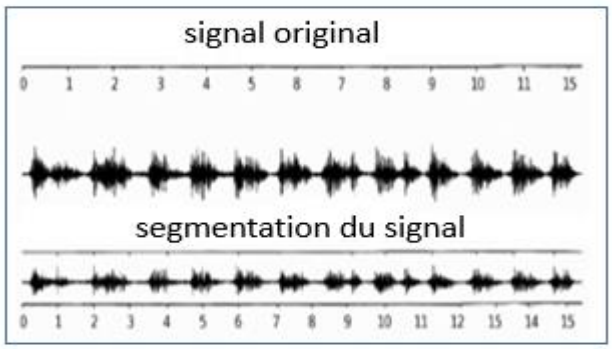

Figure 3-2: Segmentation d'un signal audio de 15 secondes en segments d'une seconde.

3.4 Spectrogramme Log-Mel

Les modèles d'apprentissage profond ne peuvent pas directement utiliser un signal audio brut comme entrée, car l'audio est une série temporelle complexe et bruyante qui contient beaucoup d'informations redondantes. Pour rendre l'audio plus exploitable, il est nécessaire de le transformer en une représentation plus compacte et pertinente. C'est pourquoi nous utilisons l'extraction du spectrogramme log-Mel, qui permet de convertir l'audio en une forme plus adaptée à notre modèle.

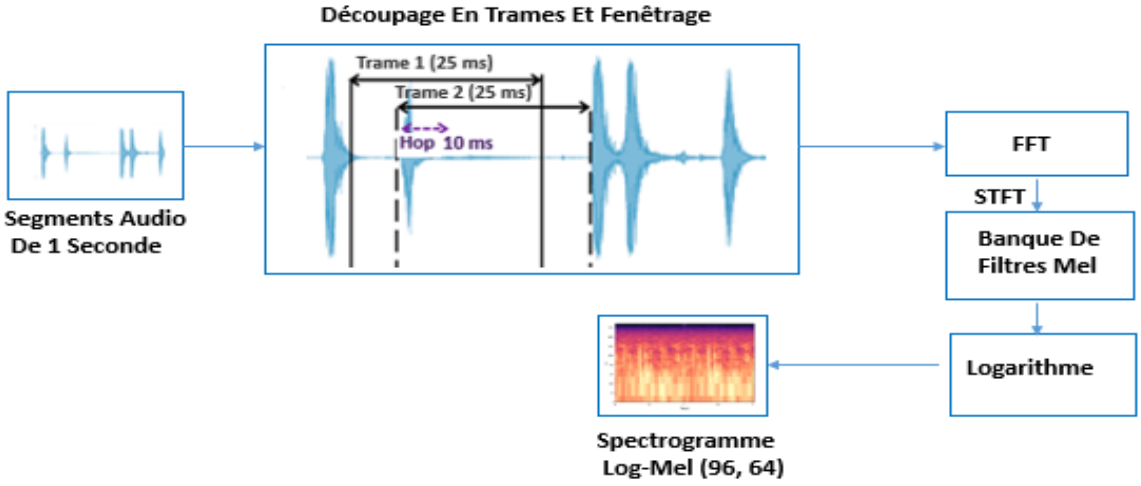

Figure 3-3: Le processus global d'extraction du spectrogramme log-Mel.

L'extraction du spectrogramme log-Mel, illustrée dans la figure 3-3, commence par le découpage du signal audio, échantillonné (SR) à 16 kHz, en trames de 25 ms, avec une taille de hop de 10 ms entre chaque trame. Chaque trame est ensuite multipliée par une fenêtre de Hann périodique pour réduire les effets de bord. Une transformée de Fourier à court terme

(STFT) est appliquée sur chaque trame à l'aide d'une FFT de taille 512, produisant ainsi un spectre de magnitudes. Ce spectre est ensuite converti en échelle Mel, calculée pour 64 bandes Mel réparties entre 125 Hz et 7500 Hz. Enfin, on applique une fonction logarithmique sur les valeurs Mel pour obtenir le spectrogramme log-Mel final. Ce spectrogramme contient des caractéristiques audio compactées et robustes, prêtes à être utilisées par des modèles d'apprentissage profond comme VGGish. Pour un segment audio de 1 seconde, cela donne un tenseur de forme (96, 64), représentant 96 trames de 64 bandes Mel.

3.5 Extraction des Embeddings Audio avec VGGish

L'utilisation directe de spectrogrammes log-Mel comme représentations d'entrée dans l'encodeur Transformer peut s'avérer insuffisante pour capturer les caractéristiques abstraites et sémantiques des signaux audio. Pour remédier à cela, des modèles préentraînés tels que VGGish sont employés afin d'extraire des embeddings audio plus riches, encapsulant des informations de plus haut niveau. Ces embeddings facilitent ainsi une meilleure performance des modèles de génération de descriptions audio. VGGish est un réseau de neurones convolutif développé par Google, basé sur l'architecture VGG (Visual Geometry Group) [120] et pré-entraîné sur le dataset AudioSet [121], une collection de 2 millions de clips audio de dix secondes, extraits de vidéos YouTube et répartis sur plus de 600 classes sonores.

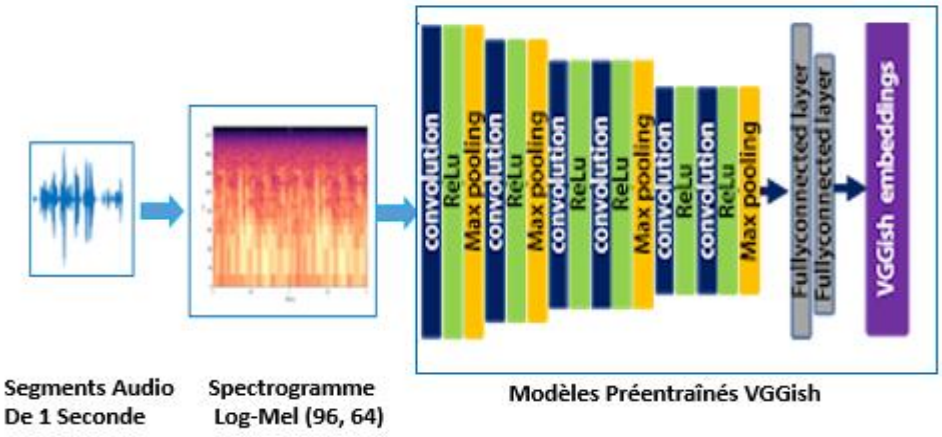

Figure 3-4: Extraction des embeddings audio à l'aide du modèle pré-entraîné VGGish.

Comme illustré dans la figure 3-4**,** VGGish prend en entrée un Spectrogramme Log-Mel de taille 96×64, calculé à partir de segments audio de 1 seconde. Ce spectrogramme est ensuite traité par le réseau VGGish à travers une série de 4 blocs convolutifs, chacun composé d'une ou deux couches convolutionnelles avec des noyaux 3×3, suivies d'une activation ReLU et d'un

max pooling. Ces blocs permettent une extraction hiérarchique des caractéristiques spectro-temporelles du signal audio. La dernière couche de max-pooling est suivie de deux couches entièrement connectées comportant chacune 4096 unités suivies d'une activation ReLU, puis d'une couche entièrement connectée de 128 unités avec activation ReLU qui génère le vecteur d'embedding de 128 dimensions. Dans notre tâche, nous avons désactivé le post-traitement d'origine du modèle, basé sur une réduction PCA suivie d'une quantification, et supprimé la dernière couche ReLU afin de conserver des représentations continues plus fidèles. Le tableau 3-1 fournit une description détaillée de l'architecture du modèle VGGish employé dans cette étude.

Tableau 3-1 : Architecture détaillée du réseau VGGish, Les sorties au format (B, C, H, W) où B : la taille du batch = 4, C : nombre de canaux, H : la hauteur et W : largeur.

Couches		Sortie
Entrée		(B, 1, 96, 64)
Bloc 1	Conv2d (1, 64, kernel size=3, padding=1) ReLU MaxPool2d (kernel_size=2, stride=2)	(B, 64, 96, 64) (B, 64, 48, 32)
Bloc 2	Conv2d (64, 128, kernel_size=3, padding=1) ReLU MaxPool2d (kernel_size=2, stride=2)	(B, 128, 48, 32) (B, 128, 24, 16)
Bloc 3	Conv2d (128, 256, kernel_size=3, padding=1) ReLU Conv2d (256, 256, kernel_size=3, padding=1) ReLU MaxPool2d(kernel_size=2, stride=2)	(B, 256, 24, 16) (B, 256, 24, 16) (B, 256, 12, 8)
Bloc 4	Conv2d (256, 512, kernel_size=3, padding=1) ReLU Conv2d (512, 512, kernel_size=3, padding=1) MaxPool2d (kernel_size=2, stride=2)	(B, 512, 12, 8) (B, 512, 12, 8) (B, 512, 6, 4)
Flatten		(B, 12288)
FC1	Linear (12288, 4096) ReLU	(B, 4096)
FC2	Linear (4096, 4096) ReLU	(B, 4096)
Embedding	Linear (4096 , 128) ReLU	(B, 128)

3.6 Projection des Embeddings

Afin de rendre les embeddings audio du modèle VGGish de dimension 128 compatibles avec le modèle Transformer BART, une couche affine est utilisée pour projeter ces vecteurs dans un espace de dimension appropriée : 768 pour un encodeur BART à 4,6 couches, ou 1024 pour un encodeur à 12 couches. Une couche affine est une transformation linéaire suivie d'un biais, qui permet de modifier la dimension des vecteurs d'entrée tout en apprenant à ajuster la représentation à la tâche cible. Cette opération facilite l'adaptation de la représentation audio aux exigences du Transformer, tout en capturant de manière plus riche les relations complexes présentes dans les données.

Ces embeddings transformés sont ensuite stockés dans des fichiers au format (npy), accompagnés du nom du fichier audio d'origine et de chacune des cinq captions textuelles associées. Ces captions sont préalablement nettoyées en supprimant la ponctuation et en les convertissant en minuscules, assurant un format uniforme. Cette structuration garantit un alignement précis entre les représentations audio et textuelles, facilitant ainsi l'entraînement efficace de notre modèle, dans lequel les embeddings audio sont utilisés comme entrées conditionnelles et les captions nettoyées servent de sorties cibles.

3.7 Tokenisation BPE

La tokenisation constitue une étape essentielle dans le prétraitement des données textuelles pour les modèles de génération de texte. Dans notre système, nous utilisons le modèle BART, un transformater séquence-à-séquence comportant un encodeur et un décodeur. Ce dernier génère les captions mot par mot à l'aide d'un tokenizer nommé facebook/bart spécialement conçu pour l'architecture du modèle BART, développé par Facebook AI. Ce tokenizer a été entraîné sur un corpus varié de contenus, incluant des articles de presse, des livres et des sites web, et dispose d'un vocabulaire de 50 265 tokens.

Ce tokenizer repose sur l'algorithme Byte Pair Encoding (BPE), qui consiste à fusionner les paires de caractères ou de sous-mots les plus fréquentes dans un corpus afin de former progressivement un vocabulaire optimisé, comme expliqué brièvement dans la Section 2.3.2 du Chapitre 2. Cette méthode permet de gérer efficacement les mots rares ou inconnus, appelés Out-Of-Vocabulary (OOV), ce qui est particulièrement important puisque le dataset

Clotho v2 contient environ 4 500 mots uniques répartis sur 34 860 captions. Grâce à la segmentation des mots en sous-unités fréquentes, BPE réduit les problèmes liés aux OOV. De plus, BPE améliore la capacité de généralisation du modèle en lui permettant de mieux comprendre et générer des mots composés ou nouveaux à partir de sous-tokens connus, ce qui est essentiel dans notre tâche où la diversité lexicale et contextuelle est importante. Pour entraîner le tokenizer BPE dans notre tâche AAC, le processus suit plusieurs étapes :

Prétraitement des données : Cette étape comprend la suppression de la ponctuation, la normalisation des espaces et la mise en minuscule de tous les caractères dans les captions du dataset. Deux tokens spéciaux sont ensuite ajoutés à chaque caption : <sos> pour indiquer le début de la phrase et <eos> pour marquer la fin de la phrase. Par exemple, une caption comme 'A dog is barking loudly' devient après prétraitement : <sos> a dog is barking loudly <eos>.

Construction du vocabulaire : Tous les mots uniques présents dans les caption du split d'entraînement sont ensuite collectés pour construire un vocabulaire initial, par exemple : {'a', 'dog', 'is', 'barking', 'running', 'man', 'shouting'}.

Application de l'algorithme BPE : Les mots du vocabulaire initial sont d'abord décomposés en caractères. L'algorithme BPE identifie ensuite les paires de caractères les plus fréquentes et les fusionne progressivement. Par exemple, le mot 'barking' est représenté initialement par {'b', 'a', 'r',' k',' i', 'n', 'g'} puis les paires fréquentes comme {'i n', 'n g'} sont fusionnées pour former {'ing'}, donnant ainsi {'b', 'ark', 'ing'}. De même, "running" peut être segmenté en {'r', 'unn', 'ing'}. Cette méthode permet de constituer un vocabulaire basé sur des sous-unités fréquentes plutôt que sur des mots entiers.

Entraînement du modèle : Une fois le tokenizer entraîné, il peut encoder efficacement de nouvelles phrases, même si certains mots ne figurent pas dans le vocabulaire initial. Par exemple, la phrase 'a bird is chirping' sera transformée en une séquence de tokens numériques : <sos>, 101, 234, 56, 876, <eos>. Cette approche rend le modèle plus robuste aux variations lexicales, tout en réduisant la taille du vocabulaire à gérer.

3.8 Transformer BART

Le Transformer de type BART est utilisé dans notre tâche pour améliorer la génération de captions audio en combinant un encodeur bidirectionnel et un décodeur auto-régressif comme illustré dans la figure 1-18 du chapitre 1. L'encodeur bidirectionnel est responsable de l'encodage des embedding audio extraites, à l'aide du modèle VGGish pré-entraîné, en une représentation contextuelle riche. Cette architecture permet à l'encodeur de traiter les informations audio de manière bidirectionnelle, c'est-à-dire en prenant en compte à la fois le contexte passé et futur des données audio. Cela lui permet de capturer efficacement les relations temporelles et les dépendances à long terme dans les séquences audio, ce qui est essentiel pour bien comprendre le contenu d'un signal audio complexe. Le décodeur, quant à lui, utilise cette représentation contextuelle pour générer les captions de manière auto-régressive de gauche à droite, c'est-à-dire mot par mot, en se basant sur les tokens précédemment générés et les informations extraites de l'encodeur. Dans notre travail, nous expérimentons avec n = 4, n = 6 et n = 12 couches pour l'encodeur et le décodeur tel que :

Tableau 3-2: Configurations du modèle BART.

Modèle	Couches (Enc/Dec)	Head Attention	Dimension FFN	Dimension des embeddings
BART (n = 4)	4 / 4	12	3072	768
BART Base (n = 6)	6 / 6	12	3072	768
BART Large (n =12)	12 / 12	16	4096	1024

La figure 3-5 illustre l'architecture du modèle Transformer BART. Pour plus de clarté, les connexions résiduelles ainsi que les opérations de normalisation de couche ne sont pas représentées. L'architecture détaillée a été présentée dans la figure 2-8, et une brève explication de chaque couche est fournie dans la section 2.7.2 du chapitre 2.

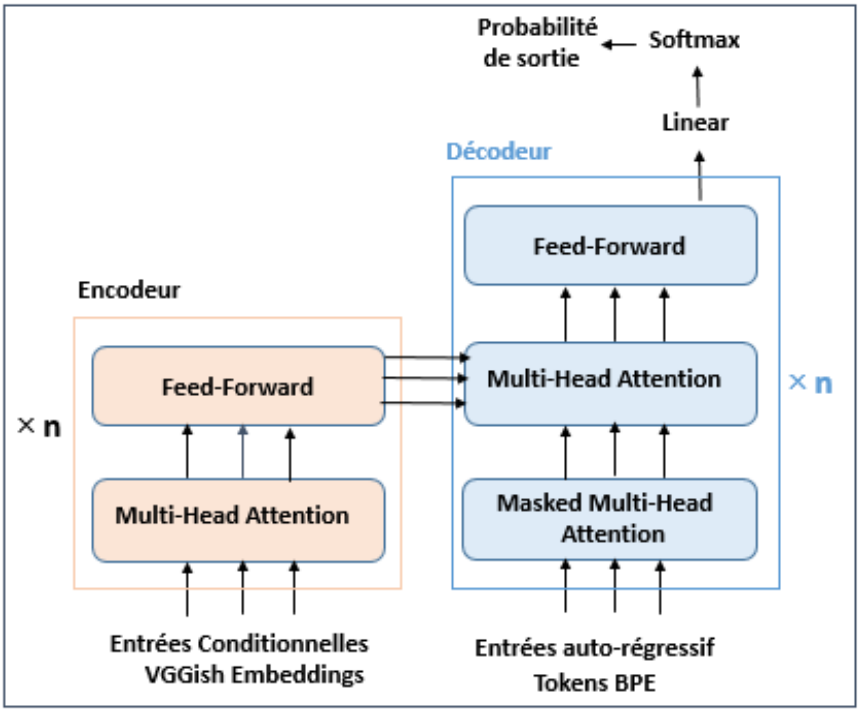

Figure 3-5: Architecture du transformer BART (connexions résiduelles et normalisation de couche omises pour plus de clarté) et configuration proposée dans AAC.

L'encodeur reçoit en entrée des vecteurs d'embedding produits par le modèle VGGish pré-entraîné. Chaque bloc de l'encodeur se compose de deux couches principales :

Une couche Multi-Head Attention qui permet au modèle de se concentrer sur différentes parties de la séquence audio en parallèle, en capturant des relations temporelles complexes entre les vecteurs d'entrée.

Un réseau de neurones Feed-Forward (FFN) qui applique une transformation non linéaire à chaque position de la séquence, afin d'enrichir la représentation des caractéristiques audio.

Chaque couche de l'encodeur produit un vecteur de caractéristiques pour chaque étape temporelle de la séquence d'entrée. L'encodeur génère ainsi une séquence de vecteurs audio encodés correspondant à la taille des embeddings utilisés dans ce modèle transformer, comme indiqué dans le tableau 3-2, prête à être utilisée par le décodeur pour produire une description textuelle.

Le décodeur auto-régressif du transformer exploite à la fois les sorties de l'encodeur et les embeddings des mots, qui ont été préalablement tokenisés à l'aide de la méthode BPE.

Chaque token issu de cette tokenisation est converti en un vecteur dense, ce qui permet de représenter les mots sous une forme compréhensible par le modèle. Ces vecteurs d'embedding sont ensuite transmis au décodeur. Chaque bloc décodeur contient :

Une couche Masked Multi-Head Attention qui permet au modèle de ne considérer que les tokens précédemment générés dans la séquence, garantissant ainsi la nature auto-régressive du décodeur et empêchant l'accès à des informations futures lors de l'entraînement ou de l'inférence.

Un bloc Multi-Head Attention avec mécanisme d'attention croisée qui interagit avec les représentations issues de l'encodeur. Cette sous-couche permet au décodeur d'intégrer le contexte global de la séquence d'entrée encodée.

Un réseau de neurones Feed-Forward appliqué indépendamment à chaque position de la séquence, qui introduit des transformations non linéaires supplémentaires afin d'augmenter la capacité expressive du modèle.

Chaque couche du décodeur produit une représentation vectorielle correspondant à la taille des embeddings utilisés dans ce modèle transformer, comme indiqué dans le tableau 3-2. Enfin, Une couche finale composée d'une fonction d'activation softmax, produit des scores de probabilité pour chaque élément du vocabulaire. La sortie finale est déterminée à l'aide d'une technique de décodage, soit le beam Search avec une taille K = 4, soit le greedy decoding.

Le beam Search consiste à générer plusieurs séquences candidates en explorant simultanément différents chemins de génération. À chaque étape, le modèle conserve les K séquences les plus probables, ce qui permet, à la fin, de sélectionner la séquence ayant la probabilité globale la plus élevée.

En revanche, **le greedy decoding** sélectionne à chaque étape uniquement le token ayant la probabilité la plus élevée, sans explorer d'autres alternatives. Bien qu'il soit plus rapide, ce mode de décodage est souvent moins optimal et peut entraîner des séquences moins diversifiées ou plus répétitives.

Cette différence entre les deux approches est illustrée dans l'expérience 4 du chapitre 4, où nous comparons les performances de chaque méthode sur le data set d'évaluation.

3.9 Conclusion

Dans ce chapitre, nous avons décrit notre modèle proposé utilisé pour mener nos investigations expérimentales, en commençant par le prétraitement audio. Nous avons présenté le spectrogramme Log-Mel pour l'extraction des caractéristiques, le modèle VGGish pour l'extraction des embeddings audio, ainsi que le modèle BART basé sur l'architecture Transformer, en mettant en évidence les étapes clés de la conduite de nos expériences. Dans le chapitre suivant, nous présenterons les résultats de quatre expériences et les analyserons afin d'en dégager des recommandations pour le développement de systèmes de génération de légendes audio plus performants.

Chapitre 4 : Résultats Expérimentaux et Discussion

4.1 Introduction

Ce chapitre présente la configuration expérimentale utilisée pour mener nos expériences. Tout d'abord, la section 4.2 présente les outils de développement et les environnements utilisés. Dans la section 4.3, nous présentons de manière détaillée le dataset utilisé pour l'entraînement de notre système. La section 4.4 expose les métriques d'évaluation utilisées. Ensuite, la section 4.5 est consacrée aux hyperparamètres choisis pour nos expériences. La section 4.6 évalue et analyse les résultats obtenus lors de nos expérimentations sur le système d'audio captioning. Enfin, nous illustrons les résultats obtenus durant la phase de test à l'aide d'une interface web que nous avons développée.

4.2 Outils et Environnements de Développement

Pour implémenter notre système, nous avons choisi **Kaggle**, une plateforme en ligne développée par Google, qui offre un environnement complet pour l'expérimentation en science des données et en apprentissage automatique. Elle propose notamment une variété de ressources telles que des datasets dans divers formats (CSV, JSON, SQLite, HDF5), des compétitions, des cours et des notebooks collaboratifs. Kaggle met à disposition des notebooks Jupyter intégrés, permettant d'écrire, d'exécuter et de partager du code Python depuis le navigateur. Ces notebooks peuvent gérer jusqu'à 100 Go de données d'entrée par projet. En revanche, le répertoire de sortie est limité à 5 Go par défaut, bien qu'il soit possible d'optimiser cet espace en compressant les données, ce qui permet de sauvegarder jusqu'à 20 Go de résultats [122]. De plus, la plateforme offre des GPU gratuits, notamment des Nvidia Tesla P100 avec 16 Go de mémoire, qui peuvent être activés à tout moment pour une durée maximale de 30 heures par semaine [123], permettant ainsi d'effectuer des calculs intensifs nécessaires à l'entraînement de modèles d'apprentissage profond.

4.2.1 Bibliothèques

Nous avons réalisé nos expériences en utilisant **Python**, un langage interactif et polyvalent, grâce à son interaction directe avec les notebooks Jupyter [124] et à ses nombreuses bibliothèques. Nous avons utilisé la version 3.7, accompagnée de bibliothèques adaptées au traitement des données et au traitement du langage naturel, telles que :

Tableau 4-1: Bibliothèques utilisées dans la tâche d'AAC.

Bibliothéque	Fonction principale
PyTorch	Créer des modèles d'apprentissage profond, basée sur les tenseurs et optimisée pour le GPU via CUDA [125].
Librosa	Simplifie la manipulation des fichiers audio [126], nous l'utilisons pour le prétraitement et l'extraction de caractéristiques.
Transformers	Pour le traitement du langage naturel, développée par Hugging Face [127]. Nous l'utilisons pour charger et entraîner notre modèle BART afin de générer des captions.
Numpy	Stocker les embeddings audio et les captions dans des tableaux structurés et effectuer rapidement des opérations mathématiques.
Pathlib	Gérer facilement les chemins de fichiers audio de manière orientée objet.
Tqdm	Affiche une barre de progression pour suivre en temps réel l'avancement du traitement des fichiers audio et annotations.
Gradio	Permet de créer facilement des interfaces web interactives pour tester et démontrer notre modèle AAC.

4.3 Dataset

Nos expériences ont été réalisées sur le dataset Clotho v2.1, qui comprend 6974 échantillons audio avec des métadonnées associées à chaque échantillon, ainsi que 34 870 captions. Les échantillons audio, d'une durée comprise entre 15 et 30 secondes, sont de bonne qualité (44,1 kHz, 16 bits) et couvrent une large gamme de sons du quotidien (nature, animaux, foules, machines, véhicules, etc.). Tous les clips audio proviennent de la plateforme en ligne Freesound [128]. Chaque échantillon audio est accompagné de cinq caption contenant de 8 à

20 mots, comme illustré dans la figure 4-1. Ces captions ont été collectées via Amazon Mechanical Turk (AMT), en suivant un protocole spécifique de crowdsourcing pour l'annotation audio, conçu pour garantir la diversité des descriptions et minimiser les erreurs grammaticales [129]. Lors de l'annotation seul le signal audio était fourni aux annotateurs, sans information supplémentaire telle que des mots-clés ou des indices vidéo [130], afin d'éviter l'introduction de biais.

```
File Name: Distorted AM Radio noise.wav
Caption 1: A muddled noise of broken channel of the TV
Caption 2: A television blares the rhythm of a static TV.
Caption 3: Loud television static dips in and out of focus
Caption 4: The loud buzz of static constantly changes pitch and volume.
Caption 5: heavy static and the beginnings of a signal on a transistor radio
```

Figure 4-1 : Exemple de cinq captions pour un fichier audio de bruit radio AM

Clotho v2.1 comprend deux sections principales : un development set et un test set. Le development set est composé de trois sous-ensembles: 3839 clips audio pour l'entraînement, et 1045 clips audio pour la validation et l'évaluation, respectivement. Le test set contient quant à lui 1043 clips audio. [6]. Nous avons téléchargé les fichiers audio pour les deux sections, compressés au format .7z et contenant des fichiers .wav, ainsi que les fichiers de captions au format .csv pour le development set. Le dataset inclut également des fichiers de métadonnées au format .csv pour les deux sections. Ces métadonnées comprennent : le nom du fichier audio, les mots-clés associés, l'URL de l'audio original sur la plateforme Freesound, les échantillons de début et de fin pour l'extrait utilisé, l'utilisateur de Freesound qui a téléchargé le fichier ainsi que le lien vers la licence d'utilisation. La figure 4-2 illustre un exemple du fichier de métadonnées pour le development set.

```
file_name: Distorted AM Radio noise.wav
keywords: noise;radio
sound_id: 117790
sound_link: https://freesound.org/people/lavatorius/sounds/117790
start_end_samples: [139264, 1293089]
manufacturer: lavatorius
license: http://creativecommons.org/publicdomain/zero/1.0/
```

Figure 4-2: un exemple du fichier clotho-metadata-development.csv.

4.4 Métriques d'Évaluation

L'évaluation automatique de la qualité des captions générées constitue une étape essentielle dans le développement des systèmes d'AAC. Plusieurs métriques ont été proposées pour mesurer à la fois la fidélité au contenu de référence et la qualité linguistique des captions produites. Ces métriques reposent sur différentes approches, allant de la comparaison des séquences de mots à l'analyse sémantique, afin d'offrir des évaluations complémentaires et robustes. Le tableau 4-2 présente les principales métriques utilisées, chacune apportant une perspective spécifique sur la qualité des captions.

Tableau 4-2: Les métriques utilisées dans notre modèle.

Métrique	Principe	Plage de score
BLEU	Précision des n-grammes entre captions générées et références	[0, 1]
METEOR	Combine la précision, le rappel, le stemming et la synonymie.	[0, 1]
ROUGE-L	Longueur de la plus longue sous-séquence commune (LCS)	[0, 1]
CIDEr	Pondération TF-IDF des n-grammes et similarité cosinus.	[0, 10]
SPICE	Évaluation sémantique basée sur les objets, attributs et relations.	[0, 1]
SPIDEr	Moyenne des scores CIDEr et SPICE.	[0, 1]

4.4.1 BLEU (Bilingual Evaluation Understudy)

La métrique BLEU mesure la qualité des captions en comparant les n-grammes des captions générées à celles de référence. Elle calcule une moyenne géométrique pondérée des précisions des n-grammes qui correspondent aux captions de référence. Le score obtenu varie entre 0 et 1 [131]. Selon la taille des n-grammes considérés, on distingue plusieurs variantes : BLEU-1 (unigrammes), BLEU-2 (bigrammes), et ainsi de suite, offrant un niveau d'évaluation de plus en plus précis de la qualité linguistique.

4.4.2 METEOR (Metric for Evaluation of Translation with Explicit ORdering)

La métrique METEOR évalue la qualité des captions générées en prenant en compte la précision, le rappel, le stemming et la synonymie. Elle établit des correspondances entre les mots de la caption générée et ceux de la caption de référence selon plusieurs critères, puis calcule une F-mesure basée sur ces alignements pour quantifier la qualité globale [132].

4.4.3 ROUGE-L (Recall-Oriented Understudy for Gisting Evaluation-Longest Common Subsequence)

La métrique ROUGE-L permet d'évaluer la similarité entre une caption générée et une caption de référence, en s'appuyant sur la longueur de la plus longue sous-séquence commune (LCS) entre les deux phrases. Elle calcule une F-mesure fondée sur la précision, correspondant à la proportion de la LCS présente dans la caption générée, et sur le rappel, correspondant à la proportion de la LCS présente dans la caption de référence [133]. Ce score reflète à la fois la qualité lexicale et la cohérence de l'ordre des mots, tout en tolérant les omissions ou réarrangements mineurs. Le score ROUGE-L varie entre 0 et 1, ce qui en fait une métrique bien adaptée à l'évaluation automatique des captions, où l'ordre des mots et la couverture du contenu sont essentiels.

4.4.4 CIDEr (Consensus-based Image Description Evaluation)

La métrique CIDEr évalue la qualité des captions en s'appuyant sur une pondération TF-IDF des n-grammes, ce qui permet de mesurer la similarité entre une caption générée et un ensemble de captions de référence à l'aide du cosinus de similarité [134]**.** Cette approche est particulièrement utile pour l'analyse des captions lorsqu'un ensemble de plusieurs captions de référence est disponible.

4.4.5 SPICE (Semantic Propositional Image Caption Evaluation)

SPICE évalue les captions sur la base de leur contenu sémantique. Elle détermine si les captions contiennent des composants linguistiques spécifiques tels que des objets, des

attributs et des relations [135]. Cela permet de mieux refléter la capacité du modèle à capturer le sens global de la scène décrite.

4.4.6 SPIDEr (Semantic Propositional Image Description Evaluation with Consensus)

SPIDEr combine CIDEr et SPICE en calculant une moyenne de leurs scores respectifs [136]. Elle vise ainsi à évaluer simultanément la similitude lexicale via CIDEr et la fidélité sémantique via SPICE des captions générées, offrant une évaluation plus équilibrée et représentative de leur qualité globale.

4.5 Hyperparamètres des Expériences

La configuration utilisée pour l'entraînement de notre modèle, présentée dans le tableau 4-3, comprend hyperparamètres soigneusement choisis pour optimiser l'apprentissage.

Tableau 4-3: Hyperparamètres d'entraînement pour toutes les expériences.

Hyperparamètre	**Valeur**
Taille de batch	4 exemples
Accumulation de gradients	2 étapes
Nombre d'époques	20
Optimiseur	AdamW
Taux d'apprentissage	10^{-5}

La taille de batch est fixée à 4 exemples, permettant une gestion raisonnable de la mémoire lors du traitement des données. L'accumulation de gradients est configurée avec 2 étapes, simulant efficacement une taille de batch plus grande de 8 pour améliorer la stabilité des gradients pendant les mises à jour. Le processus d'entraînement s'étend sur 20 époques, offrant suffisamment d'itérations pour que le modèle converge sur le dataset. Afin de minimiser la perte de cross-entropy, nous utilisons l'optimiseur AdamW, particulièrement adapté aux architectures basées sur les Transformers en raison de ses mécanismes de régularisation par décroissance des poids et de son taux d'apprentissage adaptatif. Ce dernier est fixé à 10^{-5}, assurant des mises à jour progressives et stables des poids tout au long de

l'entraînement. L'ensemble de ces hyperparamètres vise à équilibrer les performances du modèle avec l'efficacité computationnelle dans notre tâche d'AAC.

4.6 Résultats des Expérimentations et Interprétations

4.6.1 Expérience 1 : Impact du nombre de couches sur l'évaluation

Cette première expérience a pour objectif d'analyser l'influence du nombre de couches du modèle BART sur les performances en AAC. Trois configurations ont été testées : BART avec 4 couches (n = 4), BART Base avec 6 couches (n = 6) et BART Large avec 12 couches (n = 12), aussi bien pour l'encodeur que pour le décodeur. Les performances ont été évaluées à l'aide de plusieurs métriques, comme présenté dans le tableau 4-4.

Tableau 4-4: Résultats de l'évaluation des performances de différentes configurations de BART.

Modèle / Métrique	BART (n = 4)	BART Base (n = 6)	BART Large (n =12)
BLEU-1	0.555	0.554	0.547
BLEU-2	**0.363**	0.356	0.351
BLEU-3	**0.243**	0.235	0.234
BLEU-4	**0.156**	0.149	0.152
METEOR	0.165	0.165	0.161
ROUGE-L	**0.386**	0.364	0.359
CIDEr	0.349	0.348	0.339
SPICE	**0.108**	0.106	0.104
SPIDEr	0.228	0.227	0.222

Ces résultats révèlent que la configuration la plus légère, BART avec 4 couches, obtient globalement les meilleures performances sur la majorité des métriques d'évaluation. Elle se distingue notamment par des scores élevés en BLEU-2 (0.363), BLEU-3 (0.243), BLEU-4 (0.156), ROUGE-L (0.386) et SPICE (0.108). La configuration intermédiaire, BART Base avec 6 couches, présente des performances légèrement inférieures, mais reste relativement proche de BART (n = 4), en particulier sur les métriques BLEU-1, CIDEr et SPIDEr, et affiche un score identique à celui de BART (n = 4) pour la métrique METEOR (0.165). En revanche, le modèle le plus

profond, BART Large avec 12 couches, affiche une baisse plus significative des performances sur la majorité des métriques. Par exemple, le score BLEU-1 chute à 0.547, et ROUGE-L à 0.359.

Ces résultats suggèrent qu'une augmentation du nombre de couches ne garantit pas une amélioration des performances, et peut même nuire à la capacité de généralisation du modèle, en particulier dans le cas de données limitées ou spécifiques comme celles du dataset Clotho utilisé dans cette étude. Il est donc probable que les modèles plus profonds soient davantage sujets à un overfitting. Ainsi, dans l'AAC une architecture modérée comme BART Base offre un bon compromis entre performance et complexité, tandis qu'une architecture plus légère comme BART (n = 4) peut s'avérer à la fois plus efficace et moins coûteuse en ressources (temps d'entraînement, vitesse). Les performances prometteuses de cette configuration seront d'ailleurs confirmées dans l'expérience 2.

4.6.2 Expérience 2 : Impact du nombre de couches sur l'entraînement

L'expérience précédente a mis en lumière l'efficacité de la configuration légère (n = 4) en termes de performances liées à la génération de captions. Toutefois, ces performances doivent être mises en regard des coûts computationnels associés à chaque configuration. Dans cette seconde expérience, nous évaluons l'impact du nombre de couches sur les temps d'entraînement et la vitesse de traitement, mesurée en itérations par seconde. Le tableau 4-5 présente les résultats obtenus.

Tableau 4-5: Temps d'entraînement et vitesse selon le nombre de couches (n) du BART.

Modèle	**BART (n = 4)**	**BART Base (n = 6)**	**BART Large (n =12)**
Temps d'entraînement	1h21	1h46	4h15
Vitesse	9.87 it/s	7.49 it/s	3.12 it/s

Les résultats mettent en évidence que la configuration à n = 4 couches est non seulement la plus performante en termes de qualité de génération de captions, mais également la plus efficace sur le plan computationnel, avec un temps d'entraînement réduit à 1h21 et une vitesse de 9.87 it/s. À l'inverse, BART-Large (n = 12) requiert 4h15 pour l'entraînement, avec une vitesse fortement réduite (3.12 it/s), traduisant une complexité nettement plus

importante. BART Base (n = 6) se situe entre les deux, offrant un compromis acceptable avec un temps d'entraînement de 1h46 et une vitesse de 7.49 it/s.

Ces résultats soulignent donc que la configuration BART avec 4 couches ne se contente pas d'offrir de bonnes performances qualitatives, mais représente également un choix pertinent, particulièrement adapté non seulement au dataset Clotho, de taille limitée, mais aussi aux cas d'usage où les ressources computationnelles sont restreintes ou lorsque l'efficacité énergétique constitue un enjeu majeur.

4.6.3 Expérience 3 : Impact du Décodage sur la Qualité des Captions

Dans les expériences précédentes, le décodage par Beam Search a été utilisé par défaut. Dans cette expérience, afin d'évaluer l'impact des stratégies de décodage sur la qualité des captions générées, nous avons comparé les performances du décodage Beam Search et du décodage Greedy, en utilisant la configuration à n = 4 couches. Les résultats ont été évalués à l'aide de plusieurs métriques standards utilisées en génération de texte, comme illustré dans le tableau 4-6.

Tableau 4-6: Résultats d'évaluation des stratégies de décodage Beam Search et Greedy.

Modèle / Métrique	Beam Search	Greedy
BLEU-1	**0.555**	0.536
BLEU-2	**0.363**	0.328
BLEU-3	**0.243**	0.203
BLEU-4	**0.156**	0.120
METEOR	**0.165**	0.159
ROUGE-L	**0.386**	0.358
CIDEr	**0.349**	0.290
SPICE	0.108	**0.110**
SPIDEr	**0.228**	0.200

Les résultats montrent que la stratégie Beam Search surpasse globalement Greedy sur l'ensemble des métriques, à l'exception de SPICE. En particulier, les scores BLEU (de BLEU-1 à

BLEU-4) sont systématiquement plus élevés avec Beam, soulignant une meilleure capacité à générer des séquences cohérentes et proches des références, notamment pour les n-grammes plus longs (BLEU-4). Les scores METEOR et ROUGE-L confirment cette tendance en montrant un gain en précision lexicale et structurelle. De plus, les métriques adaptées à la tâche de captioning, telles que CIDEr et SPIDEr, révèlent une amélioration notable de la pertinence des descriptions produites avec Beam Search. La seule exception concerne la métrique SPICE, où Greedy atteint un score légèrement supérieur, ce qui pourrait indiquer une meilleure couverture des relations sémantiques dans certains cas. Globalement, ces résultats justifient l'utilisation du décodage Beam Search pour améliorer la qualité des captions générées dans notre système d'Audio Captioning.

4.6.4 Expérience 4 : Comparaison avec l'État de l'Art

Afin d'évaluer l'efficacité de notre modèle d'AAC, nous l'avons comparé aux travaux connexes présentés dans la section 2.10 du chapitre 2. Les résultats sont illustrés dans le tableau 4.7.

Tableau 4-7: Comparaison des performances métriques entre les travaux connexes et notre modèle.

Modèle / Métrique	MLP-Mixer +Transformer	ResNet-18 + GRU	CNN + Transformer	Notre Modèle
BLEU-1	0.461	0.449	0.483	**0.555**
BLEU-2	0.275	0.167	0.298	**0.363**
BLEU-3	0.180	0.068	0.197	**0.243**
BLEU-4	0.112	0.029	0.119	**0.156**
METEOR	0.126	0.097	0.133	**0.165**
ROUGE-L	0.312	0.284	0.322	**0.386**
CIDEr	0.210	0.098	0.243	**0.349**
SPICE	0.079	0.043	0.088	**0.108**
SPIDEr	0.144	0.071	0.166	**0.228**

Les résultats présentés montrent que notre modèle surpasse systématiquement les approches existantes sur l'ensemble des métriques. Les scores BLEU, montrent que notre modèle atteint les meilleures performances, avec un score BLEU-1 de 0.555 et un score BLEU-

4 de 0.156, ce qui indique une bonne capacité à générer des descriptions lexicalement précises, même pour des séquences plus longues. De plus, notre modèle obtient également les meilleurs résultats pour les scores METEOR et ROUGE-L, respectivement de 0.165 et 0.386, ce qui souligne une meilleure couverture sémantique et une structure linguistique plus cohérente. En ce qui concerne les métriques CIDEr, SPICE et SPIDEr, notre modèle se distingue nettement avec des scores de 0.349, 0.108 et 0.228, traduisant une similarité sémantique plus forte avec les annotations de référence. Ces résultats confirment la supériorité de notre approche, tant sur le plan lexical que sémantique, et démontrent la capacité de notre modèle à produire des descriptions audio plus fidèles, informatives et cohérentes.

4.7 Résultats de Test

Afin d'étudier ces résultats et de mettre en évidence les différences significatives, nous avons créé une application web à l'aide de la bibliothèque Gradio. Cette interface permet aux utilisateurs d'effectuer rapidement des expérimentations avec différents fichiers audio. La figure 4-3 illustre la page d'accueil que nous avons conçue, laquelle constitue le point d'entrée de notre application web d'AAC.

Figure 4-3: Page d'accueil de l'application

L'interface principale interactive de l'AAC que nous avons développée est présentée à la figure 4-4. Elle comporte trois champs principaux :

- **Champ 1** : permet d'importer n'importe quel fichier audio.
- **Champ 2** : permet de lire le fichier audio.
- **Champ 3** : affiche la caption prédite par notre modèle.

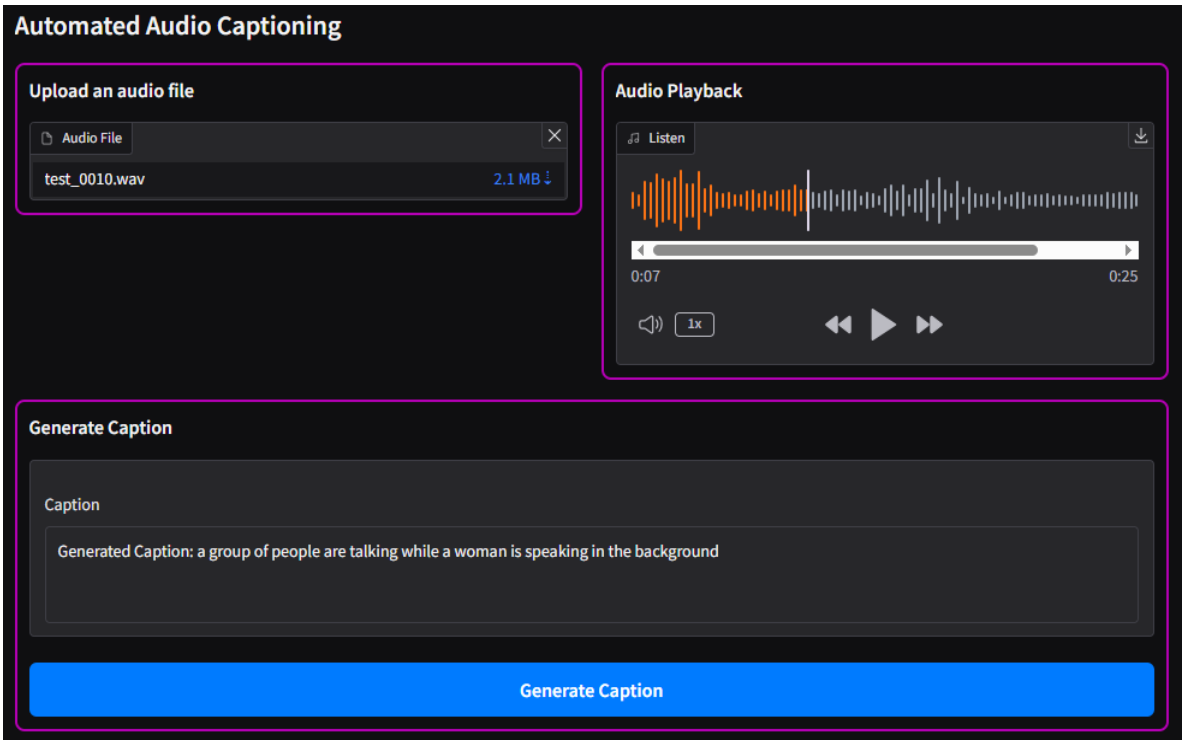

Figure 4-4: Interface interactive d'AAC.

Le tableau 4-8 présente les captions générées par notre modèle pour différents fichiers audio de test du dataset Clotho, au format wav.

Tableau 4-8: Les captions générées par notre modèle pour différents fichiers audio au format wav.

Fichier Audio	Caption Générée
test_0010.wav	a group of people are talking while a woman is speaking in the background
test_0059.wav	a bell is ringing while birds are chirping in the background
test_0648.wav	a machine is running at a constant rate
test_0704.wav	a metal object is being hit against a metal object

Ces résultats obtenus montrent que les captions générées sont globalement cohérentes, compréhensibles et pertinentes par rapport aux événements sonores détectés. Le système parvient non seulement à identifier l'événement principal, comme dans "a bell is ringing" ou "people are talking", mais aussi à capturer des éléments contextuels en arrière-plan, comme dans "a bell is ringing while birds are chirping in the background" ou "people are talking while music plays in the background". Cela démontre une certaine richesse descriptive ainsi qu'une capacité à gérer la polyphonie sonore, incluant des voix humaines, des bruits d'animaux (oiseaux, chiens), des éléments naturels (eau), des bruits mécaniques ou d'objets, ainsi que des sons musicaux et d'ambiance. De plus, les descriptions sont formulées dans un anglais grammaticalement correct, ce qui témoigne d'une bonne maîtrise de la génération en langage naturel.

Ainsi, même si les résultats sont globalement satisfaisants, des améliorations restent envisageables afin d'augmenter la précision et la richesse des descriptions générées. L'intégration de dataset plus vastes et plus diversifiés pourrait notamment contribuer à améliorer les performances du modèle.

Notre système ne se limite pas aux fichiers de test du dataset Clotho, il est également capable de générer des descriptions pour différents fichiers audio au format mp3 et wav, comme illustré dans le tableau 4-9.

Tableau 4-9: Les captions générées par notre modèle pour différents fichiers audio au format mp3 et wav.

Fichier Audio	Caption Générée
chirrido-de-puerta-335976.mp3	a squeaky door is opened and closed several times
City Ambiance-SoundBible.com.mp3	a siren sounds as a siren blaring in the background
mixkit-heavy-rain-and-thunder.wav	rain is pouring down while thunder roars in the background

Ces résultats illustrent que les fichiers sélectionnés couvrent une variété de sons du quotidien, tels qu'une porte grinçante, une ambiance urbaine avec une sirène, ou encore une pluie accompagnée de tonnerre. Les captions générées par notre système montrent qu'il est

capable de produire des descriptions cohérentes, compréhensibles et fidèles au contexte sonore. Cela met en évidence la capacité de généralisation du modèle, ce qui constitue un atout important pour son application à des données non vues pendant l'entraînement.

4.8 Conclusion et Résumé des Résultats Expérimentaux

Dans ce chapitre, nous avons présenté les résultats de nos quatre expérimentations, menées dans le but d'améliorer la qualité des descriptions générées. Voici un résumé des principaux enseignements tirés de ces analyses.

1. La configuration légère du modèle BART avec 4 couches pour l'encodeur et le décodeur s'est révélée être la plus performante, surpassant même les versions plus profondes BART Base et BART Large sur plusieurs métriques de qualité de génération.
2. Le modèle BART avec 4 couches offre également un avantage notable en termes de coût computationnel, avec un temps d'entraînement réduit et une vitesse d'exécution plus élevée, ce qui la rend particulièrement adaptée aux contextes à ressources limitées ainsi qu'aux dataset de petite taille, tels que Clotho.
3. L'adoption de la stratégie de décodage Beam Search a permis une amélioration significative de la qualité des captions générées par rapport au décodage Greedy
4. La comparaison avec des travaux connexes montre que notre modèle surpasse l'état de l'art sur l'ensemble des métriques évaluées, confirmant ainsi la pertinence de notre approche pour la tâche d'AAC, tant sur le plan lexical que sémantique.

Conclusion

Résumé des Résultats

Dans ce mémoire, nous avons abordé le défi de l'AAC, qui consiste à générer des descriptions textuelles en langage naturel à partir de contenus audio, en exploitant les avancées de l'apprentissage profond. L'objectif principal de notre démarche était de concevoir et d'évaluer un système capable d'interpréter avec précision les scènes sonores et d'en produire une description linguistique cohérente et pertinente. Pour cela, nous avons proposé un modèle séquence-à-séquence basé sur l'architecture Transformer BART, composé d'un encodeur bidirectionnel et d'un décodeur autorégressif. L'encodeur exploite des embeddings audio extraits à l'aide du modèle pré-entraîné VGGish, qui prend en entrée un spectrogramme log-Mel pour capturer les caractéristiques spectro-temporelles des signaux audio. Ces embeddings sont transformés via une couche affine pour s'adapter aux exigences du modèle BART. Le décodeur génère des captions mot par mot, en s'appuyant sur une tokenisation BPE pour gérer efficacement les mots rares et assurer une couverture linguistique robuste. Nos expériences, menées sur le dataset Clotho v2.1, ont étudié l'impact de différentes configurations du modèle BART. À partir de cette étude expérimentale, nous pouvons tirer les conclusions suivantes :

- La configuration à 4 couches a prouvé son efficacité, offrant un bon compromis entre performance et coût computationnel.
- Le décodage Beam Search améliore nettement la qualité des captions par rapport à Greedy, en produisant des séquences plus cohérentes.
- La comparaison avec d'autres architectures montre que notre modèle les dépasse, grâce à une précision lexicale plus élevée et une meilleure fidélité sémantique.

À la lumière des résultats obtenus, nous avons choisi le modèle Transformer BART avec 4 couches pour l'encodeur et le décodeur, associé à la stratégie de décodage Beam Search. Ce modèle a été intégré dans l'application web que nous avons conçue spécifiquement pour la tâche d'AAC, produisant des descriptions claires et pertinentes.

Limites et Travaux Futurs

Pour les travaux futurs de ce mémoire sur l'AAC, plusieurs pistes peuvent être explorées afin d'améliorer les performances du modèle proposé, de diversifier son champ d'application et de dépasser les limites identifiées.

1. L'intégration de modèles préentraînés plus avancés, tels que les PANNs, pourrait améliorer la qualité des embeddings audio et renforcer les performances du système.
2. Explorer des techniques d'adaptation, comme le fine-tuning par couches, pour spécialiser un modèle AAC sur des domaines spécifiques, notamment les sons liés aux animaux et aux véhicules, afin d'améliorer la qualité des captions.
3. Pour pallier la taille limitée du dataset Clotho, le transfert d'apprentissage à partir de datasets plus larges sera exploité afin d'améliorer les performances du modèle.
4. La personnalisation des captions audio en adaptant le niveau de détail, le style de langage et les priorités de contenu selon les préférences des utilisateurs. Par ailleurs, l'intégration de capacités multilingues et de modules de traduction automatique permettrait de rendre le système plus accessible à un public international et diversifié.
5. Explorer de nouvelles architectures capables de fonctionner en temps réel.
6. Nous proposons d'explorer des approches d'apprentissage multimodal intégrant des informations visuelles ou contextuelles supplémentaires.

Au cours de notre travail, nous souhaitions réaliser d'autres expériences afin d'améliorer la qualité des descriptions générées, mais nous avons rencontré des difficultés liées à la durée prolongée des temps d'entraînement, en raison de l'absence de matériel assez puissant. Malgré ces contraintes, cette expérience a été enrichissante, en nous permettant d'acquérir un ensemble de compétences précieuses. Nous avons ainsi approfondi notre compréhension des méthodes d'extraction des caractéristiques audio, découvert différentes architectures de réseaux neuronaux, et assimilé plusieurs concepts avancés en NLP. Par ailleurs, nous avons eu l'occasion d'explorer la plateforme Kaggle, que nous envisageons d'utiliser dans de futurs projets en ML

Bibliographie

[1] X. Mei, X. Liu, M. D. Plumbley, and W. Wang, “Automated audio captioning: an overview of recent progress and new challenges,” Dec. 01, 2022, *Springer Science and Business Media Deutschland GmbH*. doi: 10.1186/s13636-022-00259-2.

[2] T. Çakır, Emre; Heittola, Toni; Virtanen, “DOMESTIC AUDIO TAGGING WITH CONVOLUTIONAL NEURAL NETWORKS,” in *Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2016, pp. 448–456.

[3] Q. Kong, Y. Xu, I. Sobieraj, W. Wang, and M. D. Plumbley, “Sound Event Detection and Time-Frequency Segmentation from Weakly Labelled Data,” *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 27, no. 4, pp. 777–787, 2019, doi: 10.1109/TASLP.2019.2895254.

[4] D. Barchiesi, D. Giannoulis, D. Stowell, and M. D. Plumbley, “Acoustic Scene Classification,” 2014, doi: 10.1109/MSP.2014.2326181.

[5] K. Drossos, S. Adavanne, and T. Virtanen, “Automated audio captioning with recurrent neural networks,” *IEEE Work. Appl. Signal Process. to Audio Acoust.*, vol. 2017-Octob, pp. 374–378, 2017, doi: 10.1109/WASPAA.2017.8170058.

[6] X. Mei, X. Liu, M. D. Plumbley, and W. Wang, “Automated audio captioning: an overview of recent progress and new challenges,” *Eurasip J. Audio, Speech, Music Process.*, vol. 2022, no. 1, 2022, doi: 10.1186/s13636-022-00259-2.

[7] C. Won, Hyejin and Kim, Baekseung and Kwak, Il□Youp and Lim, “Using various pre-trained models for audio feature extraction in automated audio captioning,” *Expert Syst. Appl.*, vol. 231, p. 120664, 2023.

[8] P. A. Rossing, T. D., Moore, R. F., & Wheeler, *The Science of Sound*, 3rd ed. Addison Wesley, 2002.

[9] J. Halliday, D., Resnick, R., & Walker, *Fundamentals of Physics*, 10e ed. Wiley, 2013.

[10] L. Beranek, *Acoustics*. American Institute of Physics, 1993.

[11] G. Tipler, P. A., & Mosca, *Physics for Scientists and Engineers*, 6e ed. W. H. Freeman, 2008.

[12] É. Mixte, “Cours 1 : Conversions analogique-numérique et numérique-analogique.”

[13] K. C. Sedra, A. S., & Smith, *Microelectronic Circuits*, 8e ed. Oxford University Press, 2018.

[14] D. G. Proakis, J. G., & Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*, 4e ed. Pearson, 2006.

[15] M. Glénat, D. Chareyron, and M. Glénat, “Du passage de l’analogique au numérique,”

pp. 29–32, 2012.

[16] “CAN.” Accessed: Jul. 08, 2025. [Online]. Available: https://tpe-ssi-75.webself.net/can

[17] H. Schmickler, “Time- and Frequency-Domain Signals in Accelerators”.

[18] F. J. Harris, “On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform,” *Proc. IEEE*, vol. 66, no. 1, pp. 51–83, 1978, doi: 10.1109/PROC.1978.10837.

[19] R. G. Lyons, *Understanding Digital Signal Processing*.

[20] H. Schmickler, “Time-domain and Frequency-domain Signals and their Analysis,” Sep. 2020, [Online]. Available: http://arxiv.org/abs/2009.14544

[21] M. Müller, *Fundamentals of Music Processing*.

[22] M. Müller, “Fundamentals of Music Processing,” *Fundam. Music Process.*, no. January, 2015, doi: 10.1007/978-3-319-21945-5.

[23] O. G. Emmanuel and E. A. Kingsley, “BLIND SOURCE SEPARATION USING FREQUENCY DOMAIN INDEPENDENT COMPONENT ANALYSIS,” 2007.

[24] B. S. Goyal, Deepam and Pabla, “Condition based maintenance of machine tools—A review,” *CIRP J. Manuf. Sci. Technol.*, vol. 10, pp. 24--35, [Online]. Available: https://doi.org/10.1016/j.cirpj.2015.05.004%0A%0A

[25] G. Sharma, K. Umapathy, and S. Krishnan, “Trends in audio signal feature extraction methods,” *Appl. Acoust.*, vol. 158, p. 107020, 2020, doi: 10.1016/j.apacoust.2019.107020.

[26] B. P. C. Loizou, *Speech Enhancement Theory and Practice, Second Edition*.

[27] K. R. Borisagar, R. M. Thanki, and B. S. Sedani, “Speech enhancement techniques for Digital hearing aids,” *Speech Enhanc. Tech. Digit. Hear. Aids*, no. January 2019, pp. 1–150, 2018, doi: 10.1007/978-3-319-96821-6.

[28] D. P. W. Ellis, “An Introduction to Signal Processing for Speech,” *Handb. Phonetic Sci. Second Ed.*, pp. 755–780, 2010, doi: 10.1002/9781444317251.ch20.

[29] B. Zhang, J. Leitner, and S. Thornton, “Audio Recognition using Mel Spectrograms and Convolution Neural Networks,” *Noiselab Univ. Calif.*, vol. 3, no. 4, pp. 1–5, 2019, [Online]. Available: http://noiselab.ucsd.edu/ECE228_2019/Reports/Report38.pdf.

[30] E. Cakir, *Deep Neural Networks for Sound Event Detection Deep Neural Networks for Sound Event Detection*.

[31] K. Pyrovolakis, P. Tzouveli, and G. Stamou, “Multi-Modal Song Mood Detection with Deep Learning,” *Sensors*, vol. 22, no. 3, 2022, doi: 10.3390/s22031065.

[32] I. H. S. A. H. B. Hafeez, “Natural Language Processing: Applications, Techniques and Challenges,” no. December, 2022. doi: 10.22271/ed.book.784-CITATIONS.

[33] B. T. R.- Regulation, “Natural Language Processing (R20A6609) – Lecture Notes,” vol. 2, 2024.

[34] K. Kowsari, K. J. Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, “Text classification algorithms: A survey,” *Inf.*, vol. 10, no. 4, 2019, doi: 10.3390/info10040150.

[35] F. Qarah and T. Alsanoosy, “A Comprehensive Analysis of Various Tokenizers for Arabic Large Language Models,” *Appl. Sci.*, vol. 14, no. 13, 2024, doi: 10.3390/app14135696.

[36] “Word, Subword, and Character-Based Tokenization: Tokenizer for Deep learning (Part 1) - Zhihu.” Accessed: Jul. 08, 2025. [Online]. Available: https://zhuanlan.zhihu.com/p/512719309

[37] B. Suyunu, E. Taylan, and A. Özgür, “Linguistic Laws Meet Protein Sequences: A Comparative Analysis of Subword Tokenization Methods,” *Proc. - 2024 IEEE Int. Conf. Bioinforma. Biomed. BIBM 2024*, pp. 4489–4496, 2024, doi: 10.1109/BIBM62325.2024.10822826.

[38] A. Jakhotiya, H. Jain, B. Jain, and C. Chaniyara, “Text Pre-Processing Techniques in Natural Language Processing: A Review,” *Int. Res. J. Eng. Technol.*, vol. 9, no. 2, pp. 878–880, 2022.

[39] G. M. Di Nunzio and F. Vezzani, “A linguistic failure analysis of classification of medical publications: A study on stemming vs lemmatization,” *CEUR Workshop Proc.*, vol. 2253, 2018, doi: 10.4000/books.aaccademia.3327.

[40] H. Liu, T. Christiansen, W. A. Baumgartner, and K. Verspoor, “BioLemmatizer: A lemmatization tool for morphological processing of biomedical text,” *J. Biomed. Semantics*, vol. 3, no. 1, pp. 1–29, 2012, doi: 10.1186/2041-1480-3-3.

[41] Z. Liu, Y. Lin, and M. Sun, *Representation Learning for Natural Language Processing, Second Edition*. 2023. doi: 10.1007/978-981-99-1600-9.

[42] “Encoding Categorical Variables: One-hot vs Dummy Encoding | Towards Data Science.” Accessed: Jul. 08, 2025. [Online]. Available: https://towardsdatascience.com/encoding-categorical-variables-one-hot-vs-dummy-encoding-6d5b9c46e2db/

[43] R. Egger, “Text Representations and Word Embeddings: Vectorizing Textual Data,” *Tour. Verge*, vol. Part F1051, no. February, pp. 335–361, 2022, doi: 10.1007/978-3-030-88389-8_16.

[44] N. Dutta, “Word2Vec For Word Embeddings – A Beginner’s Guide,” pp. 1–8, [Online]. Available: https://www.analyticsvidhya.com/blog/2025/04/word2vec-for-word-embeddings-a-beginners-guide

[45] R. Patel and S. Patel, *Deep Learning for Natural Language Processing*, vol. 190. 2021. doi: 10.1007/978-981-16-0882-7_45.

[46] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient Estimation of Word Representations in Vector Space,” Jan. 2013, [Online]. Available: http://arxiv.org/abs/1301.3781

[47] D. Karani, “Introduction to Word Embedding and Word2Vec,” 2018.

[48] C. Zhang *et al.*, “Multi-Gram CNN-Based Self-Attention Model for Relation

Classification," *IEEE Access*, vol. 7, no. April 2021, pp. 5343–5357, 2019, doi: 10.1109/ACCESS.2018.2888508.

[49] B. Athiwaratkun, A. G. Wilson, and A. Anandkumar, "Probabilistic fasttext for multi-sense word embeddings," *ACL 2018 - 56th Annu. Meet. Assoc. Comput. Linguist. Proc. Conf. (Long Pap.*, vol. 1, pp. 1–11, 2018, doi: 10.18653/v1/p18-1001.

[50] T. Yeshambel, J. Mothe, and Y. Assabie, "Learned Text Representation for Amharic Information Retrieval and Natural Language Processing," *Inf.*, vol. 14, no. 3, 2023, doi: 10.3390/info14030195.

[51] G. Yenduri *et al.*, "GPT (Generative Pre-Trained Transformer) - A Comprehensive Review on Enabling Technologies, Potential Applications, Emerging Challenges, and Future Directions," *IEEE Access*, vol. 12, pp. 54608–54649, 2024, doi: 10.1109/ACCESS.2024.3389497.

[52] J. Devlin, M.-W. Chang, K. Lee, K. T. Google, and A. I. Language, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *Naacl-Hlt 2019*, no. Mlm, pp. 4171–4186, 2018, [Online]. Available: https://aclanthology.org/N19-1423.pdf

[53] J. Camacho-collados, "Embeddings in Natural Language Processing," *Morgan Publ.*, 2020, [Online]. Available: https://www.morganclaypoolpublishers.com/catalog_Orig/samples/9781636390222_sample.pdf

[54] "MLM — Sentence Transformers documentation." Accessed: Jul. 08, 2025. [Online]. Available: https://sbert.net/examples/sentence_transformer/unsupervised_learning/MLM/README.html

[55] Y. Liu *et al.*, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," no. 1, 2019, [Online]. Available: http://arxiv.org/abs/1907.11692

[56] Y. Sun, Y. Zheng, C. Hao, and H. Qiu, "NSP-BERT: A Prompt-based Few-Shot Learner Through an Original Pre-training Task ——— Next Sentence Prediction," *Proc. - Int. Conf. Comput. Linguist. COLING*, vol. 29, no. 1, pp. 3233–3250, 2022.

[57] M. Lewis *et al.*, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *Proc. Annu. Meet. Assoc. Comput. Linguist.*, pp. 7871–7880, 2020, doi: 10.18653/v1/2020.acl-main.703.

[58] L. Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020, pp. 7871–7880.

[59] D. Chen, "Challenges of Natural Language Processing from a Linguistic Perspective," vol. 13, no. 2, pp. 2–4, 2024.

[60] B. P. Yap, A. Koh, and E. S. Chng, "Adapting BERT for Word Sense Disambiguation with Gloss Selection Objective and Example Sentences," 2019.

[61] T. Shaik, X. Tao, Y. Li, C. Dann, and J. Mcdonald, “A Review of the Trends and Challenges in Adopting Natural Language Processing Methods for Education Feedback Analysis,” pp. 1–20.

[62] C.-A. Azencott, “Introduction au Machine Learning,” --, no., p., 2020.

[63] C. S. Deisenroth, M. P., Faisal, A. A., & Ong, *Mathematics for Machine Learning*, Cambridge. 2020.

[64] W. R. and Z. Wang, “Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review,” *Neural Comput.*, vol. 29, no. 9, pp. 2352–2449, 2017, doi: 10.1162/NECO.

[65] M. T. R. Prabowo, “Sentiment analysis: A combined approach,” *J. Informetr.*, vol. 3, no. 2, pp. 143–157, 2009.

[66] P. J. F. A. K. Jain, M. N. Murty, “Data clustering,” *ACM Comput. Surv.*, vol. 31, no. 3, 1999.

[67] K. S. N. Krishnaraj, M. Elhoseny, M. Thenmozhi, M. M. Selim, “Deep learning model for real-time image compression in internet of underwater things (IoUT),” *J. Real-Time Image Process.*, vol. 17, no. 6, pp. 2097–2111, 2020.

[68] E. Alpaydin, *Introduction to Machine Learning*, MIT Press. 2020.

[69] M. T. A. Goldwaser, “Deep reinforcement learning for general game playing,” *Proc. AAAI Conf. Artif. Intell.*, vol. 34, pp. 1701–1708, 2020.

[70] K. Isele, D.; Nakhaei, A.; Fujimura, “Safe reinforcement learning on autonomous vehicles,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Ed., 2018, pp. 1–6.

[71] Xihe Ge, “Brief Introduction to Artificial Neural Networks,” Jul. 2022.

[72] K. Suzuki, *Artificial Neural Networks – Methodological Advances and Biomedical Applications*. Tokyo Institute of Technology, 2011.

[73] David J. Livingstone, “Artificial Neural Networks: Methods and Applications.”

[74] “Présentation des réseaux de neurones artificiels - SINAT BLOG.” Accessed: Jul. 08, 2025. [Online]. Available: https://blog.sinatechnologie.com/presentation-des-reseaux-de-neurones-artificiels

[75] “ACTIVATION FUNCTIONS IN NEURAL NETWORKS,” 2020. [Online]. Available: http://www.ijeast.com

[76] A. Kane and A. Hussain, “Artificial Neural Networks - An overview,” *Indian Drugs*, vol. 30, no. 5, pp. 168–178, 1993, doi: 10.58496/mjcsc/2023/015.

[77] O. M. Surakhi and W. A. Salameh, “Enhancing the Performance of the BackPropagation for Deep Neural Network,” *Int. J. Comput. Technol.*, vol. 13, no. 12, pp. 5274–5285, 2014, doi: 10.24297/ijct.v13i12.5279.

[78] M. Cilimkovic, “Neural Networks and Back Propagation Algorithm,” *Inst. Technol. Blanchardstown, Blanchardst. Road North Dublin*, vol. 15, pp. 3–7, 2015.

[79] “Training ANNs: A deep dive into Backpropagation and Gradient descent.” Accessed: Jul. 08, 2025. [Online]. Available: https://medium.com/@omkarsoak/training-anns-a-deep-dive-into-backpropagation-and-gradient-descent-bea84ff5273c

[80] R. F. Li Wan, Matthew Zeiler, Sixin Zhang, Yann LeCun, “Regularization of Neural Networks using DropConnect,” in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, 2018, pp. 863–875. doi: 10.1109/TPAMI.2017.2703082.

[81] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *32nd Int. Conf. Mach. Learn. ICML 2015*, vol. 1, pp. 448–456, 2015.

[82] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.

[83] J. Bjorck, C. Gomes, B. Selman, and K. Q. Weinberger, “Understanding batch normalization,” *Adv. Neural Inf. Process. Syst.*, vol. 2018-Decem, no. NeurIPS, pp. 7694–7705, 2018.

[84] M. Vakili, M. Ghamsari, and M. Rezaei, “Performance Analysis and Comparison of Machine and Deep Learning Algorithms for IoT Data Classification,” 2020, [Online]. Available: http://arxiv.org/abs/2001.09636

[85] A. Murphy and C. Moore, “Confusion matrix,” *Radiopaedia.org*, no. October, 2019, doi: 10.53347/rid-68081.

[86] G. Varoquaux and O. Colliot, “Evaluating Machine Learning Models and Their Diagnostic Value,” *Neuromethods*, vol. 197, pp. 601–630, 2023, doi: 10.1007/978-1-0716-3195-9_20.

[87] O. Rainio, J. Teuho, and R. Klén, “Evaluation metrics and statistical tests for machine learning,” *Sci. Rep.*, vol. 14, no. 1, pp. 1–15, 2024, doi: 10.1038/s41598-024-56706-x.

[88] “Réseaux neuronaux et deep learning : différence entre domaines de l’intelligence artificielle – AWS.” Accessed: Apr. 16, 2025. [Online]. Available: https://aws.amazon.com/fr/compare/the-difference-between-deep-learning-and-neural-networks/

[89] H. Gu, Y. Wang, S. Hong, and G. Gui, “Blind channel identification aided generalized automatic modulation recognition based on deep learning,” *IEEE Access*, vol. 7, pp. 110722–110729, 2019, doi: 10.1109/ACCESS.2019.2934354.

[90] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998, doi: 10.1109/5.726791.

[91] “What are Convolutional Neural Networks? | IBM.” [Online]. Available: https://www.ibm.com/think/topics/convolutional-neural-networks

[92] V. Nair and G. E. Hinton, “Rectified linear units improve restricted Boltzmann machines,” in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 807–814.

[93] A. Goodfellow, Ian Bengio, Yoshua Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[94] “Comprendre les Réseaux de Neurones Convolutifs (CNN).” [Online]. Available: https://yannicksergeobam.medium.com/comprendre-les-réseaux-de-neurones-convolutifs-cnn-d5f14d963714

[95] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems (NeurIPS)*, Curran Associates, Inc., 2012, pp. 1097–1105.

[96] I. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 5998–6008.

[97] A. Dosovitskiy *et al.*, “an Image Is Worth 16X16 Words: Transformers for Image Recognition At Scale,” *ICLR 2021 - 9th Int. Conf. Learn. Represent.*, 2021.

[98] K. Miyazaki, T. Komatsu, T. Hayashi, S. Watanabe, T. Toda, and K. Takeda, “Detection and Classification of Acoustic Scenes and Events 2020 CONVOLUTION-AUGMENTED TRANSFORMER FOR SEMI-SUPERVISED SOUND EVENT DETECTION Technical Report,” pp. 2–5, 2020.

[99] X. Mei *et al.*, “An Encoder-Decoder Based Audio Captioning System With Transfer and Reinforcement Learning,” no. November, 2021, [Online]. Available: http://arxiv.org/abs/2108.02752

[100] J. G. Zhang, J. P. Li, and H. Li, “Language Modeling with Transformer,” *2019 16th Int. Comput. Conf. Wavelet Act. Media Technol. Inf. Process. ICCWAMTIP 2019*, pp. 249–253, 2019, doi: 10.1109/ICCWAMTIP47768.2019.9067534.

[101] “Transformer Explainer: LLM Transformer Model Visually Explained.” Accessed: Apr. 16, 2025. [Online]. Available: https://poloclub.github.io/transformer-explainer/

[102] S. S. Sayanta Paul, “CyberBERT: BERT for cyberbullying identification,” *Multimed. Syst.*, p. 28, doi: 10.1007/s00530-020-00710-4.

[103] A. Vaswani *et al.*, “Attention is all you need,” *Adv. Neural Inf. Process. Syst.*, vol. 2017-Decem, no. Nips, pp. 5999–6009, 2017.

[104] E. Models, “Machine Translation and Encoder-Decoder Models,” 2021.

[105] T. Virtanen, M. D. Plumbley, and D. Ellis, *Computational analysis of sound scenes and events*. 2017. doi: 10.1007/978-3-319-63450-0.

[106] R. J. W. D.E. Rumelhart, G.E. Hinton, “Learning representations by back-propagating errors. Nature,” *Nature*, vol. 324, p. 2, 1986.

[107] G. Kim, C.D., Kim, B., Lee, H., & Kim, “AudioCaps: Generating Captions for Audios in the Wild,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019, pp. 119–132.

[108] Y. Wu *et al.*, “Audio Captioning Based on Transformer and Pre-trained CNN,” *Detect.*

Classif. Acoust. Scenes Events 2020, no. November, pp. 2–5, 2020.

[109] X. Mei, X. Liu, J. Sun, M. D. Plumbley, and W. Wang, “Towards Generating Diverse Audio Captions via Adversarial Training,” *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 32, pp. 3311–3323, 2024, doi: 10.1109/TASLP.2024.3416686.

[110] K. Ikawa, S., & Kashino, “Neural audio captioning based on conditional sequence-to-sequence model,” in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*, New York University, New York, 2019, pp. 99–103.

[111] Y. Ye, Z., Wang, H., Yang, D., & Zou, “Improving the performance of automated audio captioning via integrating the acoustic and semantic information,” in *Proceedings of the 6th Detection and Classification of Acoustic Scenes and Events 2021 Workshop (DCASE2021)*, Barcelona, 2021, pp. 40–44.

[112] C. Won, H., Kim, B., Kwak, I.Y., & Lim, “Transfer learning followed by transformer for automated audio captioning,” in *Proceedings of the 6th Detection and Classification of Acoustic Scenes and Events 2021 Workshop (DCASE2021)*, Barcelona, 2021, pp. 221–225.

[113] F. Xiao, J. Guan, and Q. Kong, “AUTOMATED AUDIO CAPTIONING WITH MLP-MIXER AND PRE-TRAINED ENCODER Technical Report Group of Intelligent Signal Processing , College of Computer Science and Technology ByteDance , Shanghai , China,” 2021.

[114] A. Gebhard, A. Triantafyllopoulos, and A. Baird, “AN AUTOMATED AUDIO CAPTIONING APPROACH UTILISING A RESNET-BASED ENCODER Technical Report EIHW – Chair of Embedded Intelligence for Healthcare and Wellbeing , University of Augsburg , Augsburg , Germany GLAM – Group on Language , Audio , and Music , Imperi,” pp. 3–5, 2021.

[115] “THE DCASE2021 CHALLENGE TASK 6 SYSTEM : AUTOMATED AUDIO CAPTION Technical Report Liu Yang Beijing Institute of Technology Beijing Institute of Technology,” pp. 1–2, 2021.

[116] B. Dai, S. Fidler, R. Urtasun, and D. Lin, “Towards Diverse and Natural Image Descriptions via a Conditional GAN,” *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2017-Octob, pp. 2989–2998, 2017, doi: 10.1109/ICCV.2017.323.

[117] X. Mei, X. Liu, J. Sun, M. D. Plumbley, and W. Wang, “Diverse Audio Captioning via Adversarial Training,” Oct. 2021, [Online]. Available: http://arxiv.org/abs/2110.06691

[118] X. Xu, M. Wu, and K. Yu, “DIVERSITY-CONTROLLABLE AND ACCURATE AUDIO CAPTIONING BASED ON NEURAL CONDITION,” pp. 971–975, 2022.

[119] X. Xu, Z. Xie, M. Wu, and K. Yu, “Beyond the Status Quo: A Contemporary Survey of Advances and Challenges in Audio Captioning,” *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 32, pp. 95–112, 2024, doi: 10.1109/TASLP.2023.3321968.

[120] S. Hershey *et al.*, “CNN architectures for large-scale audio classification,” *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, pp. 131–135, 2017, doi: 10.1109/ICASSP.2017.7952132.

[121] M. Gemmeke, J.F.; Ellis, D.P.W.; Freedman, D.; Jansen, A.; Lawrence, W.; Moore, R.C.; Plakal, M.; Ritter, “Audio Set: An ontology and human-labeled dataset for audio events,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 776–780.

[122] “How To Increase Kaggle Output Directory Size? - Genspark.” Accessed: Apr. 25, 2025. [Online]. Available: https://www.genspark.ai/spark/how-to-increase-kaggle-output-directory-size/1f7e121d-650a-4e4a-a4ce-202388fedc90?utm_source=chatgpt.com

[123] “What is Kaggle? | DataCamp.” Accessed: Apr. 25, 2025. [Online]. Available: https://www.datacamp.com/blog/what-is-kaggle

[124] TutorialsPoint, “Machine Learning with Python,” *Data Vault 2.0*, pp. 1–167, doi: 10.1007/978-3-322-94873-1.

[125] E. Stevens, *Deep leanring with PyTorch*. 2021.

[126] A. Vidhya, “Hands-On Guide To Librosa For Handling Audio Files”.

[127] “Transformers.” Accessed: Apr. 27, 2025. [Online]. Available: https://huggingface.co/docs/transformers/index

[128] F. Font, G. Roma, and X. Serra, “Freesound technical demo,” *MM 2013 - Proc. 2013 ACM Multimed. Conf.*, pp. 411–412, 2013, doi: 10.1145/2502081.2502245.

[129] S. Lipping, K. Drossos, and T. Virtanen, “Crowdsourcing a Dataset of Audio Captions,” no. October, pp. 139–143, 2019, doi: 10.33682/sezz-vd31.

[130] K. Drossos, S. Lipping, and T. Virtanen, “Clotho: An Audio Captioning Dataset,” Oct. 2019, [Online]. Available: http://arxiv.org/abs/1910.09387

[131] W.-J. Papineni, K.; Roukos, S.; Ward, T.; Zhu, “BLEU: a Method for Automatic Evaluation of Machine Translation,” in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, A. for C. Linguistics, Ed., Philadelphia, Pennsylvania, USA, pp. 311–318. doi: 10.1002/andp.19223712302.

[132] A. Banerjee, Satanjeev ; Lavie, “METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments,” in *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, Ann Arbor, Michigan, USA: Association for Computational Linguistics, 2005, pp. 65–72. doi: 10.1145/2567940.

[133] C.-Y. Lin, “ROUGE: A Package for Automatic Evaluation of Summaries,” in *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, Association for Computational Linguistics, 2004, pp. 74–81. doi: 10.1253/jcj.34.1213.

[134] R. Vedantam, C. L. Zitnick, and D. Parikh, “CIDEr: Consensus-based image description evaluation,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07-12-June, pp. 4566–4575, 2015, doi: 10.1109/CVPR.2015.7299087.

[135] S. Anderson, P.; Fernando, B.; Johnson, M.; Gould, “SPICE: Semantic propositional image caption evaluation,” in *Computer Vision – ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part V*, 2016, pp. 382–398. doi: 10.1007/978-3-319-46454-1_24.

[136] S. Liu, Z. Zhu, N. Ye, S. Guadarrama, and K. Murphy, “Improved Image Captioning via Policy Gradient optimization of SPIDEr,” in *Proceedings of the IEEE International Conference on Computer Vision*, Institute of Electrical and Electronics Engineers Inc., Dec. 2017, pp. 873–881. doi: 10.1109/ICCV.2017.100.