Description automatique des événements sonores par des méthodes d'apprentissage profond

Etienne Labbé

RÉPUBLIQUE FRANÇAISE
Liberté Égalité Fraternité

Université de Toulouse

Doctorat de l'Université de Toulouse

préparé à l'Université Toulouse III - Paul Sabatier

Description automatique des événements sonores par des méthodes d'apprentissage profond

---

Thèse présentée et soutenue, le 3 avril 2024 par

Étienne LABBÉ

École doctorale
EDMITT - Ecole Doctorale Mathématiques, Informatique et Télécommunications de Toulouse

Spécialité
Informatique et Télécommunications

Unité de recherche
IRIT : Institut de Recherche en Informatique de Toulouse

Thèse dirigée et encadrée par
Julien PINQUIER et Thomas PELLEGRINI

Composition du jury
Mme Farah BENAMARA, Présidente, Université Toulouse III - Paul Sabatier
M. Geoffroy PEETERS, Rapporteur, Institut Polytechnique de Paris
M. Romain SÉRIZEL, Rapporteur, Université de Lorraine
Mme Annamaria MESAROS, Examinatrice, Tampere University
M. Alain RAKOTOMAMONJY, Examinateur, Université de Rouen Normandie
M. Julien PINQUIER, Directeur de thèse, Université Toulouse III - Paul Sabatier

Membres invités
M. Thomas PELLEGRINI, Encadrant de thèse, Université Toulouse III - Paul Sabatier

Remerciements

Je n'ai jamais été très doué pour écrire de beaux paragraphes bien construits, et c'est pourquoi je n'arriverai pas à totalement exprimer en ces quelques lignes tout ce que je dois aux gens qui m'ont entouré. Durant ces trois dernières années, et malgré quelques confinements, j'ai eu l'occasion de rencontrer de nombreuses personnes qui m'ont aidé à réussir cette thèse. Tout d'abord, je remercie le Ministère de l'Enseignement Supérieur de la Recherche et de l'Innovation pour le financement de cette thèse ainsi que l'équipe SAMoVA de l'Institut de Recherche en Informatique de Toulouse dans laquelle s'est déroulée cette thèse. Je remercie tous les membres du jury pour leur bienveillance durant la soutenance, et en particulier les rapporteurs pour leurs relectures détaillées de ce document.

Je tiens à remercier tout particulièrement mes encadrants, Thomas Pellegrini et Julien Pinquier, pour m'avoir guidé tout au long de ces trois années. C'est grâce à vous que j'ai pu accepter mes résultats et analyses, ce qui m'a permis d'arriver à les réaliser, mais aussi à les publier et à présenter durant certaines conférences. Je n'oublierai jamais nos réunions hebdomadaires à trois, durant lesquelles nous lancions de nombreuses idées (bien souvent farfelues).

Je félicite aussi tous ceux qui m'ont supporté dans leur bureau pendant ces trois années : Vincent, partenaire dos-à-dos du bureau 220 ; Benjamin, débatteur sur le *deep learning* aux connaissances infinies ; Lucile, partenaire de la pomme du soir qui a beaucoup inspiré cette thèse ; Mathieu, modèle de docteur presque parfait ; Alexis, disciple le jour et compère en ligne le soir ; et Ludovic, le prêcheur de ChatGPT. Je n'oublierai pas nos discussions, nos rires et ce temps passé ensemble à travailler à seulement quelques mètres pendant nos thèses.

Merci infiniment à Verdiana, qui a été ma consœur avec laquelle nous avons enduré nos derniers mois de thèse. Nos (très longues) pauses sur la terrasse de l'IRIT, m'ont beaucoup aidé à supporter toute la pression. Merci aussi à Léo, pour m'avoir introduit au monde de la recherche, m'avoir appris tant de choses sur les réseaux de neurones et l'apprentissage automatique durant mon stage ainsi que ma première année de thèse (malgré les confinements). Je remercie aussi énormément Lila, qui a été la première à combattre mon négativisme et à me supporter moralement. Merci aussi à Sebastião pour ton enthousiaste et ta bonne humeur contagieuse.

Merci également à toutes les autres personnes de l'équipe SAMoVA, qui ont grandement contribué à une ambiance exceptionnelle à travers nos nombreuses parties de cartes en salle machine, et qui m'ont motivé et permis de garder le sourire durant tout ce temps : Robin, Jim, Timothy, Romain, Matéo, Bilal, Florent, Adrian, Gautier, Amélie, Antoine, Adrien, Philippe, Martin, Clément... Je remercie aussi tous les permanents de l'équipe (Hervé, Jérôme, Julie, Christine, Régine et Isabelle) pour leur accueil et leurs conseils.

Merci à ma famille, qui a continué de me soutenir et posé tant de questions sur la fameuse « intelligence artificielle ». Merci aussi à Matthieu et Apolline, pour toutes nos soirées passées et futures à jouer, s'amuser, ou simplement à parler de nos vies. Et pour terminer, la meilleure pour la fin : merci infiniment à Bérangère, qui est mon plus grand soutien depuis des années. Tu as toujours été le centre de mon monde sur lequel tout repose, et je sais que cette thèse n'aurait même pas commencé si tu n'avais pas été là.

Table des matières

Table des figures

Liste des tableaux

Glossaire et acronymes

A2T *Audio-to-Text retrieval*.
AAC *Automated Audio Captioning*.
AC Jeu de données AudioCaps, décrit dans la section 1.3.1.1.
ACES Métrique *Audio Captioning Evaluation on Semantics of Sound*, décrite dans la section 2.3.6.
ACT *Audio Captioning Transformer*.
AS Jeu de données AudioSet.
AS$^{\text{-AC}}$ Jeu de données AudioSet sans aucun recouvrement avec tous les sous-ensembles d'AudioCaps.
AS-SL *AudioSet Strongly Labeled subset*.
AST *Audio Spectrogram Transformer*.
AT *Audio Tagging*.

B1 Métrique BLEU1, décrite dans la section 2.1.1.
B2 Métrique BLEU2, décrite dans la section 2.1.1.
B3 Métrique BLEU3, décrite dans la section 2.1.1.
B4 Métrique BLEU4, décrite dans la section 2.1.1.
BART *Bidirectional and Auto-Regressive Transformers*.
BAT *Before-After Test*, décrit dans la section 8.5.
BBC Site web BBC Sound Effects.
BEATs *Bidirectional Encoder representation from Audio Transformers*.
BERT *Bidirectional Encoder Representations from Transformers*.
BiGRU *Bidirectionnal Gated Recurrent Unit*, décrit dans la section 1.2.2.3.
BiLSTM *Bidirectionnal Long Short-Term Memory*, décrit dans la section 1.2.2.3.
BiRNN *Bidirectionnal Recurrent Neural Network*, décrit dans la section 1.2.2.3.
BN *Batch Normalization*.
BP *Brevity Penalty*, décrit dans l'équation 2.2.
BPE *Byte-Pair Encoding*, décrit dans la section 1.4.1.2.

CCP Test *Correct-Caption Pair*, décrit dans la section 2.3.8.
CD Métrique CIDEr-D, décrite dans la section 2.2.1.
CE *Cross Entropy loss*, décrite par l'équation 1.7.
CL Jeu de données Clotho, décrit dans la section 1.3.1.2.
CLAP *Contrastive Language-Audio Pretraining*.
CLv Sous-ensemble de validation de Clotho.
CNN *Convolutional Neural Network*, décrit dans la section 1.2.1.
CoNeTTE *ConvNext-Transformer with Task Embedding*, décrit dans le chapitre 7.
CPU *Central Processing Unit*.

DCASE *Detection and Classification of Acoustic Scenes and Events*.

DNN *Deep Neural Network*.
DSC *Depthwise Separable Convolution*.
DTAA Description Textuelle Automatique de l'Audio.
DTAI Description Textuelle Automatique de l'Image.
DTAV Description Textuelle Automatique de la Vidéo.
DTAVA Description Textuelle Automatique de la Vidéo et de l'Audio.

EMA Moyenne glissante exponentielle, *Exponential Moving Average*.
ESC-10 Jeu de données *Environmental Sound Classification*, décrit dans la section 9.3.1.

FER Taux d'erreur de fluence, *Fluency Error Rate*.
FFN *Feed-Forward Network*.
FM *FixMatch*, décrite dans la section 9.2.5.
FNS Métrique FENSE, décrite dans la section 2.3.2.
FSD Site web Freesound.
FSD50K Jeu de données FSD50K.

GELU *Gaussian Error Linear Unit*.
GPU *Graphics Processing Unit*.
GRU *Gated Recurrent Unit*, décrit dans la section 1.2.2.2.
GSC Jeu de données *Google Speech Commands*, décrit dans la section 9.3.1.

Hc Test Humain-correct, décrit dans la section 2.3.2.
Hi Test Humain-incorrect, décrit dans la section 2.3.2.
Hm Test Humain-machine, décrit dans la section 2.3.2.
HTSAT *Hierachical Token-Semantic Audio Transformer*, décrit dans la section 1.4.1.3.

ICP Test *Incorrect-Caption Pair*, décrit dans la section 2.3.8.

LAS *Listen, Attend and Spell*, décrit dans la section 3.1.
LAT *Listen, Attend and Tell*, décrit dans la section 3.1.
LCS *Longest Common Subsequence*.
LS *Label Smoothing*.
LSTM *Long Short-Term Memory*, décrit dans la section 1.2.2.1.

MA Jeu de données *Multi Annotator Captioned Soundscapes*, décrit dans la section 1.3.1.3.
MACs *Multiply-Accumulate operations*.
mAP Métrique *mean Average Precision*.
ME Métrique METEOR, décrite dans la section 2.1.3.
MHA Attention à plusieurs têtes, *Multi-Head Attention*, décrit par l'équation 3.1.
MM *MixMatch*, décrite dans la section 9.2.3.
Mm Test Machine-machine, décrit dans la section 2.3.2.
MSE *Mean Squared Error*.
MT *Mean Teacher*, décrite dans la section 9.2.2.

NLLB *No Language Left Behind*.

PANN *Pretrained Audio Neural Network*.
PaSST *Patchout faSt Spectrogram Transformer*, décrit dans la section 1.4.1.3.
PL *Pseudo-Label*, décrite dans la section 9.2.1.

PYB PANN-Yamnet-BART model, décrit dans la section 1.4.1.4.

RAP Reconnaissance Automatique de la Parole.
RAT Recherche Audio-Texte.
ReLU *Rectified Linear Unit*.
RG Métrique ROUGE-L, décrite dans la section 2.1.2.
RL *Reinforcement Learning*.
RMM *ReMixMatch*, décrite dans la section 9.2.4.
RNN *Recurrent Neural Network*, décrit dans la section 1.2.2.

S2V Métrique s2vscore, décrite dans la section 2.3.8.
SA SpecAugment, décrit dans la section 1.4.4.1.
SB Site web SoundBible.
SB_{sim} Similarité cosinus entre les représentations données par un modèle *Sentence-BERT*.
SBERT Modèle *Sentence-BERT*.
SBF Métrique *Similarity-Based F-score*, décrite dans la section 2.3.7.
SC Métrique SPICE, décrite dans la section 2.2.2.
SCST *Self-Critical Sequence Training*.
SD Métrique SPIDEr, décrite dans la section 2.2.3.
SD_{FL} Métrique SPIDEr-FL, décrite dans la section 2.3.3.
SDPA *Scaled Dot-Product Attention*, décrit par l'équation 1.6.
SDs Jeu de données SoundDescs, décrit dans la section 1.3.2.2.
SER *Sentence Embedding Regression loss*.
SJ Site web SoundJay.
SLURM *Simple Linux Utility for Resource Management*.
SR1 Système de Référence 1, décrit dans le chapitre 3.
SR2 Système de Référence 2, décrit dans le chapitre 5.
SR3 Système de Référence 3, décrit dans le chapitre 5.
SSL *Semi-Supervised Learning*.
SUP10 Apprentissage supervisé n'utilisant que 10% des données d'entrainement.
SUP100 Apprentissage supervisé utilisant 100% des données d'entrainement.

T2A *Text-to-Audio retrieval*.
TE *Task Embedding*, Représentation de tâche.
TFD Transformée de Fourier Discrète, décrit par l'équation 1.11.

UBS8K Jeu de données *UrbanSound 8k*, décrit dans la section 9.3.1.
UDA *Unsupervised Data Augmentation*, décrite dans la section 9.2.6.

WC Jeu de données WavCaps, décrit dans la section 1.3.1.4.
WC** Jeu de données WavCaps sans aucun recouvrement avec tous les sous-ensembles d'AudioCaps et Clotho, et sans les fichiers audio durant plus de 30s.
WC* Jeu de données WavCaps sans aucun recouvrement avec les sous-ensembles de validation et de test d'AudioCaps et Clotho.
WSJ Jeu de données *Wall Street Journal corpus*.
WT Jeu de données WavText5k, décrit dans la section 1.3.2.3.

ZT Site web Zapsplat.

Introduction

Problématique et contexte de la thèse

Parmi les cinq sens de l'être humain, l'ouïe nous fournit de précieuses informations sur notre environnement. Il arrive souvent que ce sens soit complémentaire de la vision, ce qui nous permet de reconnaître quotidiennement de nombreux événements pour nous aider à interagir avec le monde. Le traitement automatique de l'audio peut donc aider plusieurs applications, comme la domotique, l'assistance à l'autonomie à domicile ou la sécurité. Mais dans la plupart des contextes, les systèmes automatiques traitant les événements sonores se focalisent sur un nombre restreint de sons, et définissent donc en amont un ensemble de classes à détecter. Par exemple, la tâche de classification ou d'étiquetage audio (*Audio Tagging*, AT), doit reconnaître la ou les classes d'événements sonores présentes dans un fichier audio. La classification de scènes acoustiques vise à reconnaître le lieu d'où provient un enregistrement audio (*Acoustic Scene Classification*). Enfin, la détection d'événements sonores (*Sound Event Detection*) se focalise sur la localisation temporelle de certains sons dans un signal audio.

Cependant, lorsque l'on s'intéresse à traiter n'importe quel type de son, les modèles de classification ou de détection se heurtent à une limite vis-à-vis des classes prévues. Pour briser cette barrière, les créateurs des jeux de données d'événements sonores ont commencé à avoir recours au langage naturel sous forme de texte, qui permet une plus grande expressivité et diversité. Parmi ces tâches audio-texte, il existe la Recherche Audio-Texte (*Audio-Text Retrieval*, RAT) [Koepke 2022], qui vise à concevoir des systèmes capables d'extraire un fichier audio ou texte de sa base de données à partir d'une requête exprimée dans une modalité différente. Nous parlons de recherche Texte-vers-Audio (*Text-to-Audio retrieval* ou *Language-Based Audio Retrieval*, T2A) lorsque la requête est exprimée sous forme de texte et la base de données sous forme d'une liste de fichiers audio. Si les modalités sont opposées, nous nommons la tâche la recherche Audio-vers-Texte (*Audio-to-Text retrieval*, A2T). Il existe également d'autres tâches, comme la réponse automatique aux questions audio (*Audio Question Answering*) [Abdelnour 2018] qui vise à créer des systèmes capables de générer un texte qui répond à une question sur un fichier audio en entrée. Les réponses peuvent être de simples mots ou des phrases plus détaillées (comme *« Combien de tics d'horloge entend-on ? »*, avec la réponse *« Trois. »*). La détection d'événements sonores contrôlée par du texte (*Text-to-Audio Grounding*) [Xu 2021c] mélange les tâches de détection d'événements sonores et de recherche audio-texte, avec pour objectif de localiser un événement sonore décrit par une phrase dans un fichier audio. Nous trouvons également la séparation d'événements sonores contrôlée par du texte (*Text-driven Audio Source Separation* ou *Language-queried audio source separation*), qui propose de séparer un événement sonore décrit par du texte d'un fichier audio en entrée. Il existe également la tâche inverse de la DTAA, nommée la génération texte-vers-son (*Text-to-Sound Generation*) [Yang 2023], qui vise à créer des modèles qui produisent un signal audio à partir d'une phrase. Récemment, la description de différences audio par du texte (*Audio Difference Captioning* ou *Audio change captioning*) [Tsubaki 2023, Takeuchi 2023], a été introduite pour réaliser des systèmes capables

de décrire la différence entre deux fichiers audio, pour détecter des détails précis dans un signal ou repérer des événements anormaux.

Dans cette thèse, nous portons notre attention sur une tâche audio-texte en particulier : la Description Textuelle Automatique de l'Audio (*Automated Audio Captioning*, DTAA) [Drossos 2017]. Elle vise à concevoir des systèmes automatiques capables de décrire les événements sonores d'un fichier audio à l'aide du texte. Une illustration de l'entrée audio et de la sortie texte d'un système de DTAA est donnée dans la figure 1. Dans cette dernière, le modèle prend en entrée un signal audio et génère la phrase *« Un groupe d'hommes, de femmes et d'enfants discutent. »*.

FIGURE 1 – Illustration du traitement d'un système de DTAA.

Les descriptions produites par les systèmes de DTAA doivent résumer au mieux l'information sonore, en imitant la perception humaine. De plus, les phrases produites doivent être claires, précises et grammaticalement correctes. La DTAA traite toutes sortes d'événements sonores comme des sons environnementaux (pluie, vent, cris d'animaux...), urbains (voitures, klaxon, avion...), domestiques (vaisselle, ventilateurs, lave-linge...), des bruitages (frottements, sonneries, effets sonores synthétiques...), de la musique (chant, guitare...), des sons d'origine humaine (applaudissements, ronflements...) et bien évidemment de la parole (discussions, brouhaha, voix non verbales, parole de femmes, d'hommes ou d'enfants...). Cependant, même si le fichier audio peut contenir de la parole, les mots prononcés ne seront pas retranscrits par un système de DTAA. Nous éviterons donc le terme « sous-titrage » dans ce manuscrit pour ne pas confondre avec la Reconnaissance Automatique de la Parole (RAP), qui vise à transcrire la voix sous forme de texte. La DTAA se distingue également de l'audiodescription, qui a pour objectif d'ajouter une voix off pour décrire le contenu visuel et non sonore d'une vidéo, à destination de personnes aveugles ou malvoyantes. Bien évidemment, des tâches de description multimodale existent également pour d'autres modalités que l'audio comme l'image (DTAI), la vidéo (DTAV) ou la combinaison vidéo et audio (DTAVA). Le principe de la tâche reste le même, mais le modèle, les métriques et les jeux de données doivent s'adapter à la modalité en question. Plusieurs applications de la DTAA peuvent être envisagées. L'une d'entre elles est l'aide aux personnes sourdes ou malentendantes. Cela pourrait prendre la forme d'une application pour aider ces personnes à reconnaître les événements environnants, ou bien pour enrichir automatiquement les descriptions de films ou de contenus vidéos en général. Une autre application serait l'analyse de l'environnement sonore à destination de systèmes de surveillance automatique, pour repérer des comportements ou événements suspects à partir du son. Enfin, une autre application possible est la recherche de contenu audio plus efficace dans de larges bases de données audio, qui pourrait être réalisé à l'aide des résumés texte produits par les systèmes de DTAA.

Cette thèse se concentre essentiellement sur les aspects d'apprentissage automatique et d'évaluation nécessaires pour concevoir un système de DTAA (nommé description automatique des événements sonores dans le titre de ce manuscrit). Plus précisément, nous cherchons à créer un système « généraliste », c'est-à-dire obtenant une performance comparable aux autres systèmes

de l'état de l'art sur plusieurs jeux de données par un seul modèle. Notre modèle généraliste est nommé CoNeTTE, pour ***Co**nv**Ne**X**t**-**T**ransformer with **T**ask **E**mbedding*, que nous décrivons dans ce manuscrit. Elle a été financée par le Ministère de l'Enseignement Supérieur de la Recherche et de l'Innovation, ainsi que par le projet de l'Agence Nationale de la Recherche nommé *Lightly Unsupervised Discovery of Audio Units*[1], mené par l'un de mes encadrants, Thomas Pellegrini. Ce projet vise à étudier les méthodes d'apprentissage faiblement et semi-supervisées pour classifier et détecter les événements sonores. La thèse s'est déroulée pendant trois ans dans l'équipe Structuration, Analyse et Modélisation de documents Vidéo et Audio, et dans les locaux de l'Institut de Recherche en Informatique de Toulouse, situés à l'Université Toulouse III - Paul Sabatier.

Verrous scientifiques

La DTAA est une tâche assez difficile à réaliser par les systèmes automatiques. En effet, les systèmes doivent non seulement reconnaître n'importe quel événement sonore, mais résumer les informations sous forme de texte. Chaque mot du vocabulaire est considéré comme une unité différente par les systèmes, ce qui rend l'apprentissage automatique difficile. Par exemple, *« Birds call/chirp/sing/tweet »* indiquent tous des événements similaires, mais avec quatre verbes différents. Les annotateurs des jeux de données peuvent aussi utiliser des expressions ou des tournures de phrases (comme *« A door **that needs to be oiled** [...] »*) pour décrire certaines caractéristiques sonores. De plus, les descriptions contiennent également des éléments subjectifs, comme la qualité d'une conversation. Plusieurs phrases contiennent des informations directement ou indirectement liées au contenu de la parole, comme la langue parlée ou l'intelligibilité d'une personne. En outre, certains jeux de données sont parfois annotés par plusieurs phrases, qui ne décrivent pas toujours les mêmes événements sonores d'un fichier audio. Des caractéristiques sonores précises peuvent être détaillées dans les phrases, comme la fréquence, l'intensité, la durée, la réverbération ou encore la qualité du microphone ayant effectué l'enregistrement (comme *« Person chatting with **poor audio record** »*). Toutes ces informations peuvent être sporadiquement fournies par les annotateurs dans les descriptions des jeux de données, qui ne sont pas dans l'obligation d'ajouter tous ces détails. De plus, comme les systèmes de DTAA ont pour objectif d'imiter la perception humaine, mais certaines phrases décrivent des sources sonores ambiguës ou indiscernables par les annotateurs (comme *« **Water or air** is being pushed out of something [...] »*). Les modèles doivent également être capables de reconnaître des informations de plus haut niveau à partir des phrases, ce qui inclut les relations entre les événements sonores comme la superposition, la séquence et la répétition. Certaines descriptions peuvent aussi préciser le nombre d'occurrences, la différence d'intensité ou la variation de fréquence entre deux événements sonores d'un même fichier.

Face à toutes ces difficultés, la DTAA pose nombreux défis pour arriver à créer des systèmes traitant l'audio et le texte tout en liant les deux modalités. Dans ce manuscrit, nous souhaitons répondre à plusieurs questions autour de ces systèmes :

1. La méthode d'évaluation actuelle des systèmes est-elle suffisante ? Comment la métrique principale se comporte-t-elle lorsqu'elle est soumise à plusieurs phrases similaires ?
2. Quelles sont les meilleures architectures et méthodes d'apprentissage pour construire un modèle de DTAA ? Comment créer une architecture efficace en dépit des limites des jeux

1. https://anr.fr/Project-ANR-18-CE23-0005

de données disponibles ?

3. Comment pouvons-nous tirer parti de plusieurs jeux de données pour créer un système généraliste et performant ?
4. Pouvons-nous étendre ce travail à d'autres tâches (comme la RAT) et d'autres domaines (comme une autre langue) ?

Organisation du manuscrit

Ce manuscrit est divisé en trois grandes parties : un état de l'art, des contributions sur la DTAA et une ouverture sur des tâches et des domaines différents. Le total est composé de 10 chapitres.

La première partie se consacre à la description de nombreuses études de la littérature de DTAA et à la présentation de notre premier système. Le chapitre 1 détaille l'état de l'art des méthodes et des jeux de données de DTAA. Il décrit notamment les données publiquement accessibles qui seront utilisées pour nos expériences, ainsi que l'historique des différents modèles de la littérature. Nous donnons un aperçu d'une majorité de ce qui est et a été employé entre 2017 et 2023, et nous reportons également l'évolution des scores de la principale métrique nommée SPIDEr, par année et nombre de paramètres des modèles.

Le chapitre 2 explique l'ensemble des métriques qui sont employées pour évaluer les modèles de DTAA. L'évaluation nécessitant de comparer du texte, il existe un très large éventail de métriques pour comparer les sorties d'un modèle avec les références. Nous rapportons le fonctionnement de chaque métrique, ainsi que leurs limites. Puis, nous donnons également un aperçu des scores de ces métriques sur les jeux de validation et de tests disponibles.

Le chapitre 3 résume nos premières expériences pour concevoir un système de DTAA fortement inspiré de la littérature. Nous avons commencé par adapter un réseau de neurones récurrent initialement développé pour la RAP, puis nous avons suivi des méthodes plus modernes en récupérant les poids d'un modèle pré-entraîné et en modifiant l'architecture. Nous détaillons le processus d'entraînement, de validation et de test de ce premier système nommé SR1, ainsi que les résultats obtenus. Nous mentionnons également nos contributions sous forme de logiciels libres au travers de deux bibliothèques. La première permet de facilement télécharger et lire les jeux de données. La seconde facilite l'utilisation et l'installation des métriques de DTAA.

La seconde partie se concentre sur nos contributions spécifiques à la DTAA à travers l'entraînement et l'évaluation de cette tâche.

Le chapitre 4 analyse la principale métrique adoptée pour évaluer les systèmes de DTAA, SPIDEr. À l'aide d'une proposition de nouvelle métrique connexe nommée SPIDEr-max, nous étudions la variabilité de la métrique SPIDEr lorsqu'elle est soumise à plusieurs phrases similaires générées par notre système SR1. Nous montrons également comment les scores de cette métrique peuvent évoluer lorsque nous choisissons automatiquement la meilleure phrase pour chaque audio parmi celles générées par notre modèle.

Le chapitre 5 montre l'évolution de notre modèle vers un système plus robuste, performant et rapide. Nous améliorons la génération des phrases à l'aide de modifications de l'algorithme de recherche en faisceaux, et modernisons l'architecture de notre encodeur pour enrichir les

représentations audio. Le système obtenu sera nommé SR2, et réutilisé dans un chapitre suivant. Pour aller plus loin, nous présenterons une amélioration de l'encodeur de notre système, ainsi que l'introduction d'augmentations plus spécifiques à la DTAA. Enfin, en étudiant les différents jeux de données disponibles, nous avons également trouvé plusieurs recouvrements entre eux, ce qui peut faire surestimer les modèles de DTAA. Nous avons donc corrigé ce problème et donné son impact pour obtenir la performance réelle de notre système, qui sera nommé SR3.

Le chapitre 6 décrit l'introduction d'une méthode d'apprentissage multitâche, visant à améliorer la sémantique des phrases générées par notre modèle SR2. Nous détaillons notre stratégie et la manière dont elle se différencie des autres méthodes similaires. Nous donnons les résultats de cette stratégie sur un jeu de données et nous discuterons de son impact sur l'apprentissage. Enfin, nous proposons une autre méthode d'optimisation réalisant un impact similaire durant l'apprentissage, et nous identifions quelle méthode peut être conservée pour améliorer notre système.

Le chapitre 7 présente CoNeTTE, une variante plus généraliste du modèle SR3. En effet, les modèles entraînés sur certains jeux de données obtiennent des performances assez faibles sur des données différentes. Afin de résoudre ce problème, le système CoNeTTE propose l'utilisation d'une représentation de tâche (*task embedding*), pour essayer de modéliser différents styles de descriptions. Nous étudions d'abord comment les représentations de tâche impactent les performance du modèle, puis comment elle affecte la forme et le contenu les phrases générées.

La troisième et dernière partie propose plusieurs recherches liées à la DTAA ne visant pas directement à améliorer notre modèle, mais à étudier d'autres tâches et d'autres paradigmes d'apprentissage.

Le chapitre 8 propose une méthode utilisant un modèle de DTAA pour réaliser les tâches de RAT pour laquelle il n'a pas été conçu. Nous comparons cette méthode en terme de performance ou en taille avec les systèmes dédiés à la RAT sur deux jeux de données différents. Puis, nous essayons d'évaluer la capacité de notre modèle à discriminer certaines relations entre les événements sonores, et nous comparons notre approche à un système de RAT.

Le chapitre 9 se focalise sur l'étude d'une stratégie d'apprentissage automatique différente, nommée apprentissage semi-supervisé. Cette dernière exploite un ensemble de données non annotées, qui sont beaucoup plus faciles à collecter, pour améliorer les performances d'un modèle. En premier lieu, nous étudions plusieurs méthodes semi-supervisées dans un cas simplifié de classification audio, puis nous tentons d'appliquer l'une de ces méthodes à la DTAA.

Le chapitre 10 introduit la DTAA multilingue, en proposant de traduire automatiquement les données de l'anglais vers plusieurs autres langues. En premier lieu, nous entraînons un système pour chaque langue, et nous les comparons avec un système multilingue. Nous proposons ensuite un test français ré-annoté par un humain pour comparer les performances des systèmes français dans un contexte plus réaliste. Enfin, nous comparons les sorties d'un système anglais traduit automatiquement vers le français par rapport à un système français et un système multilingue, pour déterminer si la création d'un système entraîné sur chaque langue est nécessaire pour la DTAA.

Première partie

État de l'art

Chapitre 1

Méthodes de DTAA

Dans ce chapitre, nous présentons un état de l'art général sur les méthodes appliquées pour réaliser la DTAA. La totalité des études de la littérature emploient des réseaux de neurones dits de « bout-en-bout » (*seq2seq*), inspirés des domaines de la traduction automatique et de la reconnaissance de la parole. Les architectures sont composées d'un encodeur qui prend en entrée le fichier audio et produisent une représentation servant d'entrée au décodeur pour générer la phrase. Le décodeur prend également en entrée le ou les mots précédents de la phrase, et produit en sortie la distribution de probabilité du mot suivant.

En premier lieu, nous commençons par présenter le fonctionnement de certaines couches des réseaux de neurones en général employés en DTAA. Puis, nous décrivons les différents jeux de données publics conçus ou liés à cette tâche. Nous établissons ensuite un état de l'art des systèmes de DTAA selon leur méthode de traitement des données audio-texte et selon leur méthode d'apprentissage ou d'inférence. Enfin, nous terminons par résumer l'évolution de ces systèmes en fonction de leur taille, performance et année.

Sommaire

1.1 Contexte

Durant la dernière décennie, le domaine de l'apprentissage automatique a beaucoup progressé suite à de nombreuses évolutions des réseaux de neurones profonds (*Deep Neural Network*, DNN). Ces derniers ont notamment pris de l'ampleur en 2012, année où plusieurs DNN ont atteint l'état de l'art en vision par ordinateur [Krizhevsky 2012] et reconnaissance de la parole [Hinton 2012b]. Cela a entraîné une augmentation de leur utilisation pour d'autres types de données et de tâches, comme le traitement de texte [Mikolov 2013a]. L'amélioration de la puissance de calcul et des bibliothèques de code a facilité de plus en plus l'usage de cette technologie, menant à traiter des jeux de données plus grands, plus complexes, et même multimodaux. C'est en 2015 que la Description Textuelle Automatique d'Images (*Image Captioning*, aussi appelée Sous-titrage automatique d'images, DTAI) [Vinyals 2015] apparaît. La DTAI a pour but de créer des systèmes pouvant décrire des images automatiquement, bien souvent à l'aide de DNN. Puis en 2017 [Drossos 2017], apparaît la variante audio de cette tâche, la Description Textuelle Automatique de l'Audio (*Automated Audio Captioning*, DTAA). Les deux tâches partagent de nombreux points communs, comme certaines architectures ou métriques. Il existe aussi une variante vidéo de cette tâche (DTAV), ainsi qu'une version vidéo et audio (DTAVA). Cependant, les données de descriptions actuelles restent plutôt focalisées sur les modalités audio-texte ou image-texte.

Une compétition dédiée à la DTAA [2] a été par la suite proposée en 2020 dans le *Detection and Classification of Acoustic Scenes and Events workshop* (DCASE), qui se focalise sur le traitement automatique et l'analyse des événements et des scènes acoustiques. La compétition a grandement contribué à stimuler la recherche autour de cette thématique, en mettant en avant de nombreux systèmes et méthodes pour réaliser la DTAA.

1.2 Réseaux de neurones

La tâche de DTAA étant une tâche multimodale audio-texte, il existe plusieurs types de réseaux de neurones utilisés dans la littérature. On retrouve notamment trois types de couches : les convolutions, les cellules récurrentes et les mécanismes d'attention. Durant la phase d'apprentissage, chaque exemple est donné en entrée du réseau pour produire une sortie. Les poids du réseau de neurone sont mis à jour pour minimiser une fonction de coût entre sa sortie et l'étiquette que l'on cherche à obtenir. La mis à jour des poids se fait à l'aide de méthodes reposant essentiellement sur une descente de gradient.

1.2.1 Réseaux de neurones convolutifs

Introduit en 1998 [LeCun 1995], les réseaux de neurones convolutifs (*Convolutional Neural Network*, CNN) emploient des opérations de convolution entre une entrée et un noyau contenant des poids appris. Le schéma 1.1 montre un exemple de convolution à deux dimensions (2D) avec un seul noyau sur une entrée représentant une image couleur (avec C_{in} canaux). Le noyau parcourt l'image en longueur et largeur et réalise la somme des multiplications de chaque élément du noyau par celui correspondant dans l'image. Une illustration d'une convolution est donnée par la figure 1.1.

2. https://dcase.community/challenge2020/task-automatic-audio-captioning

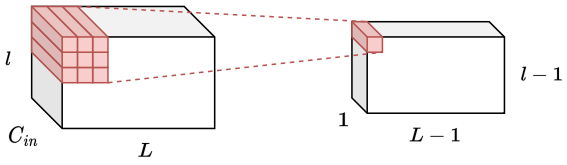


FIGURE 1.1 – Illustration d'une convolution 2D avec un noyau de taille 3×3 pour une entrée taille $L \times l \times C_{in}$.

L'utilisation de la convolution permet aux CNN d'apprendre à reconnaître des motifs dans les données indépendamment de leur position, ce qui la rend particulièrement pertinente pour la détection d'objet dans le domaine de l'image. Pour repérer plusieurs motifs différents, plusieurs noyaux peuvent être utilisés sur la même entrée pour produire C_{out} sorties. En audio, des convolutions 1D peuvent être employées sur le signal brut [Tran 2021] ou bien des convolutions 2D sur une représentation temps/fréquence du signal. Les CNN emploient également plusieurs couches de convolutions à l'intérieur de leur architecture qui permettent de repérer des motifs plus complexes.

1.2.2 Réseaux de neurones récurrents

Les réseaux de neurones récurrents (*Recurrent Neural Network*, RNN) sont un type de DNN créé pour traiter des séquences de données de longueurs variables comme le texte ou l'audio. Contrairement aux autres types de réseaux, les RNN possèdent un cycle, c'est-à-dire que les sorties d'un neurone pour l'élément t sont réutilisées en tant qu'entrée pour l'élément suivant $t+1$. Le schéma 1.2 montre le fonctionnement d'une cellule RNN.

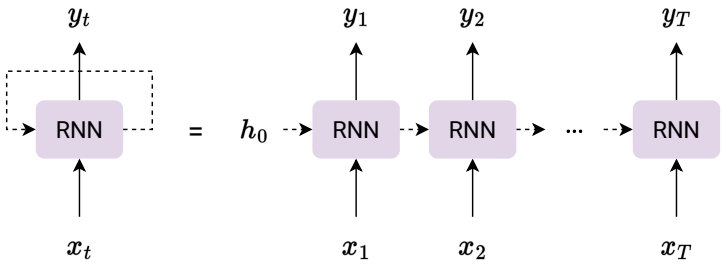


FIGURE 1.2 – Cellule RNN. La version compressée est à gauche et la version dépliée de la même cellule est à droite. L'entrée est la séquence x et la sortie est la séquence y, toutes deux de longueur T.

Ce cycle dans le réseau permet au neurone d'obtenir un contexte, une information sur les éléments précédents de la séquence pour modifier la sortie suivante. Cependant, l'apprentissage des réseaux de neurones récurrents souffre de deux problèmes : la disparition du gradient qui tend à faire oublier aux neurones l'information des éléments précédents, ce qui rend le neurone moins performant quand une dépendance lointaine dans la série intervient. Le second problème est l'explosion du gradient, qui apparaît lorsque la série est longue et que la norme des valeurs en sortie et entrée du réseau augmente.

1.2.2.1 Réseaux de neurones LSTM

Les réseaux de neurones de type *Long-Short Term Memory* (LSTM) [Hochreiter 1997] ont été créés pour limiter le phénomène d'évaporation du gradient des RNN. Les LSTM possèdent plusieurs entrées et sorties à chaque état t : x_t indique l'élément de la séquence, h_t est l'état caché de sortie du neurone et c_t est la cellule mémoire. Le schéma 1.3 résume le fonctionnement global de la cellule récurrente.

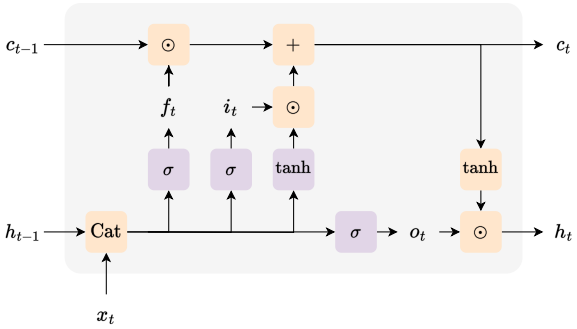


FIGURE 1.3 – Cellule LSTM. Les cases **orange** désignent les fonctions et les **violettes** des couches linéaires suivies d'une fonction d'activation. « Cat » désigne la concaténation bout-à-bout de deux vecteurs.

Les LSTM font intervenir trois fonctions nommées « portes » : f_t sert à oublier une information de la cellule c_{t-1}, i_t pondère l'information d'entrée selon x_t et h_{t-1}, et o_t contrôle l'information en sortie du neurone selon x_t et h_{t-1}. Ces portes sont définies dans l'équation 1.1.

$$\begin{aligned} f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\ i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\ o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \end{aligned} \tag{1.1}$$

σ désigne ici la fonction sigmoïde, $\odot$ le produit matriciel d'Hadamard (ou produit élément-par-élément) et tanh la tangente hyperbolique. Les poids de la cellule sont notés W, U et les biais b.

Les sorties du neurone c_t et h_t sont ensuite définies par l'équation 1.2 :

$$\begin{aligned} \tilde{c}_t &= \tanh(W_c x_t + U_c h_{t-1} + b_c) \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\ h_t &= o_t \odot \tanh(c_t) \end{aligned} \tag{1.2}$$

1.2.2.2 Réseaux de neurones GRU

Les réseaux de neurones *Gated Recurrent Unit* (GRU) [Cho 2014] sont une version simplifiée et plus rapide à calculer des LSTM. Ils n'utilisent pas la cellule mémoire c_t, possèdent moins de paramètres entraînables et appliquent une fonction d'activation non-linéaire en moins. Le schéma 1.4 résume le fonctionnement d'une cellule GRU. La fonction σ est ici une fonction logistique, qui est une généralisation de la fonction sigmoïde.

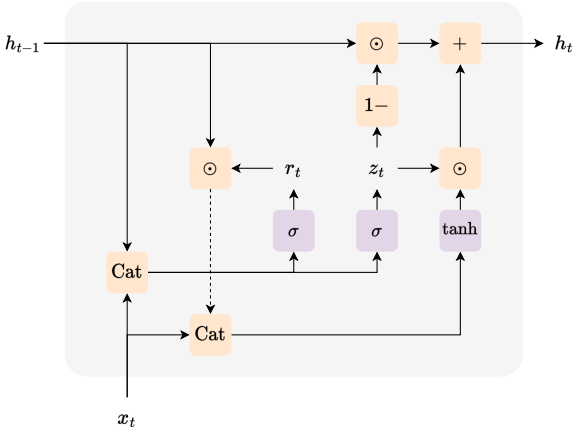


FIGURE 1.4 – Cellule GRU. Les cases **orange** désignent les fonctions et les **violettes** des couches linéaires suivies d'une fonction. « Cat » désigne la concaténation bout-à-bout de deux vecteurs.

Les cellules GRU ne font intervenir que deux portes : z_t pour la mise à jour de l'état caché et r_t pour la remise à zéro de l'état caché. Ces portes sont définies par l'équation 1.3 :

$$\begin{aligned} z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z) \\ r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r) \end{aligned} \tag{1.3}$$

La sortie du neurone h_t est définie par l'équation 1.4 :

$$\begin{aligned} \tilde{h}_t &= \tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \\ h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \end{aligned} \tag{1.4}$$

1.2.2.3 Réseaux récurrents bidirectionnels

Les RNN bidirectionnels (BiRNN) sont des réseaux dans lequel la mémoire circule dans les deux sens, ce qui permet aux neurones de se servir des informations précédentes et suivantes pour traiter l'élément en entrée. Le réseau peut ainsi modéliser des relations plus complexes dans la séquence pour produire une représentation plus riche en sortie qui dépend du contexte du futur de la séquence. Son utilisation se limite donc à la partie encodeur des réseaux, qui doivent pouvoir parcourir toute la séquence d'entrée dans les deux sens. Les cellules utilisées peuvent être des LSTM ou des GRU, et on parlera respectivement de BiLSTM et BiGRU. En pratique, une couche bidirectionnelle est composée de deux couches unidirectionnelles qui parcourent la séquence dans des ordres opposés. Le fonctionnement déroulé d'une couche bidirectionnelle est illustré par la figure 1.5.

1.2.3 Mécanismes d'attention

Les mécanismes d'attention sont des modules ajoutés à des DNN visant à aider le modèle à se focaliser uniquement sur une partie de l'entrée. Introduit à l'origine pour la traduction automatique [Bahdanau 2016], le mécanisme d'attention permet au modèle à prendre en compte qu'une partie de la phrase d'entrée pour prédire le i-ème mot en sortie à l'aide de coefficients. Les mécanismes d'attention reposent sur trois entrées : une requête Q, une clé K et une valeur

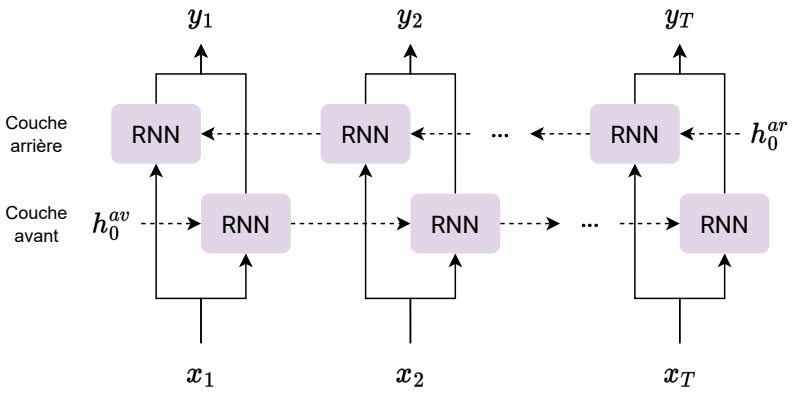


FIGURE 1.5 – Fonctionnement d'une couche BiRNN pour une entrée x et une sortie y.

V, et est décrit dans l'équation 1.5. Pour la traduction avec des RNN, la requête correspond à une représentation du ou des mot(s) précédent(s) (h_{t-1}) de la langue cible, alors que la clé et la valeur sont des projections des représentations de la phrase d'entrée de la langue source.

$$\text{Attn}(Q, K, V) = \text{softmax}\left(QK^T\right)V \tag{1.5}$$

Le module d'attention à produit scalaire échelonné (*Scaled Dot-Product Attention*, SDPA) décrit par l'équation 1.6 est une version légèrement modifiée du mécanisme d'attention. Il divise par la racine carrée de la dimension d_k des matrices Q et K pour éviter que le produit QK^T n'ait une variance trop élevée. Elle est principalement utilisée dans les architectures *Transformer* [Vaswani 2017], qui seront détaillées plus tard dans le chapitre 3.

$$\text{SDPA}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1.6}$$

Pour la DTAA, les modules d'attention seront utilisés notamment pour permettre à la sortie correspondant au mot w_i de se focaliser sur une partie de l'audio d'entrée, et également sur une partie des mots précédents $w_{0..i-1}$.

1.2.4 Entropie croisée

Pour entraîner un système de DTAA, la plupart des approches utilisent des réseaux entraînés avec la fonction d'entropie croisée (*Cross Entropy*, CE), décrite dans 1.7.

$$\text{CE}(p, y) = -\frac{1}{|y|}\sum_{i=1}^{|y|}\sum_{w\in V} y_{i,w}\log p_{i,w} \tag{1.7}$$

avec p l'ensemble des probabilités des mots suivants prédits par le modèle et $y_{i,w}$ la valeur binaire indiquant si le mot w du vocabulaire V en i-ème position est présent ou non.

Bien que cette fonction soit très utilisée pour la génération de texte, elle tend à générer un contenu textuel répétitif, peu divers [Holtzman 2020a] et ne prend pas en compte les synonymes, les différentes structures ou la proximité sémantique des phrases. La fonction CE est aussi très utilisée pour évaluer un modèle de DTAA sur les sous-ensembles de validation, pour sélectionner la meilleure époque d'entraînement. Dans ce cas, on ne monitore donc pas exactement la capacité du modèle à générer une nouvelle description.

1.2.5 Optimisation

Durant la phase d'apprentissage, l'objectif est de modifier les poids du réseau de neurones pour minimiser une fonction de coût à l'aide de méthode d'optimisations. En DTAA, seuls deux algorithmes d'optimisation différents sont utilisés pour entraîner les DNN : *ADAptive Moment estimation* (Adam) [Kingma 2015] et *ADAptive Moment estimation with decoupled Weight decay* (AdamW) [Loshchilov 2017]. L'optimiseur Adam est un algorithme de descente de gradient qui a pour particularité d'ajuster le pas d'apprentissage de manière dynamique de chaque paramètre. Pour cela, l'algorithme calcule une moyenne continue exponentielle (*Exponential Moving Average*, EMA) des gradients g_t et de leurs carrés g_t^2. L'EMA sert à lisser les fluctuations durant l'entraînement (comme l'optimiseur Momentum [Qian 1999]) et à ajuster le pas d'apprentissage (comme l'optimiseur RMSProp [Hinton 2012a]). Le fonctionnement d'Adam est décrit par les équations 1.8.

$$g_t = \nabla_\theta f(\theta_{t-1}) \tag{1.8a}$$
$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{1.8b}$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \tag{1.8c}$$
$$\hat{m}_t = m_t / (1-\beta_1^t) \tag{1.8d}$$
$$\hat{v}_t = v_t / (1-\beta_2^t) \tag{1.8e}$$
$$\theta_t = \theta_{t-1} - \gamma \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon) \tag{1.8f}$$

avec θ_t les paramètres du modèle à l'itération t, g_t le gradient à l'itération t, γ le pas d'apprentissage global de l'optimiseur et ϵ une petite valeur positive pour éviter une division par zéro lors des calculs numériques. Les variables m_t et v_t sont respectivement des estimations des premiers et seconds moments du gradient (moyenne et variance). Enfin, les hyperparamètres β_1 et β_2 régulent la mise à jour des moments. Au début de l'entraînement, l'estimation des moments est proche de zéro (car $m_0 = v_0 = 0$), ce qui peut perturber l'apprentissage. Pour éviter ce problème, Adam emploie une correction de l'estimation des moments en divisant leur valeur par $(1-\beta_i^t)$, ce qui fait croître leur norme quand t est petit.

La plupart des implémentations d'Adam intègrent également une méthode de régularisation au travers de la pénalisation de la norme ℓ_2 des poids du réseau. L'objectif de cette méthode est de minimiser la valeur des poids du réseau, ce qui évite d'obtenir des poids élevés qui généralisent moins bien et provoquent une plus grande variance en sortie du réseau [Geman 1992, Krogh 1991]. En pratique, cette pénalisation est implémentée pour Adam en modifiant l'initialisation du gradient avant l'estimation des moments, en modifiant l'équation 1.8a par 1.9. L'hyperparamètre λ_{wd} est appelé par abus de langage *weight decay*, car la pénalisation de la norme ℓ_2 et le *weight decay* sont équivalents pour certains optimiseurs, mais ce n'est pas le cas pour Adam.

$$g_t = \nabla_\theta f(\theta_{t-1}) + \lambda_{\text{wd}} \theta_{t-1} \tag{1.9}$$

Pour mieux intégrer la régularisation, les auteurs de l'étude [Loshchilov 2017] ont proposé une variante nommée AdamW. Dans cette dernière, la pénalisation se fait sur les poids θ_t directement plutôt que sur le gradient, en remplaçant l'étape 1.8f par 1.10.

$$\theta_t = \theta_{t-1} - \gamma \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon) - \lambda_{\text{wd}} \theta_{t-1} \tag{1.10}$$

Bien que plus efficace qu'Adam, AdamW reste moins utilisé en DTAA, avec seulement 23 systèmes qui utilisent AdamW comparé à 40 qui utilisent Adam sur 77 systèmes que nous avons glanés.

Cependant, AdamW a eu un impact assez important sur notre modèle, que nous détaillerons dans le chapitre 6.

1.3 Jeux de données audio-texte

Pour entraîner les différents DNN présentés ci-dessus, il est nécessaire d'exploiter de large jeux de données annotés. Les quatre principaux utilisés en DTAA sont : AudioCaps, Clotho, MACS et WavCaps. Ils diffèrent sur de nombreux aspects, comme leur taille, leurs sources des fichiers audio, leur processus d'annotation ou encore leurs différents sous-ensembles. Les fichiers audio des jeux de données de DTAA sont souvent collectés sur divers sites de stockage de fichiers audio comme YouTube [3], Freesound [4] ou encore SoundBible [5]. Dans certains cas, ces fichiers sont annotés par des humains qui doivent écouter, puis décrire le fichier audio en question en se focalisant sur les événements sonores. Certains processus d'annotation exploitent également la description d'origine de l'utilisateur qui l'a posté sur le site. Selon la source, des métadonnées supplémentaires peuvent être récupérées automatiquement, comme des étiquettes ou des classes de fichiers audio. Plusieurs jeux de données de DTAA sont également des sous-ensembles ou contiennent des fichiers communs avec d'autres jeux de données de classification d'événements sonores ou de scènes, comme AudioSet (AS) [Gemmeke 2017] ou FSD50K [Fonseca 2021], qui contiennent respectivement 2 millions et 50 mille fichiers audio de 10 secondes. Les auteurs de certaines études [Yuan 2021] vont parfois récupérer eux-mêmes des données de ces sites web pour agrandir la quantité de données d'entraînement, mais ne publient pas toujours la liste des fichiers récupérés.

1.3.1 Jeux de données de description audio

1.3.1.1 AudioCaps

AudioCaps (AC) [Kim 2019a] est un jeu de données contenant originellement 51 308 fichiers audio issus de vidéos postées sur YouTube. Plus précisément, les audios sont un sous-ensemble d'AudioSet (AS), ce qui permet d'exploiter également les classes d'événements sonores. Ce jeu de données a été annoté à l'aide de l'*Amazon Mechanical Turk* [6], un service facilitant l'annotation de données par des utilisateurs pour créer des jeux de données. AC est composé de trois sous-ensembles distincts pour l'apprentissage (AC-train), la validation (AC-val) et le test (AC-test), contenant respectivement 49 838, 495 et 975 fichiers audio. Chaque audio n'est associé qu'à une seule référence pour le jeu d'entraînement, et cinq pour les jeux de validation et de test. La majorité des fichiers durent dix secondes, et contiennent les principaux événements sonores d'AS, à l'exception de ceux considérés comme trop rares (<1000 occurrences), contenant de la musique, ou liés à du contenu visuel (annoté par la classe *« Inside small room »* par exemple). Au total, les phrases ont été écrites par 108 annotateurs anglophones. Les sous-ensembles de validation et de test ont été corrigés par trois examinateurs. Des exemples de descriptions issus du sous-ensemble de validation sont disponibles dans le tableau 1.1 pour un fichier audio d'AC-val [7].

3. https://youtube.com/
4. https://freesound.org/
5. https://soundbible.com/
6. https://www.mturk.com/
7. https://www.youtube.com/embed/yBksF4L5Ics?start=30&end=40

TABLE 1.1 – Descriptions du fichier « `yBksF4L5Ics` », issu d'AC-val.

Descriptions
A woman speaks and a child cries
A beep occurs then a woman and child speak
A grown woman speaks and a younger girl speaks
Female speech followed by a child whining
A woman speaks and a child whines

Pour décrire chaque fichier audio, les annotateurs humains ont eu accès à l'audio d'origine, aux étiquettes d'événements sonores et à la vidéo d'origine de YouTube dans le cas où l'audio ne suffisait pas. L'accès à ces métadonnées impacte cependant certaines descriptions qui se retrouvent à être moins diversifiées (comme *« A man speaks »*) et qui contiennent parfois certains éléments exclusivement visuels.

Cependant, les fichiers audio d'AC n'ont pas été enregistrés en 2019, car ils ne sont pas tous disponibles sous une licence libre, et doivent donc être ré-extraits depuis YouTube. Certaines vidéos sont supprimées par leur auteur ou par les administrateurs de YouTube, ce qui réduit le nombre de fichiers disponibles au cours des années, y compris pour le sous-ensemble de test qui sert à comparer les systèmes.

1.3.1.2 Clotho

Clotho (CL) [Drossos 2020] est un autre jeu de données sorti peu après AudioCaps, et contenant 6 974 fichiers audio collectés depuis le site Freesound. Contrairement à AudioCaps, la taille des fichiers audio est plus longue et varie entre 15 et 30 secondes. À noter que les auteurs ont tronqué la durée des fichiers audio d'origine pour obtenir une distribution uniforme. Chaque fichier audio du jeu de données est décrit par cinq phrases différentes. Les données audio et texte de CL peuvent être facilement téléchargées depuis le site Zenodo [8].

Dans sa première version, le jeu de données était divisé en deux parties : *development* (CL-dev, pour l'entraînement) et *evaluation* (CL-eval, pour le test). Par la suite, le sous-ensemble d'entraînement a été agrandi à 3 839 fichiers et un nouvel ensemble de 1 045 fichiers de validation (CL-val) a été ajouté pour la seconde version du jeu de données en 2021. Ce jeu de données est au cœur de la tâche 6 de la compétition DCASE. Certains participants de la compétition ont utilisé le sous-ensemble de validation comme données d'entraînement et le sous-ensemble d'évaluation comme une validation, ce qui biaise certains résultats. Il existe également d'autres sous-ensembles proposés pour la compétition, nommés *test* et *analysis*. Ils sont utilisés pour classer les systèmes de la compétition et seuls les fichiers audio ont été partagés pour ces deux derniers sous-ensembles. Dans cette thèse, nous utiliserons la convention de nommage du jeu de données d'origine, qui diffère légèrement de celui de la compétition DCASE. En effet, Les sous-ensembles *development*, *validation* et *evaluation* sont respectivement appelés *development-training*, *development-validation* et *development-testing* pour la compétition.

8. https://zenodo.org/record/3490684

Contrairement à AudioCaps, les auteurs de CL adoptent une stratégie d'annotation en trois étapes : la première consiste à demander aux annotateurs de décrire des fichiers audio, sans aucune autre information supplémentaire, pour éviter tout biais. Dans la seconde étape, d'autres annotateurs corrigent les erreurs typographiques, de grammaire ou même reformulent la description si nécessaire. Enfin, pour la troisième étape, un troisième ensemble d'annotateurs évalue les descriptions de chaque audio par deux scores entiers de 1 à 4 (4 étant le meilleur). Ces deux scores représentent respectivement la précision et la fluence de la description, et permettent de classer les descriptions entre elles pour de ne récupérer que les cinq meilleures de chaque audio. Malgré tout ce processus d'annotation, les auteurs ont rajouté une quatrième étape pour homogénéiser certains termes (par exemple : *« nonstop »*, *« non stop »* et *« non-stop »*), enlever quelques éléments liés de la parole (comme la langue prononcée), supprimer les entités nommées et corriger les quelques phrases qui contenaient encore de la transcription de parole. Des exemples de descriptions sont donnés dans le tableau 1.2 pour un fichier audio de CL-dev [9].

TABLE 1.2 – Descriptions du fichier « `fdv_orage_26082011.wav` », issu de CL-dev.

Descriptions
Rain falling at a moderate pace as thunder chimes in the background.
Rain is falling at a moderate pace and thunder rumbles in the background.
Rain is splattering on a hard surface continually and thunder bellows out.
Rain is splattering on a hard surface continually and thunder rolls in the distance.
Steady rain is falling with a light rumble of thunder in the background.

Les différents sous-ensembles de CL ont été conçus selon plusieurs critères : un partage de 60% / 20% / 20% des fichiers audio, fait de sorte que tous les mots de la validation et du test sont présents au moins une fois dans la partie entraînement. Ils ont également forcé une distribution de mots similaire entre les trois différents sous-ensembles. À noter que durant nos expériences, nous avons remarqué quelques erreurs dans les métadonnées. En effet, les liens des sources Freesound d'où sont extrait certains fichiers sont manquants et quelques fichiers ont des sources communes entre les sous-ensembles *development*, *validation* et *evaluation*. Les taux de recouvrement de CL entre ses propres sous-ensembles et ceux d'autres jeux de données sont donnés dans l'annexe I.

1.3.1.3 *Multi-Annotator Captioned Soundscapes*

Multi-Annotator captioned soundscapes (noté ici MA, mais parfois nommé MACS dans la littérature) [Martin 2021] est un jeu de données de 3 930 fichiers audio provenant du jeu de données TAU Urban Acoustic Scenes 2019 [Mesaros 2018]. Ce dernier est un jeu de données pour la classification de dix différentes scènes acoustiques (aéroport, centre commercial, station de métro, rue piétonne, place publique, rue avec du trafic moyen, tram, bus, métro et parc). MA ne contient pas de sous-ensemble de validation ou de test, et il est utilisé exclusivement en tant que jeu de données d'entraînement additionnel. Comme pour Clotho, les données sont librement disponibles sur le site Zenodo [10].

Durant le processus d'annotation de MA, chaque humain donne une étiquette d'événement sonore puis décrit par une phrase le fichier audio. D'après les auteurs, cette méthode d'annotation

9. https://freesound.org/people/FaireDesVagues/sounds/127269
10. https://zenodo.org/record/5114771

semble biaiser les descriptions, en incitant les annotateurs à utiliser les noms des classes des événements dans les phrases elles-mêmes. Des exemples de descriptions sont donnés dans le tableau 1.3. MA fourni également des métadonnées contenant le score de chaque annotateur humain, ce qui permet d'estimer de la qualité de ses annotations. Cependant, ces scores ne semblent cependant pas avoir été exploités dans la littérature.

TABLE 1.3 – Descriptions du fichier « `airport-barcelona-0-13-a.wav` ».

Descriptions
A lot of people are talking and walking around
People talking and walking by
People passing by and talking
Footsteps and adults talking also a random thump and quick shimmering sound
Adults talking and some footsteps coming across

1.3.1.4 WavCaps

WavCaps (WC) [Mei 2023] est l'un des plus grands jeux de données dédié à la DTAA à ce jour. Il contient 403 050 paires d'audio-texte provenant de quatre sources différentes : une partie fortement annotée d'AudioSet (AS) et trois sites web : BBC Sound Effects (BBC), Freesound (FSD) et SoundBible (SB). Pour filtrer les fichiers audio et descriptions brutes de ces trois sites, les auteurs enlèvent les fichiers audio trop courts (inférieurs à une seconde) ainsi que les descriptions trop communes. Cependant, les paires qui en résultent ne contiennent pas toujours de phrase correcte et ne focalisent pas toujours sur les événements sonores, ce qui pousse à rajouter un traitement supplémentaire. Les descriptions de WC sont donc générées par un modèle ChatGPT [11], à partir des métadonnées de chaque fichier audio. Pour la partie issue d'AS, les métadonnées utilisées sont les noms des classes d'événements sonores dans leur ordre d'apparition. Pour les autres sous-ensembles, les métadonnées utilisées sont les descriptions brutes données par les utilisateurs qui ont posté l'audio sur les sites d'origine. Malgré la requête donnée à ChatGPT, il reste tout de même régulièrement des informations non liées aux événements sonores dans les descriptions, comme de la parole ou des informations visuelles. Quelques exemples de phrases traitées par ChatGPT sont donnés dans les tableaux 1.4 et 1.5, pour deux fichiers audio de la partie WC-SB [12] [13].

TABLE 1.4 – Descriptions originale et finale pour le fichier « `Kid Says Hey 4x` », issu de WC-SB.

Description d'origine
My son doing some vocal work for our site. He is 4 years old and is saying hey 4 times.
Description traitée par ChatGPT
Someone is saying "hey".

11. https://openai.com/blog/chatgpt/
12. https://soundbible.com/2013-Kid-Says-Hey-4x.html
13. https://soundbible.com/2005-Elaborate-Thunder.html

TABLE 1.5 – Descriptions originale et finale pour le fichier « `Elaborate Thunder` », issu de WC-SB.

Description d'origine
This clip details a few different thunder strike sounds. One even has that type of thunder that strikes and then you hear it echo back the original clap but only backwards.
Description traitée par ChatGPT
Thunder is striking in different ways.

1.3.1.5 Résumé des jeux de données de DTAA

Pour mieux visualiser ces différents jeux de données, nous présentons dans le tableau 1.6 quelques statistiques générales sur les données audio et texte contenues dans les sous-ensembles d'entraînements. On y observe de larges variations des longueurs des fichiers audio, de 0,5 seconde à plusieurs heures, mais aussi dans le texte avec des longueurs de phrase de 2 à 52 mots. La diversité du vocabulaire et la longueur moyenne des phrases sont également très variables, ce qui semble indiquer que les différents processus d'annotation ont mené à plusieurs types de descriptions entre les quatre jeux. Il est cependant difficile de déterminer la distribution des événements sonores présente dans chaque jeu, car tous ne disposent pas des métadonnées nécessaires à cette analyse.

TABLE 1.6 – Statistiques globales des sous-ensembles d'entraînement de chaque jeu de données.

Jeu de données (acronyme)	**AudioCaps (AC)**	**Clotho (CL)**	**MACS (MA)**	**WavCaps (WC)**
Fréq. d'échantillonnage (Hz)	32 000	44 100	48 000	32 000
Durée par audio min-max (s)	0,5-10	15-30	10	1-67 109
Nb. audio	49 838	3839	3930	403 050
Durée totale (h)	136,6	24,0	10,9	7563,3
Vocabulaire	4724	4369	2721	24 600
Nb. de mots	402 482	217 360	159 879	3 161 823
Nb. de mots par phrase min-max	2-52	8-20	2-40	2-38
Nb. de mots moyens par phrase	8,7	11,3	9,3	7,8
Nb. de phrases par audio	1	5	2-5	1

1.3.2 Autres jeux de données audio-texte

Il existe d'autres jeux de données audio-texte qui ont été ou qui pourraient être exploités pour la DTAA. Ils n'ont cependant pas été étudiés dans cette thèse, à cause de la langue, de la qualité, du caractère audiovisuel des descriptions, ou simplement de la date de création du jeu de données.

1.3.2.1 AudioCaption

AudioCaption [Wu 2019, Xu 2021b] est un autre jeu de données dédié à la DTAA, dont des descriptions sont écrites en mandarin. Les enregistrements audio ont été faits dans deux types de lieux : des hôpitaux et des voitures, avec respectivement 3 710 et 3 602 fichiers. Ce jeu de données est peu utilisé dans la littérature, notamment à cause de la langue utilisée pour les descriptions et de la faible diversité des événements sonores présents par rapport aux autres jeux de données.

1.3.2.2 SoundDescs

SoundDescs (SDs) [Oncescu 2021] contient 32 979 échantillons audio issus du site BBC. La durée des fichiers audio est plus élevée que bien d'autres jeux de données, avec une moyenne de 115 secondes et un maximum de 4475 secondes. Contrairement à d'autres jeux de données, les descriptions n'ont pas été post-traitées après leur téléchargement, ce qui laisse des erreurs ou des descriptions contenant des informations non liées aux événements sonores comme *« medium close-up shrill, staccato calls from kingfisher. medium distance to background calls from other bird species. N.B. Slight distortion, as recorded. »*.

1.3.2.3 WavText5k

WavText5k (WT) [Deshmukh 2022] est un jeu de données de 4525 fichiers audio téléchargés depuis les sites web SoundBible et BigSoundBank. Les phrases sont issues des descriptions brutes données par les utilisateurs des sites correspondants. Comme pour SoundDescs, les descriptions n'ont pas été corrigées, ce qui rend ce jeu moins intéressant pour la DTAA et laisse des phrases comme *« Soundscape taken at 1 am in Paris at the second floor balcony of an apartment. Place Saint Augustin. »*.

1.3.2.4 LAION-Audio-630K

Le jeu de données LAION-Audio-630K [Wu 2023c] est une concaténation de huit autres jeux de données audio étiquetés avec du texte. Les fichiers audio sont issus de divers sites web : BBC sound effects, Freesound, Epidemic Sound, Audiostock, Free To Use Sounds, Sonniss Game Effects, We Sound Effects et Paramount Motion Sound Effects. Il contient également de nombreux autres jeux de données d'événements sonores, comme Clotho, AudioCaps ou AudioSet. Pour créer une description du fichier audio, les étiquettes des événements sonores sont simplement concaténées pour former une phrase. Par exemple, un fichier audio d'AudioSet contenant les étiquettes *« Male speech »*, *« Birds singing »* et *« Rain »* donnera la phrase *« The sounds of male speech, birds singing and rain. »*. Ce traitement limite beaucoup la diversité des phrases de ce jeu de données, et le rend plutôt destiné à des tâches comme la recherche audio-texte ou pour l'apprentissage de représentations audio-texte.

1.3.2.5 Auto-ACD

Auto-ACD [Sun 2023b] est un nouveau jeu de données récent de 1,9 million de paires audio-texte issues d'AudioSet et de VGGSound [Chen 2020]. Comme pour WC, les descriptions sont générées par ChatGPT, qui prend en entrée les étiquettes des jeux d'origines ou les prédictions faites par d'autres modèles de classification. Les auteurs ont également donné des informations visuelles à ChatGPT pour générer une description plus riche, tout en ajoutant au *prompt* de supprimer toute information inaudible. Contrairement aux autres jeux de données mentionnés

dans cette section, Auto-ACD propose en plus un sous-ensemble d'évaluation de mille fichiers audio manuellement filtrés et corrigés par des humains. Ce jeu de données n'a pas été utilisé car il a été publié vers la fin de ma thèse.

1.3.2.6 MusicCaps

Le récent jeu de données MusicCaps [Agostinelli 2023] contient 5521 pairs de musiques et descriptions écrites par des musiciens. Les fichiers audio sont des enregistrements de dix secondes issus d'AudioSet, annotés par plusieurs descriptions et par des étiquettes indiquant le genre, le tempo ou encore le rythme de la musique. Ce jeu de données n'est pas étudié dans ce manuscrit car il est relativement petit et a été rendu public vers la fin de ma thèse.

1.3.2.7 Song Describer Dataset

Encore plus récemment, les auteurs de [Manco 2023] ont proposé le *Song Describer Dataset*, qui contient 706 fichiers audio de 2 minutes provenant du jeu de classification de genres musicaux MTG-Jamendo Dataset [Bogdanov 2019]. Les descriptions contiennent des informations riches sur la musique avec une moyenne de 21 mots par phrase, mais n'ont pas été annotées spécifiquement par des musiciens. Elles ont été relues et corrigées maanuellement par les créateurs du jeu de données. Comme pour MusicCaps, ce jeu n'est pas étudié dans ce manuscrit car il est disponible depuis novembre 2023.

1.3.2.8 MusicTextClips

MusicTextClips [McKee 2023] est un troisième jeu focalisé sur la musique proposé en 2023 et contenant 4 169 enregistrements audio et vidéo dont 3 169 d'entraînement et 1000 de test. Ces données sont issues de YouTube8M [Abu-El-Haija 2016] qui contient des extraits de 10 secondes. Cependant, le caractère visuel des annotations rend ce jeu inexploitable pour la DTAA directement.

1.3.2.9 VALOR-1M

VALOR-1M [Chen 2023b] est un jeu de données contenant des descriptions des vidéos et audios provenant d'AudioSet. Contrairement à d'autres jeux de données de cette taille, chaque paire audio-vidéo a été annotée et corrigée par des humains manuellement pour assurer une bonne qualité dans les descriptions. Cependant, ces descriptions contiennent des informations visuelles inaudibles, ce qui le rend inapproprié pour la DTAA.

1.3.2.10 VAST-27M

Enfin, le jeu de données VAST-27M [Chen 2023c] est un sous-ensemble du jeu HD-VILA-100M [Xue 2022], contenant 27 millions de descriptions audio-vidéos générées automatiquement par des modèles de DTAI et DTAA entraînés sur VALOR-1M et WavCaps. Le modèle de génération de texte Vicuna-13b [Chiang 2023] prend la description automatique de l'audio, de la vidéo et la description manuelle d'origine du fichier pour créer une unique description audiovisuelle riche. Comme pour le jeu de données VALOR-1M, ce jeu de données ne peut être utilisé pour la DTAA car les phrases décrivent parfois des éléments uniquement visuels.

1.4 Systèmes de DTAA

Bien que les approches de DTAA utilisent toutes des DNN, elles se différencient sur certains aspects. On retrouve plusieurs manières de traiter les données audio et texte, avec l'utilisation de divers modèles pré-entraînés pour l'une ou les deux modalités. L'immense majorité des systèmes sont entraînés de manière supervisées. Certaines études se focalisent plutôt sur la fonction de coût du modèle, en remplaçant ou complétant la CE par de l'apprentissage par renforcement ou en ajoutant un second terme pour aider la convergence ou généralisation du modèle. Quelques augmentations sont également utilisées en DTAA pour aider à limiter le surapprentissage. Enfin, d'autres systèmes se concentrent plutôt sur la méthode d'inférence, en modifiant l'algorithme de génération de phrase du modèle pour obtenir de meilleures performances.

1.4.1 Représentations des données

Comme la tâche de DTAA est concerne deux modalités, on retrouve les approches temps-fréquences issues du traitement du signal, mais aussi quelques approches provenant du domaine du traitement automatique de texte, visant à utiliser d'autres unités lexicales que les mots ou les caractères. Beaucoup de systèmes combinent ces représentations avec des modèles pré-entraînés sur l'une des modalités sur de plus large corpus, pour permettre au modèle de mieux repérer les événements sonores ou mieux générer une phrase correcte.

1.4.1.1 Extraction de caractéristiques audio

La majorité des systèmes prend en entrée une représentation fréquentielle de la totalité du signal, qui repose sur la Transformée de Fourier Discrète (TFD). Cette dernière est définie par l'équation 1.11 pour un signal x de longueur N :

$$\mathrm{TFD}(k) = \frac{1}{N} \sum_{n=1}^{N} x(n) \exp\left(-2i\pi k \frac{n}{N}\right) \tag{1.11}$$

Quelques premiers travaux en DTAA [Ikawa 2019, Kim 2019a] ont utilisé des coefficients cepstraux en fréquences Mel (*Mel-frequency cepstral coefficients*) qui représentent la transformée en cosinus discrète (*Discrete Cosine Transform*) du logarithme des spectres des fenêtres du signal, sur une échelle Mel. Cependant, l'immense majorité des travaux utilisent des log-Mel spectrogrammes, qui sont une représentation plus précise du signal que les coefficients cepstraux en fréquences Mel, car elle n'applique pas la transformée en cosinus qui compresse l'information. Le schéma 1.6 résume les opérations nécessaires à la création du log-Mel spectrogramme.

FIGURE 1.6 – Illustration du processus de création du log-Mel spectrogramme depuis un signal brut.

Certaines études [Perez-Castanos 2020, Kim 2021] ont également utilisé des gammatone-spectrogrammes, qui utilisent une échelle gammatone qui est légèrement plus robuste aux bruits

que les log-Mel spectrogrammes. Seul le modèle WaveTransformer [Tran 2021] semble utiliser deux représentations de l'audio en parallèle (signal brut et log-Mel spectrogramme) pour exploiter des informations fréquentielles issues du spectrogramme, mais aussi des informations plus précises issues du signal brut.

1.4.1.2 Représentation du texte

Pour traiter le texte, la totalité des systèmes met les mots en minuscules et supprime la ponctuation, car il n'apporte pas d'information sémantique pour l'audio. La majorité des systèmes découpe (*tokenize*) les phrases en séquence de mots, ce qui permet de représenter les phrases en séquences relativement courtes par rapport à une découpe en caractères. Quelques travaux utilisent d'autres types de sous-unités, comme les unités *Byte-Pair Encoding* (BPE) [Gage 1994, Sennrich 2015] ou les unités *WordPiece* [Wu 2016].

BPE est à l'origine un algorithme de compression sans perte qui consiste à trouver la paire d'octets la plus présente pour la remplacer par une valeur d'un octet non utilisé. Pour le texte, cela consiste à démarrer en utilisant les caractères comme unités lexicales (*tokens*), puis fusionner à chaque itération la paire d'unités lexicales la plus fréquente jusqu'à atteindre la taille du vocabulaire recherchée. *WordPiece* est une variante du BPE dans laquelle on sélectionne la paire d'unités lexicales qui permet de compresser au maximum l'information de la séquence, plutôt que la plus fréquence. Ces unités permettent d'unifier la représentation de plusieurs mots similaires (comme *« ticking »*, *« ticks »* qui deviennent respectivement *« tick-ing »* et *« tick-s »*). Elles permettent également de traiter de nouveaux mots qui n'ont pas été aperçus pendant l'entraînement. De plus, la distribution des fréquences des unités utilisées est parfois plus uniforme que celle des mots bruts, ce qui peut aider l'apprentissage du modèle de langage.

1.4.1.3 Apprentissage par transfert pour l'audio

Pour surmonter le manque de données publiques disponibles pour une tâche complexe comme la DTAA, une importante majorité des systèmes utilisent de l'apprentissage par transfert. Cette technique consiste à exploiter l'apprentissage d'un modèle d'une tâche source vers une ou plusieurs autres tâches cibles. La tâche d'origine peut différer de plusieurs manières de celle(s) ciblée(s), en ayant par exemple des distributions de données différentes ou des étiquettes de nature différente. Dans le cas où la totalité des poids réutilisés sont en amont du réseau et gelés, on parlera plutôt d'extraction de représentations.

En DTAA, la principale méthode d'apprentissage par transfert consiste en premier lieu à entraîner un modèle sur un large jeu de données de classification audio (AT) comme AudioSet. Puis dans un second temps, une partie des poids de ce modèle sont réutilisés pour construire un second modèle capable de générer une phrase 1.7. Les poids réutilisés formeront la partie encodeur du système de DTAA, et pourront être gelés ou non. Dans la plupart des cas, un encodeur audio plus efficace en AT le sera également pour la DTAA.

La plupart des encodeurs récupérés pour la DTAA utilise des CNN comme VGGish [Hershey 2017], YAMNet [Plakal 2020] ou ESResNeXt [Guzhov 2021], mais la majorité exploite l'un des modèles de l'étude *Pretrained Audio Neural Network* (PANN) [Kong 2020]. PANN propose un ensemble de divers modèles entraînés sur AudioSet, composés de différentes architectures CNN

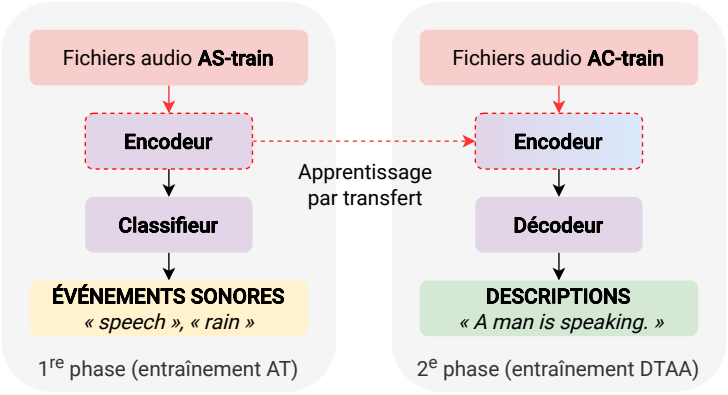


FIGURE 1.7 – Illustration de l'apprentissage par transfert / extraction de représentations le plus utilisé en DTAA.

qui diffèrent par leur représentation du signal, leur taille et leur performance. On y retrouve notamment les modèles CNN10, CNN14, ResNet38, MobileNetV2 et Wavegram-Logmel-CNN, très utilisés en DTAA [Kim 2021, Liu 2021, Mei 2021a, Liu 2022a, Koh 2022, Sun 2023a, Xiao 2023]. C'est notamment dû à la disponibilité des poids [14] et du code source [15].

L'état de l'art en classification audio évoluant rapidement depuis PANN, des encodeurs plus performants apparaissent régulièrement, fondés sur des architectures plus récentes, comme ConvNeXt que nous avons adapté à l'audio [Pellegrini 2023]. Nous pouvons également retrouver des modèles de type *Transformer*, comme l'*Audio Captioning Transformer* (ACT) [Mei 2021c], qui propose une architecture classique, mais spécifiquement entraîné pour la DTAA. Bien que les résultats des *Transformer* soient souvent comparables à ceux d'un CNN, plusieurs études comme [Tran 2021] ont par la suite commencé à utiliser des encodeurs de type *Transformer*, en partant du principe que ce type de réseau modélise mieux les dépendances temporelles qu'un CNN et mieux les dépendances à long terme qu'un RNN.

Le modèle *Patchout faSt Spectrogram Transformer* (PaSST) [Koutini 2022] utilisé par [Kouzelis 2022] est inspiré d'un modèle du domaine de la vision [Dosovitskiy 2021]. Il utilise la méthode *Patchout* qui extrait du spectrogramme des morceaux de 16×16 pixels, auquel seront rajoutés des représentations positionnelles qui formeront une séquence. Certains morceaux peuvent être supprimés aléatoirement pour diminuer la taille de la séquence et limiter le surapprentissage.

L'état de l'art sur AC [Mei 2023] utilise un autre encodeur de type *Transformer* pré-entraîné nommé le *Hierachical Token-Semantic Audio Transformer* (HTSAT) [Chen 2022b], qui obtient une meilleure performance pour l'AT que les modèles de PANN. Toutefois, cet encodeur semble être performant principalement pour les fichiers audio de dix secondes, car le modèle CNN14 s'est révélé être plus efficace que HTSAT pour la DTAA sur CL dans cette même étude. Plus récemment, l'état de l'art sur CL [Wu 2023b] utilise un nouvel encodeur pré-entraîné pour l'AT nommé le *Bidirectional Encoder representation from Audio Transformers* (BEATs) [Chen 2022c]. Contrairement aux précédents encodeurs, celui-ci est entraîné de manière autosupervisée en

14. https://zenodo.org/record/3576403
15. https://github.com/qiuqiangkong/audioset_tagging_cnn

essayant de discrétiser les trames audios. Au moment de l'écriture de ce manuscrit, BEATs atteint le meilleur score en classification audio sur AudioSet pour un unique modèle.

Une autre manière de pré-entraîner un encodeur consiste à utiliser un apprentissage contrastif entre des paires audio-texte. Cela permet à l'encodeur de représenter plus d'informations que les simples classes d'événements sonores d'AudioSet, mais cela rend le pré-entraînement plus complexe et coûteux. Ces systèmes peuvent être pré-entraînés sur des jeux de données de DTAA, mais aussi sur des jeux de données ne contenant que des mots clés, sans phrase syntaxiquement correcte. Le jeu de données LAION-Audio-630K [Wu* 2023d] a par exemple été utilisé pour créer le modèle *Contrastive Language-Audio Pretraining* (CLAP) [16]. Son encodeur audio est fondé sur une architecture de type *Transformer* et a été utilisé pour créer quelques systèmes de DTAA, comme [Emmanouilidou 2023, Schaumlöffel 2023].

Enfin, un dernier type de modèles utilisés pour l'apprentissage par transfert est le modèle de détection d'événements sonores [Xie 2023]. L'objectif est ici d'obtenir une explicite information sur la temporalité des événements, pour mieux décrire leurs relations (séquence ou superposition) dans la phrase générée, par rapport à de « simples » modèles de classification. Cependant, peu de systèmes utilisent ces modèles, car ils n'ont pas un impact positif sur les métriques classiques de DTAA.

1.4.1.4 Apprentissage par transfert pour le texte

Même si la tâche de DTAA est avant tout focalisée sur la reconnaissance des événements sonores, la génération d'une phrase correcte et riche peut également être importante. Pour cela, plusieurs systèmes [Xu 2020a, Eren 2020b] emploient des représentations de mots pré-entraînés comme Word2Vec [Mikolov 2013b] pour accélérer l'entraînement du décodeur. Cependant, l'utilisation de ces représentations semble avoir un impact assez marginal sur les résultats [Weck 2021]. D'autres approches ont alors proposé d'utiliser un décodeur de génération de texte pré-entraîné comme le *Bidirectional and Auto-Regressive Transformers* (BART) [Lewis 2020]. BART est un modèle de type encodeur-décodeur dont l'architecture est copiée sur celle de *Bidirectional Encoder Representations from Transformers* (BERT) [Devlin 2018]. Mais contrairement à ce dernier, BART a pour objectif de générer le prochain mot d'une phrase à l'aide des mots précédents plutôt qu'à prédire le mot masqué au milieu d'une phrase. De plus, BART a été entraîné avec de nombreuses augmentations appliquées en entrée comme des masquages, des permutations, des rotations ou des suppressions de mots.

La difficulté d'utiliser un décodeur pré-entraîné pour la DTAA est d'arriver à réutiliser l'information d'un décodeur qui est censé prendre en entrée une représentation du texte et non de l'audio. Ces deux représentations sont en général très différentes, ce qui fait oublier l'information que le décodeur pré-entraîné a acquise avant la spécialisation pour la DTAA. Pour éviter cela, les auteurs de [Gontier 2021] ont proposé de combiner les représentations audio et texte d'une manière différente. Ils utilisent deux encodeurs audio pré-entraînés (Wavegram-Logmel-CNN et YAMNet). Le premier sert à produire une représentation de l'audio sous forme d'un vecteur à taille fixe et le second sert à prédire les noms des classes d'événements sonores présentes. Les noms des événements reconnus sont ensuite donnés à la couche de représentation de BART pour produire une séquence de représentations texte à laquelle sera additionnée la représentation

16. https://github.com/LAION-AI/CLAP

audio. Cette séquence permet de combiner à la fois des informations audio et texte, qui seront données ensuite au décodeur de BART. Ce système a été l'état de l'art sur AC en 2021, et l'un des premiers à exploiter des informations d'un modèle de texte pré-entraîné efficacement. Dans la suite de ce manuscrit, ce modèle PANN-YAMNet-BART sera abrégé avec l'acronyme PYB. D'autres études [Koizumi 2020b, Schaumlöffel 2023] ont également utilisé le décodeur pré-entraîné GPT-2 (*Generative Pretrained Transformer*-2) [Radford 2019] pour initialiser leur décodeur.

De récentes approches [Kouzelis 2023, Deshmukh 2023] proposent d'entraîner une architecture encodeur-décodeur sur les descriptions seulement, puis de réutiliser la partie décodeur avec un autre encodeur audio pré-entraîné pendant l'inférence pour créer le modèle de DTAA. Dans ces études, les auteurs testent plusieurs méthodes différentes pour limiter l'incompatibilité entre les représentations du texte et de l'audio, et ils arrivent à obtenir des performances comparables à certains systèmes de DTAA (environ 40% et 43% sur la métrique principale, sur le jeu de données AC).

Enfin, une dernière approche inédite nommée *zero-shot audio captioning* présentée par [Salewski 2023] propose de ne pas utiliser directement des annotations durant l'apprentissage. Pour cela, ils exploitent un modèle de RAT pour récupérer un ensemble de mots clés d'un fichier audio, puis un modèle de langage pour générer des descriptions automatiquement à partir du top-k des mots clés les plus similaires à l'audio. Bien que ce type d'approche a pour avantage de ne pas exploiter d'annotations directement, il dépend beaucoup de l'ensemble de mots clés considéré au départ, du modèle de RAT et du modèle de langage pré-entraînés. Cette approche est aussi beaucoup moins efficace qu'une approche par apprentissage supervisé standard.

1.4.2 Apprentissage par renforcement

En DTAA, plusieurs études [Xu 2020b, Mei 2021b] ont tenté d'utiliser d'autres fonctions de coût que la CE, dont notamment le *Self-Critical Sequence Training* (SCST) [Rennie 2017]. Cette méthode est issue du domaine de l'apprentissage par renforcement (*Reinforcement Learning*, RL), et a originalement été appliquée en DTAI. Le but de SCST est ici d'entraîner les modèles directement sur une métrique non différentiable comme CIDEr-D (qui sera décrite dans le chapitre suivant, section 2.2.1). La première phase consiste à pré-entraîner un système de DTAA (typiquement avec la CE). Puis dans un second temps, le modèle génère une première prédiction pour chaque audio à l'aide du décodage glouton (décrit plus tard dans la section 1.4.5.1), et il crée une seconde prédiction à l'aide d'un décodage stochastique différent. Si cette seconde phrase a un score CIDEr-D plus élevé que la première, le modèle est récompensé et les probabilités de sortie de cette nouvelle phrase sont augmentées.

La fonction de coût finale est définie par l'équation 1.12, avec r la fonction de récompense pour un candidat, w^s la phrase échantillonnée par le modèle, w la phrase générée par défaut par le modèle (avec une recherche gloutonne) et p_θ la probabilité donnée par le modèle ayant pour paramètres θ.

$$\nabla_\theta L(\theta) = -\left(r(w^s) - r(w)\right) \nabla_\theta p_\theta(w^s) \tag{1.12}$$

Cette stratégie augmente considérablement le score CIDEr-D, avec 5% à 8% d'augmentation par rapport à un modèle seulement entraîné avec la CE ([Ye 2021a], [Ye 2022], [Mei 2022a]). Malgré ce gain, cette méthode semble conduire bien souvent à des phrases dites « dégénérées » [Mei 2021b], c'est-à-dire des phrases longues répétant les événements sonores avec différents mots

(*« a man speaks and a man is speaking »*) et de nombreuses erreurs de syntaxe comme des phrases non terminées (*« birds chirp and »*). Cependant, seules les métriques intégrant une vérification de la syntaxe comme FENSE ou SPIDEr-FL semblent réellement pénaliser ces phrases dégénérées. Nous détaillerons ces métriques dans le chapitre suivant (2).

1.4.3 Apprentissage multitâche

Lorsque la tâche de DTAA a émergé, l'emploi de données externes n'était pas encore la norme. Par exemple, durant la première itération de la tâche de DTAA pour la compétition DCASE, l'utilisation de données externes et de modèles pré-entraînés était interdite. Le seul jeu de données utilisable était CL, et ne contenait qu'environ 2800 fichiers d'entraînement dans sa première version. Pour palier au manque de données annotées par des descriptions, de nombreuses stratégies ont essayé de rajouter plusieurs fonctions de coût au réseau pour éviter le surapprentissage et améliorer la détection des événements.

Beaucoup de stratégies utilisent certains mots lexicaux (*content words*) extraits automatiquement des descriptions [Wu 2020, Eren 2020a, Kicinski 2022, Shin 2023, Eren 2023], en se basant sur leur fréquence ou sur un modèle de thème comme BERTopic [Grootendorst 2022]. Ces mots peuvent aussi être extraits à partir des noms d'origines des fichiers audio [Koizumi 2020c], ou encore issus des métadonnées [Koizumi 2020a] des jeux de données. Les mots-clés peuvent être employés comme des étiquettes à prédire pour l'encodeur, lui permettant d'associer un fichier audio à un ensemble d'événements ou de caractéristiques, car un encodeur directement entraîné sur les descriptions peine à converger. Les mots-clés produits par un encodeur peuvent également être concaténés aux représentations audio [Mei 2022a] ou combinés par un mécanisme d'attention [Ye 2021b] pour donner un contexte plus riche au décodeur. Ils peuvent aussi être prédits par le décodeur, en utilisant le dernier état caché du décodeur des descriptions ou en utilisant un autre décodeur en parallèle [Çakır 2020]. La figure 1.8 donne un aperçu de l'utilisation des mots-clés dans un système de DTAA.

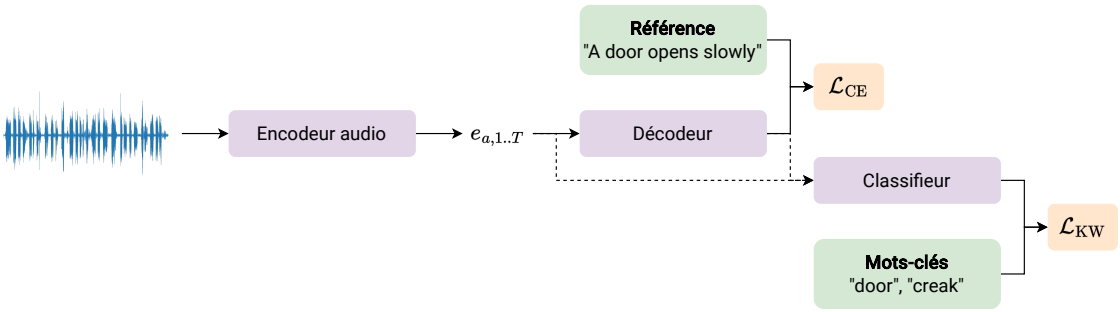


FIGURE 1.8 – Aperçu des méthodes DTAA utilisant une fonction de coût sur les mots-clés, notée $\mathcal{L}_{\text{KW}}$.

Une autre stratégie pour entraîner le modèle en utilisant plusieurs tâches consiste à conditionner les représentations audio produites par ce dernier. Cette contrainte se fait souvent en ajoutant une seconde fonction de coût visant à rapprocher la représentation issu de l'encodeur audio et la représentation de la vérité terrain, comme dans la figure 1.9. Cela peut se faire en générant un vecteur à taille fixe pour la référence à l'aide de modèles de texte pré-entraînés comme GloVe [Pennington 2014], SBERT [Reimers 2019], Instructor-XL [Su 2023] ou encore d'un modèle

entraîné à partir de zéro. Puis, il faut comparer ce vecteur avec celui de la représentation audio issue de l'encodeur comme pour [Wu 2023b], ou la représentation moyennée des sorties du décodeur comme pour [Xu 2021b, Mahfuz 2023b]. La fonction utilisée pour rapprocher les représentations peut être une similarité cosinus [Chen 2022a] ou une fonction contrastive plus complexe comme InfoNCE [van den Oord 2018]. Tous ces apprentissages peuvent donc permettre de créer un seul système réalisant la DTAA mais aussi la RAT.

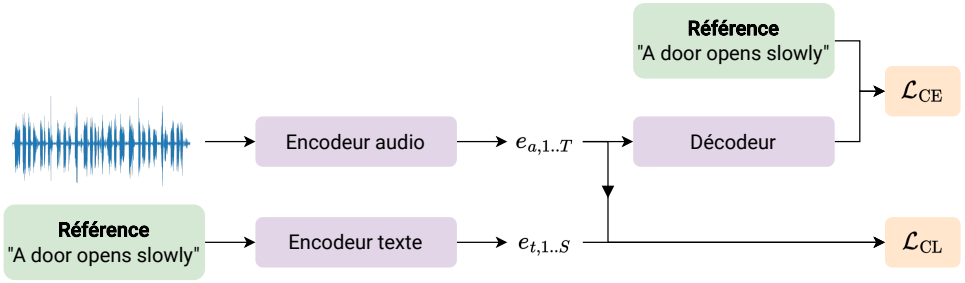


FIGURE 1.9 – Aperçu des méthodes DTAA utilisant une fonction de coût contrastive, notée $\mathcal{L}_{\text{CL}}$.

1.4.4 Augmentations de données

Pour améliorer la robustesse des modèles d'apprentissage automatique de DTAA, de nombreux systèmes emploient des augmentations de données. Le principe est d'appliquer une transformation aux données d'entrées pendant l'entraînement pour le rendre plus robuste à de légers changements et créer artificiellement de nouvelles données. Une bonne augmentation de données doit donc déformer les données d'entrée, mais ne doit pas changer l'étiquette de la donnée.

1.4.4.1 Augmentations audio

Pour la DTAA, l'utilisation d'augmentation sur l'audio est plus difficile que pour d'autres tâches, car une transformation du signal sonore peut directement impacter les descriptions. Par exemple, l'ajout de bruit en arrière-plan (comme un bruit gaussien ou bruit de brouhaha en parole) modifie l'étiquette elle-même, car ce bruit pourrait être décrit dans la phrase. C'est également le cas pour d'autres augmentations comme la réverbération, l'addition ou la concaténation de fichiers audio, le décalage de hauteur (*pitch shift*) ou la perturbation de la vitesse du signal. C'est pourquoi l'augmentation la plus utilisée [Wang 2020, Schaumlöffel 2023] est SpecAugment (SA) [Park 2019], qui masque aléatoirement une partie du spectrogramme d'entrée, ce qui impacte *a priori* peu les descriptions. Des exemples d'application de SpecAugment sont donnés en illustration dans la figure 1.10. Une variante nommée SpecAugment++ [Wang 2021] est également utilisée dans quelques études [Ye 2022], et somme plusieurs signaux entre eux sur certaines bandes.

Malgré l'impact des augmentations sur l'étiquette des phrases, plusieurs études en DTAA emploient des augmentations comme le bruit gaussien [Huang 2022], le bruit blanc [Kicinski 2022] ou la réverbération et la perturbation de vitesse [Han 2021]. Dans ces études, ces augmentations semblent nécessaires pour limiter le surapprentissage et parfois améliorent légèrement les performances (+0,5% de la métrique SPIDEr pour du bruit gaussien par exemple dans [Huang 2022]).

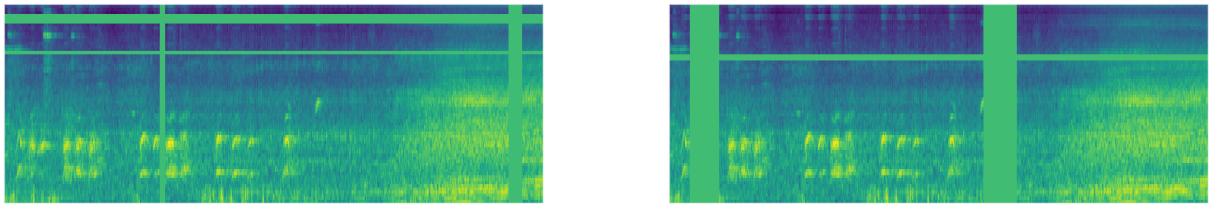


FIGURE 1.10 – Exemples de deux log-Mel spectrogrammes modifiés par le SpecAugment.

1.4.4.2 Mixup et ses variantes

Mixup [Zhang 2018] est une augmentation de données qui somme deux signaux sonores différents entre eux ainsi que leurs étiquettes. Dans un cadre de classification, le but est d'encourager le modèle à lisser ses courbes de décisions pour se comporter de manière plus linéaire entre les exemples d'entraînement. La figure 1.11 est un exemple du Mixup dans le domaine des images.

(a) Mon chat : 100%
Mon serpent : 0%

(b) Mon chat : 0%
Mon serpent : 100%

(c) Mon chat : 40%
Mon serpent : 60%

FIGURE 1.11 – Exemple de Mixup dans le domaine des images.

Si les deux exemples sont notés x_1 et x_2 et que leurs étiquettes respectives y_1 et y_2 sont encodées sous forme de distribution, on obtient x^{mix} et y^{mix} ainsi :

$$\begin{aligned} x^{mix} &= \lambda x_1 + (1-\lambda)x_2 \\ y^{mix} &= \lambda y_1 + (1-\lambda)y_2 \end{aligned} \tag{1.13}$$

Le coefficient λ est une valeur dans l'intervalle [0, 1] qui sert à contrôler à quel point le résultat x^{mix} est proche de x_1 ou de x_2. En pratique, λ est échantillonné à chaque batch à partir de la loi Bêta paramétrée par α :

$$\lambda \sim \text{Beta}(\alpha, \alpha) \tag{1.14}$$

L'hyperparamètre α est un nombre positif réel qui permet de contrôler la force du mélange pendant l'entraînement. Comme on peut voir dans la figure 1.12, lorsque α tend vers 0, la densité de la distribution se concentre autour de 0 et 1, ce qui signifie que x^{mix} est souvent proche de x_1 ou de x_2. Si α tend vers 1, la distribution devient uniforme et le mélange est globalement plus accentué avec des coefficients λ pouvant être à 0,5. Ce dernier cas peut permettre de renforcer la généralisation, mais il peut aussi mener à une sous-adaptation (*underfitting*) du modèle en perturbant trop les exemples de l'entraînement.

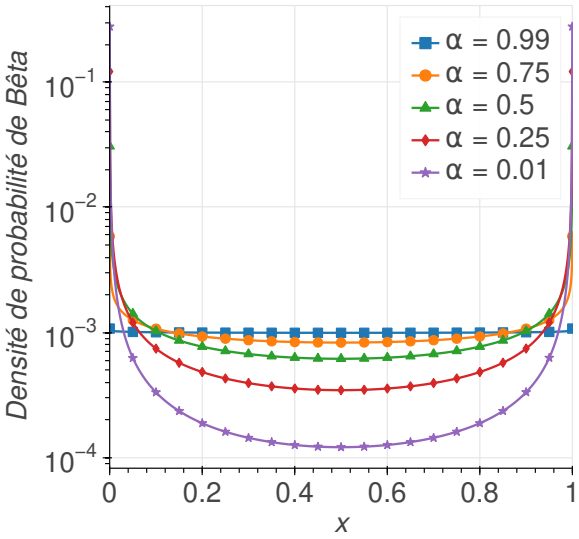


FIGURE 1.12 – Densité de probabilité de la loi Bêta pour plusieurs valeurs de α.

Mixup est en général utilisé comme augmentation de données avec x_1 le lot d'origine et x_2 correspondant au même lot dont l'ordre des exemples a été mélangé. Une opération peut être ajoutée après la génération du coefficient λ pour rendre Mixup « asymétrique », comme dans l'équation 1.15. Après cette opération, la valeur de λ est donc dans l'intervalle [0,5, 1] et force le résultat x^{mix} à être plus proche de x_1 que de x_2 (et y^{mix} sera plus proche de y_1 que de y_2).

$$\lambda = \max(\lambda, 1 - \lambda) \tag{1.15}$$

Pour la description automatique, l'addition des phrases est plus difficile à réaliser que pour de simples classes, ce qui a poussé la recherche à créer plusieurs variantes de Mixup. De plus, les modèles de DTAA prennent en entrée les mots précédents de la phrase, qui peuvent également être mélangés par le Mixup.

MixGen [Hao 2023] utilisé en DTAA par [Kim 2022, Emmanouilidou 2023] est une variante qui concatène les phrases au lieu de les additionner. Elle permet donc d'entraîner le réseau à mieux repérer une superposition d'événements, mais elle ne lie pas les phrases concaténées par un mot comme *« while »*. De plus, elle peut encourager le modèle à prédire des phrases beaucoup plus longues que celles présentes dans le corpus d'entraînement, ce qui nécessite d'appliquer l'augmentation avec parcimonie. Les équations 1.16 montrent les étapes du Mixup modifiées par cette augmentation. Les suffixes *prev* et *next* indiquent respectivement les mots précédents et suivants nécessaires durant l'entraînement. Par exemple, si x_1 est *« <bos> a man speaks <eos> »*, $x_{1,prev}$ désigne *« <bos> a man speaks »* (l'entrée du modèle) et $x_{1,next}$ désigne *« a man speaks <eos> »* (la sortie attendue du modèle).

$$\begin{aligned} y_{1,2,prev} &= \text{Cat}(y_{1,prev}, y_{2,prev}) \\ y_{1,2,next} &= \text{Cat}(y_{1,next}, y_{2,next}) \\ z_{1,2} &= f(x^{mix}, y_{1,2,prev}) \\ \mathcal{L} &= \text{CE}(z_{1,2}, y_{1,2,next}) \end{aligned} \tag{1.16}$$

À noter que la valeur du λ échantillonnée ne doit pas être trop faible pour que le modèle puisse repérer les événements en arrière-plan dans l'audio, ce qui pousse à utiliser des valeurs proches ou au-dessus de 1 pour l'hyperparamètre α.

Une autre version de Mixup que nous nommons « **Mixup-in** » est employée dans les études [Kouzelis 2022, Takeuchi 2020]. Curieusement, elle ne s'applique que sur l'audio et sur les représentations des mots précédents en entrée du décodeur, mais pas sur les mots suivants attendus. L'algorithme est présenté dans l'équation 1.17. W désigne la couche des représentations du décodeur pour les mots d'entrée et f est le reste du décodeur.

$$\begin{aligned} w_1 &= W(y_{1,prev}) \\ w_2 &= W(y_{2,prev}) \\ w^{mix} &= \lambda w_1 + (1-\lambda) w_2 \\ z^{mix} &= f(x^{mix}, w^{mix}) \\ \mathcal{L} &= \mathrm{CE}(z^{mix}, y_{1,next}) \end{aligned} \tag{1.17}$$

Selon la valeur de l'hyperparamètre α, cette version du Mixup pourrait rendre le modèle moins sensible aux bruits de fond, car il a été entraîné sur des entrées mélangées avec des étiquettes non mélangées. Contrairement au **MixGen**, cette version du Mixup ne doit pas utiliser de valeur de α trop élevée pour éviter de perturber le modèle en lui demandant de ne décrire qu'une partie de l'audio mélangé.

Enfin, une dernière version de Mixup utilisant un modèle ChatGPT a été proposée par [Wu 2023b], nommée « ChatGPT-Mixup ». Les fichiers audio sont toujours ajoutés entre eux, mais en s'assurant au préalable que les parties soient toutes les deux audibles. Pour cela, ils vérifient que la racine carrée moyenne des carrés de leurs énergies relatives est de 5 dB au maximum. Puis, les deux références correspondantes sont données au modèle ChatGPT, avec l'instruction de créer un mélange de ces deux phrases de moins de 25 mots. Un exemple de mélange de descriptions est donné dans le tableau 1.7. La capacité d'un grand modèle de langage à reformuler et à synthétiser de nouvelles phrases semble permettre de créer des références vraisemblables qui enrichissent leur modèle de DTAA, avec 1,9% à 2,4% d'amélioration de la métrique SPIDEr d'après l'étude.

Table 1.7 – Exemple de phrases mélangées par ChatGPT.

N°	Phrases
y_1	*Water flowing over some rocks throughout a creek.*
y_2	*In the distance fireworks pop and crackle constantly as they are set off.*
$y_{1,2}^{mix}$	*A serene creek babbles over rocks as distant fireworks pop and crackle in celebration.*

1.4.4.3 Autres augmentations

Plus marginalement, il existe d'autres augmentations appliquées au niveau du modèle ou au niveau du texte uniquement. Par exemple, les auteurs de [Kim 2022] ont proposé d'utiliser une augmentation de données nommée *Multi-Modal Test-Time Adaptation* (MM-TTA) [Shin 2022], pour améliorer la représentation du modèle ACT pendant l'inférence. MM-TTA consiste à

appliquer une augmentation stochastique sur l'entrée pour produire plusieurs représentations qui seront ensuite moyennées sur différentes couches du réseau. La représentation finale sera donc plus complète et robuste, ce qui améliore les performances pour la génération de phrases candidates. Les augmentations utilisées pour produire des variantes de l'audio sont un bruit additif Gaussien et SpecAugment. À noter que dans la littérature de DTAA, il existe aussi le remplacement de synonymes appliqué sur les descriptions, qui est appliqué dans l'étude [Cho 2023]. Plus précisément, il permet d'améliorer les capacité de génération du modèle et semble être bénéfique lors de la phase de spécialisation de leur modèle sur le jeu de données CL.

1.4.5 Méthodes d'inférence

Lors de la génération de texte par un modèle de langage, il existe de nombreuses manières d'exploiter la distribution de probabilité du mot suivant crée par le modèle. En DTAA, certaines de ces méthodes ont été utilisées pour générer une ou plusieurs phrases candidates. La méthode la plus simple est la recherche gloutonne, mais la plus utilisée est sans conteste la recherche en faisceaux. Quelques approches se servent également des méthode stochastiques, qui permettent de proposer plusieurs phrases par fichier audio. Enfin, de nombreux participants aux compétitions DCASE emploient une méthode de décodage ensembliste, qui fusionne les prédictions de plusieurs systèmes durant la génération.

1.4.5.1 Recherche gloutonne

La recherche gloutonne (*greedy search*) est une méthode de génération de texte qui consiste à ne prendre que le mot le plus probable selon le modèle comme prédiction. Elle se rapproche de l'algorithme du parcours en profondeur, qui ne cherche qu'un seul chemin vers un état final. Cet algorithme a pour avantage d'être très rapide pour générer une phrase, mais il peut manquer des phrases plus vraisemblables, car il ne prend pas en compte la probabilité des choix futurs qui découle d'un état et ne permet aucun retour en arrière (*backtracking*).

1.4.5.2 Recherche en faisceau

La méthode de recherche en faisceau (*beam search*) est un algorithme glouton d'exploration de graphe. Il repose sur le principe du parcours en largeur, mais ne considère que les k prochains meilleurs nœuds (mots) à chaque étape pour éviter l'explosion combinatoire du coût de calcul. Les nœuds suivants sont ordonnés selon la probabilité que leur donne le réseau, ce qui permet d'explorer d'autres chemins et donc de générer d'autres phrases. L'hyperparamètre k est nommé largeur du faisceau, avec une valeur comprise entre 2 à 10 pour la DTAA. Contrairement à la recherche gloutonne, l'heuristique pour sélectionner les mots suivants utilise la probabilité de tous les mots précédents de la phrase, définie par l'équation 1.18. Une phrase s'arrête lorsque le mot de fin de phrase (`<eos>`) est atteint, ou lorsqu'une taille maximale de la phrase $L_{\max}$ est atteinte. Une fois toutes les k phrases construites, la meilleure sera sélectionnée selon sa probabilité totale, qui dépend des probabilités de chacun de ses mots et de sa longueur.

$$\begin{aligned} \underset{w}{\operatorname{argmax}}\, P(w_{1\ldots L}) &= \underset{w}{\operatorname{argmax}} \left(\prod_{i=1}^{L} P\left(w_i | w_{1..i-1}\right) \right)^{1/L} \\ &= \underset{w}{\operatorname{argmax}} \frac{1}{L} \left(\sum_{i=1}^{L} \log P(w_i | w_{1..i-1}) \right) \end{aligned} \tag{1.18}$$

avec w_i le i-ème mot de la phrase de longueur L et P la probabilité donnée par le modèle pour un mot.

1.4.5.3 Recherches stochastiques

En génération de texte, les méthodes de recherches gloutonnes ou en faisceau produisent en général des textes assez répétitifs sur la durée [Holtzman 2020a]. De plus, les modèles de génération tendent à générer les n-grammes les plus probables, en ignorant certains moins fréquents, mais tout aussi valides. Pour améliorer la diversité du contenu généré, les modèles intègrent des méthodes d'inférence stochastiques, qui permettent d'échantillonner le mot suivant selon la distribution de probabilité donnée par le modèle. L'une de ces méthodes est l'échantillonnage par noyau (*nucleus sampling* ou top-p) [Holtzman 2020b] considère le nombre minimum de mots les plus probables dont la somme des probabilités est supérieure à p.

Dans l'étude [Wu 2023b], les auteurs ont proposé une méthode de génération avancée pour sélectionner la meilleure phrase parmi une liste de 50 candidats par audio générée par l'échantillonnage par noyau. Le problème s'apparente à la tâche d'audio-vers-texte, où l'ensemble de texte correspond aux candidats générés pour le fichier audio. En premier lieu, le détecteur d'erreurs provenant de la métrique FENSE (dont nous parlerons dans le chapitre suivant) est utilisé pour filtrer les candidats contenant des erreurs. Puis, pour chaque candidat, un score est calculé à l'aide de la probabilité donnée par le modèle et de la similarité entre la représentation de l'audio et la représentation texte du candidat issu d'un modèle Instructor-XL. Ce score rend l'inférence plus coûteuse, mais fournit un gain de 1 à 2% de la métrique SPIDEr sur CL-eval.

1.4.5.4 Recherches par ensemble

Pour combiner plus modèles de DTAA, de nombreux participants aux compétitions DCASE ont recours à une méthode d'ensemble [Yuan 2021, Primus 2022, Cho 2023]. La méthode consiste à moyenner les sorties brutes (*logits*) ou les probabilités de sorties de chaque modèle pour définir le prochain mot de la phrase. Ce mot prédit sera donné à chaque modèle à l'itération suivante du décodage, qui auront donc chacun la même suite de mots précédents. Bien souvent, les modèles utilisés sont des modèles entraînés sur plusieurs graines, ou bien des modèles légèrement différents dans leur architecture. Les auteurs [Kim 2023a] indiquent également que la combinaison de modèles différents (entraînés par renforcement ou par CE) peut être plus bénéfique que d'utiliser des modèles entraînés de la même manière. Ces méthodes peuvent être combinées avec la recherche en faisceau ou des recherches stochastiques pour générer de meilleures phrases. Le nombre de modèles varie beaucoup en fonction des études, allant de 3 à 50 modèles, ce qui peut rendre l'inférence coûteuse et créer des systèmes possédant un très grand nombre de paramètres (plusieurs milliards de paramètres au total pour les premières soumissions DCASE en 2022 et 2023).

1.4.6 Aperçu de l'évolution des systèmes de DTAA

Bien que tous les systèmes de DTAA emploient des DNN, les architectures, augmentations et méthodes d'apprentissage diffèrent sur de nombreux aspects. Elles sont souvent inspirées d'autres tâches comme l'AT, la RAT, la RAP ou encore la DTAI. Le tableau 1.8 donne un aperçu chronologique des différentes architectures et quelques points clés utilisés dans les différents états de l'art de la DTAA. Quand la tâche de DTAA est apparue, la majorité des systèmes employaient des réseaux récurrents. Puis, les systèmes ont évolués en se servant de modèles

pré-entraînés qui reposent principalement des CNN. Les architectures *Transformer* ont d'abord supplenté certains décodeurs récurrents, puis certains encodeur pré-entraînés avec PaSST ou BEATs. Rapidement, les systèmes de DTAA ont introduit plusieurs autres fonction de coût à la CE comme des apprentissages multitâches pour aider le modèle à converger ou des apprentissages par renforcement pour améliorer directement les performances d'une métrique. Puis, les systèmes ont commencé à introduire des information textuelles au travers de modèles pré-entraînés, qui sont venu aider la génération du modèle ou reclasser les candidats produits par ce dernier.

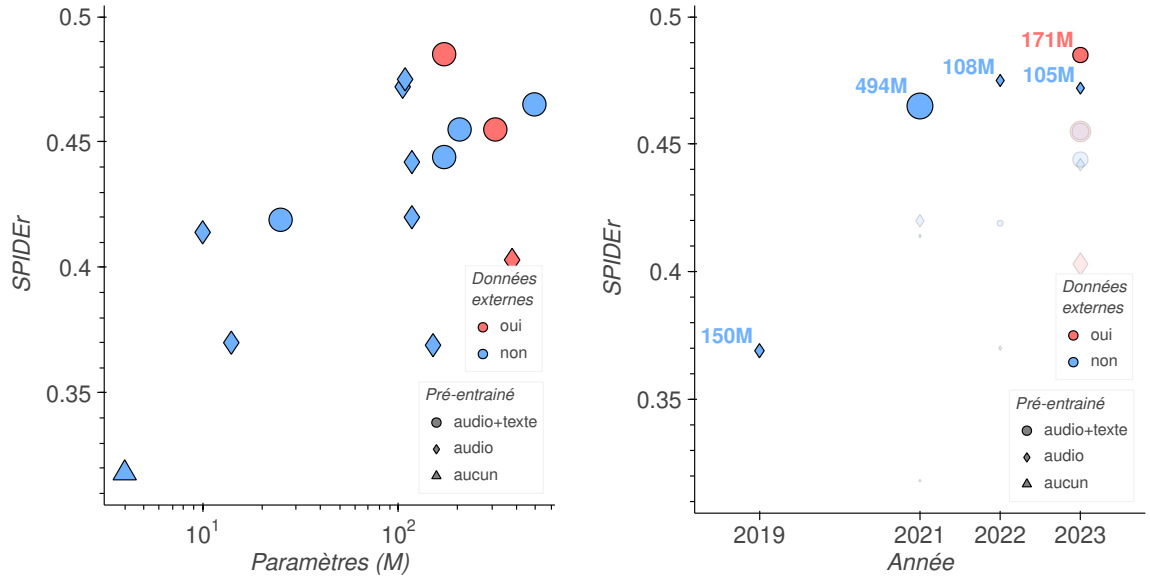
Table 1.8 – Aperçu de plusieurs méthodes de DTAA entre 2017 et 2023.

Référence	**Modèle**	**Points-clés**
[Drossos 2017]	BiGRU-GRU	Introduction de la DTAA
[Kim 2019a]	VGGish-LSTM	AudioCaps
[Drossos 2020]	BiGRU-GRU	Clotho
[Koizumi 2020c]	CNN-BiLSTM-LSTM	Apprentissage multitâche
[Xu 2020b]	CRNN-GRU	Apprentissage par Renforcement
[Gontier 2021]	PYB	Classes de YAMNet+Décodeur BART
[Han 2021]	PANN-Transformer	Apprentissage faiblement supervisé
[Kouzelis 2022]	PaSST-Transformer	Pré-entraînement de PaSST+Mixup
[Kim 2022]	ACT	Multi-TTA et MixGen aug.
[Mei 2023]	CNN14-BART	WavCaps
[Wu 2023b]	BEATs-BART	BEATs+ChatGPT aug.+Instructor

Les figures 1.13 et 1.14 présentent l'état de l'art des différents systèmes pour la métrique principale nommée SPIDEr, qui sera présentée dans le chapitre suivant 2. Une version dynamique de ces figures est également disponible en ligne [17], et les valeurs détaillées par système sont données en annexe dans les tableaux B.2 et B.3. Ces figures montrent l'évolution de la performance des systèmes en fonction de leur nombre de paramètres (en échelle logarithmique pour 1.13a et 1.14a) et par année. La taille des disques pour les figures 1.13b et 1.14b est proportionnelle au carré du nombre de paramètres. À noter que nous ne considérons que les systèmes qui n'emploient pas de RL, qui obtiennent un score SPIDEr supérieur à 15% ainsi que ceux qui utilisent le jeu de validation de CL pour l'entraînement. Au final, nous avons récupéré les résultats de 15 systèmes pour AC et 32 pour CL entre 2019 et 2023.

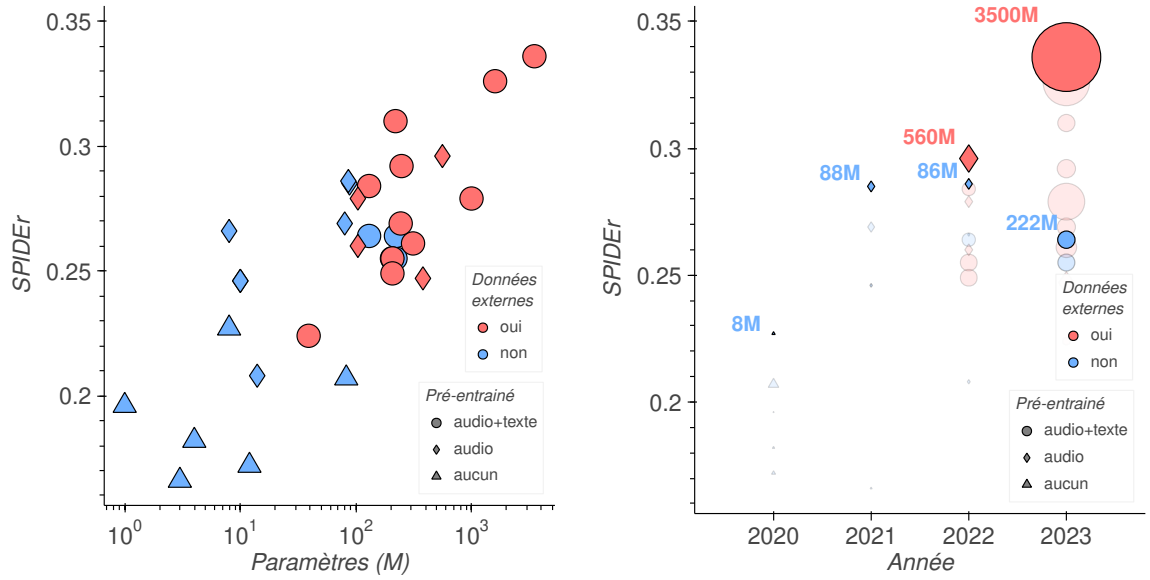
Sur AC, la plupart des modèles sont entraînés sans données externes, mais en 2023 le nouvel état de l'art dépasse légèrement le précédent à l'aide de données supplémentaires, avec 48,5% de SPIDEr. La plupart des systèmes dépassent 100 millions de paramètres, qui semble nécessaire pour obtenir une bonne performance sur ce jeu de données. Cependant, l'utilisation d'un modèle de texte pré-entraîné n'améliore pas toujours les performances, mais fait croître le nombre de paramètres. L'état de l'art évolue lentement avec approximativement +2% de SPIDEr entre 2021 et 2023, avec un modèle environ trois plus petit en nombre de paramètres. Assez peu d'études exploitent des données externes pour ce jeu de données, car AC est plus large que CL et le sous-ensemble de test n'est pas au centre d'une compétition comme DCASE.

17. https://labbeti.github.io/blog/manuscrit/sota.html



(a) SPIDEr par nombre de paramètres sur AC. (b) SPIDEr par année sur AC.

FIGURE 1.13 – État de l'art de DTAA sur AC, par nombre de paramètres et année.



(a) SPIDEr par nombre de paramètres sur CL. (b) SPIDEr par année sur CL.

FIGURE 1.14 – État de l'art de DTAA sur CL, par nombre de paramètres et année.

Sur CL, les meilleurs systèmes exploitent des données externes et des modèles de texte pré-entraînés, mais sont également ceux qui comportent le plus de paramètres. La différence entre les tailles des modèles sur CL par rapport à AC s'explique notamment par la compétition DCASE, où les participants utilisent des modèles plus grands ou un ensemble de plusieurs modèles pour améliorer les performances. Le meilleur système en 2023 obtient 33,6% de SPIDEr avec un ensemble de 20 modèles de DTAA combinés avec un modèle de représentation de texte. Cet ensemble contient plus de trois milliards de paramètres, ce qui est immense pour un jeu de

données aussi petit que CL, et laisse douter de sa capacité de généralisation sur d'autres types de données de DTAA. On peut également noter que le meilleur système sans encodeur audio pré-entraîné obtient 23% de SPIDEr, qui est en dessous de la majorité des systèmes qui en utilisent un. De nombreux systèmes contiennent plusieurs centaines de millions de paramètres et obtiennent des performances autour de 26% de SPIDEr, et semblent plutôt correspondre aux systèmes exploitant un modèle de PANN. Les systèmes de moins de cent millions de paramètres semblent stagner autour de 29% de SPIDEr au maximum. Ceux allant plus loin en SPIDEr utilisent beaucoup plus de paramètres à cause de leur modèle de texte pré-entraîné ou d'une méthode ensembliste. On peut également observer que les systèmes proposés en 2023 semblent exploiter de plus en plus de données externes, ce qui explique que l'état de l'art sans données externes ait reculé.

1.5 Bilan

Ce chapitre avait pour but de présenter l'état de l'art des méthodes de DTAA. En premier lieu, nous avons détaillé les différents types de neurones, les méthodes d'optimisation et la principale fonction de coût employée pour créer ces systèmes. En particulier, nous avons décrit les convolutions, les cellules récurrentes et les modules d'attention qui permettent de créer des réseaux de neurones robustes.

Puis, nous avons présenté l'ensemble des jeux de données publics qui sont utilisés pour entraîner les modèles de DTAA. Nous avons montré que ces jeux diffèrent de plusieurs manières, que ce soit en termes de quantité de données, de méthode d'annotation, ou de métadonnées accessibles.

Enfin, nous avons donné un aperçu des différentes architectures de DTAA qui existent, et leurs diverses manières d'exploiter les données audio et texte. Nous y retrouvons notamment l'utilisation de mots-clés pour améliorer les modèles, ou l'apprentissage par transfert pour exploiter de plus larges jeux de données de classification audio. Finalement, nous avons décrit plusieurs manières de représenter et d'augmenter les données audio et texte pour simplifier et parachever l'apprentissage des modèles de DTAA.

Durant mes trois années de thèse, les modèles de la littérature se sont beaucoup améliorés, ce qui nous a fait progressivement modifier mes systèmes pour suivre cette évolution au niveau de l'architecture du modèle, des données d'entraînement et du pré-entraînement de l'encodeur. Nous avons également choisi de ne pas utiliser l'apprentissage par renforcement, car il optimise la métrique CIDEr-D au détriment de la syntaxe des phrases générées. Dans le chapitre suivant, nous présenterons le fonctionnement de l'ensemble des métriques qui ont été utilisées pour évaluer les systèmes de DTAA, ainsi qu'un aperçu de leurs valeurs sur les différents jeux de données de validation et de test disponibles.

Chapitre 2

Métriques

L'évaluation des systèmes de DTAA est un élément crucial de la recherche sur cette tâche. En effet, la DTAA est une tâche de génération de texte où l'on n'attend pas nécessairement à ce que le système soit capable de générer une phrase identique à la vérité terrain. On cherche plutôt à produire une phrase qui contienne la même information que dans une ou plusieurs phrases de référence pour les événements sonores présents, leurs attributs et leurs relations. Il est donc impossible de se contenter de certaines métriques comme le taux d'erreur mot.

La difficulté d'évaluer automatiquement les systèmes a donné lieu à de nombreuses métriques, qui prennent en compte des aspects différents des descriptions. En premier lieu, nous décrirons les premières métriques qui ont été utilisées pour comparer les systèmes de DTAA, héritées d'autres domaines comme la traduction automatique, de la DTAI ou la génération de texte. Ensuite, nous présenterons les nouvelles métriques spécialement conçues pour la DTAA, qui prennent en compte d'autres aspects comme la fluence des descriptions. Nous citerons également certaines métriques liées à la diversité des phrases. Enfin, nous présenterons un aperçu des valeurs maximales de ces métriques à l'aide des scores humains, sur les jeux de données AudioCaps et Clotho.

Sommaire

Dans ce chapitre, la phrase générée par un système automatique (prédiction) est nommée « candidat » et notée c, tandis que la phrase correspondante annotée par un humain (vérité terrain) est appelée « référence » et notée r. Comme précisé dans la partie sur les jeux de données en section 1.3, chaque fichier audio de validation ou de test peut être associé à un ensemble de plusieurs références qui sera noté R. Pour un audio, la plupart des métriques comparent un candidat avec une seule référence, puis moyennent ces scores pour chaque référence. L'ensemble des candidats prédits pour une liste de fichiers audio est noté C et leurs ensembles de références multiples sont notés R_M.

2.1 Traduction automatique

Historiquement, les métriques utilisées pour la DTAA sont héritées de celle de la DTAI, qui elles-mêmes s'appuient sur les métriques de traduction automatique. Ces métriques sont principalement fondées sur le recouvrement des n-grammes entre le candidat et la référence.

2.1.1 BLEU

BiLingual Evaluation Understudy (BLEU) [Papineni 2001] est un score de précision modifié (voir équation 2.1) pour les n-grammes allant de 1 à N. Autrement dit, le score évalue si les mots du candidat prédit sont bien présents dans la référence.

$$p_n(c, R) = \frac{\sum_{n\text{-gram}\in c} \min(\text{Occurence}(n\text{-gram}, c), \max_{r\in R} \text{Occurence}(n\text{-gram}, r))}{\sum_{n\text{-gram}\in c} \text{Occurence}(n\text{-gram}, c)} \tag{2.1}$$

Pour éviter que le système surévalue des phrases trop courtes (qui pourraient plus facilement contenir quelques bons mots), les auteurs ajoutent une fonction de pénalité (*Brevity Penalty*, BP) décrite dans l'équation 2.2. Cette dernière dépend de la longueur totale de tous les candidats L_C, comparés à la somme des longueurs totales de toutes les références L_{R_M} les plus proches de celles de chaque candidat. À noter que certaines implémentations de BLEU n'appliquent pas toujours cette pénalité. De plus, cette version du score BLEU de chaque candidat varie donc en fonction des autres candidats et des autres références, puisque la pénalité dépend de la longueur totale de tous les candidats et références.

$$\text{BP}(C, R_M) = \exp(1 - L_{R_M}/L_C), \text{ si } L_{R_M} \geq L_C \text{ sinon } 1 \tag{2.2}$$

Le score BLEU combine les scores de précision et de pénalité par la formule suivante :

$$\text{BLEU}_N(C, R_M) = \text{BP}(C, R_M) \cdot \exp\left(\sum_{(c,R)\in(C\times R_M)} \sum_{n=1}^{N} w_n \log p_n(c, R)\right) \tag{2.3}$$

BLEU utilise des poids w_n uniformes ($w_n = \frac{1}{N}$). Pour la DTAA, la valeur de N choisie varie de 1 à 4, et nous nommons respectivement ces métriques BLEU1, BLEU2, BLEU3 et BLEU4 (avec les abréviations B1, B2, B3 et B4).

2.1.2 ROUGE-L

Recall-Oriented Understudy for Gisting Evaluation (ROUGE, abrégé RG) [Lin 2004] est un ensemble de métriques conçues pour évaluer les systèmes résumant du texte automatiquement. ROUGE-L est l'une de ces métriques, et calcule le F-score modifié reposant sur la taille de la plus

longue sous-séquence (*Longest Common Subsequence*, LCS) commune entre le candidat et les références. Contrairement à la plupart des autres métriques, elle prend en compte de multiples références pour chaque candidat en calculant le maximum des LCS. Les équations 2.4 décrivent le calcul du score ROUGE-L.

$$\mathrm{Pr}(c, R) = \max_{r \in R} \frac{\mathrm{LCS}(c, r)}{|c|} \tag{2.4a}$$

$$\mathrm{Re}(c, R) = \max_{r \in R} \frac{\mathrm{LCS}(c, r)}{|r|} \tag{2.4b}$$

$$\text{ROUGE-L}_\beta(c, R) = \frac{(1 + \beta^2) \cdot \mathrm{Pr}(c, R) + \mathrm{Re}(c, R)}{\beta^2 \cdot \mathrm{Pr}(c, R) + \mathrm{Re}(c, R)} \tag{2.4c}$$

Par défaut, la métrique ROUGE-L utilise la valeur $\beta = 1{,}2$, ce qui signifie que le rappel est légèrement plus important que la précision dans le calcul du F-score.

2.1.3 METEOR

Metric for Evaluation of Translation with Explicit ORdering (METEOR, abrégé ME) [Denkowski 2014] est une évolution de BLEU visant à être mieux corrélée avec le jugement humain en intégrant des mécanismes plus complexes. Premièrement, la métrique repose sur un F-score au lieu d'une simple précision. Grâce à la prise en compte du rappel dans le F-score, les candidats qui ne manquent aucun mot des références sont mieux évalués que les autres, ce qui pénalise déjà les courts candidats. La fonction de pénalité de BLEU est donc supprimée. Contrairement à BLEU, METEOR ne prend en compte que les ensembles des mots (1-gramme) de chaque phrase, notés $W(c)$ et $W(r)$. De plus, METEOR ajoute un détecteur de synonymes basé sur WordNet [Miller 1994], une racinisation (*stemming*) sur les mots [Porter 2001] pour faire correspondre des formes différentes comme « *speak* », « *speaking* » ou « *talk* » et un détecteur de paraphrase se basant sur des traducteurs automatiques [Bannard 2005]. L'objectif ici est de relaxer la comparaison mot-à-mot pour éviter de pénaliser les systèmes qui produiraient une phrase valide, mais contenant des mots légèrement différents. Pour prendre en compte l'ordre des mots, cette métrique ajoute un score de pénalité différent. Ce dernier repose sur le nombre de morceaux (*chunk*), qui est égal au nombre minimal de sous-séquences communes et noté $\mathrm{Chunk}(c, r)$. Cette valeur sera divisée par le nombre de mots correspondants (vrais positifs), noté $\mathrm{Match}(c, r)$.

$$\mathrm{METEOR}(c, r) = \mathrm{F}_\alpha\left(W(c), W(r)\right) \cdot \gamma \left(\frac{\mathrm{Chunk}(c, r)}{\mathrm{Match}(c, r)}\right)^\beta \tag{2.5}$$

α, γ et β sont trois hyperparamètres optimisés pour mieux correspondre au jugement humain selon la langue et sont respectivement égaux à 0,9, 3 et 0,5.

2.2 Description Textuelle Automatique de l'Image

Après l'apparition de la tâche de DTAI en 2015, de nombreuses études ont montré les limites de l'utilisation des métriques de traduction pour évaluer des systèmes de description automatique [Elliott 2014, Hodosh 2013, Novikova 2017, Chaganty 2018]. En effet, ces métriques ne prennent pas assez en compte la variabilité et la diversité des descriptions. Par exemple, une phrase comme « *A man is speaking and a car is passing by* » et « *Male speech occurs in the street* » sont en général considérées comme peu similaires. De plus, les poids associés à un mot indiquant un objet dans l'image, un déterminant ou la scène entière sont identiques pour les métriques de traduction.

2.2.1 CIDEr

La première métrique apparue pour évaluer les systèmes de DTAI est *Consensus-based Image Description Evaluation* (CIDEr, abrégé CD) [Vedantam 2015]. Le principal aspect de cette métrique est la prise en compte de la fréquence des n-grammes des phrases, permettant de pondérer l'importance de ces derniers. Les n-grammes peu fréquents sont considérés comme plus informatifs, et augmentent donc considérablement le score de la métrique lorsqu'ils sont présents dans les candidats et les références. S'ils ne sont pas présents dans les références, ils pénaliseront radicalement le score. À l'inverse, les n-grammes très fréquents n'impacteront que marginalement le score de CIDEr.

Pour prendre en compte ces fréquences, CIDEr repose sur le *Term Frequency-Inverse Document Frequency* (TF-IDF) des n-grammes. Cette mesure permet de calculer l'importance d'un terme t dans un document d en se basant sur sa fréquence dans un corpus D. Plus précisément, elle repose sur le nombre d'occurrences du terme t dans le document dont il est issu (*Term Frequency*) et sur l'inverse de son nombre d'occurrences dans un plus large corpus (*Inverse Document Frequency*). L'équation 2.6 détaille le calcul à effectuer pour calculer la TF-IDF.

$$\begin{aligned} \text{TF-IDF}(t, d, D) &= \text{TF}(t, d) \cdot \log \text{IDF}(t, D) \\ &= \frac{\text{Occurrence}(t, d)}{|d|} \cdot \log \frac{|D|}{|d_2 \in D : t \in d_2|} \end{aligned} \tag{2.6}$$

Pour la DTAI, un terme correspond à un n-gramme, un document à une phrase et le corpus de documents à l'ensemble des phrases des références dans le sous-ensemble de test. Cela signifie que le score CIDEr d'un candidat comparé à une référence dépend également de la fréquence des mots dans l'ensemble des références du sous-ensemble de test, ce qui est assez variable en fonction des jeux de données. Une fois les TF-IDF des n-grammes du candidat et de la référence calculés, les scores seront séparément concaténés en deux vecteurs puis alignés pour calculer une similarité cosinus (voir équation 2.7). La figure 2.1 résume le procédé global de CIDEr pour des 1-grammes.

$$\text{cos-sim}(e_a, e_b) = \frac{e_a \cdot e_b}{||e_a||\ ||e_b||} \tag{2.7}$$

En pratique, CIDEr calcule la similarité cosinus pour les n-grammes allant de 1 à 4, et moyenne ces similarités pour chaque phrase. Les scores sont multipliés par un facteur dix pour correspondre aux jugements donnés par des humains dans l'étude d'origine. Une racinisation des mots est appliqué en amont pour trouver les radicaux des mots avant de calculer la TF-IDF, ce qui permet de faire correspondre des mots qui ne sont pas nécessairement sous la même forme. Cependant, la métrique CIDEr peut parfois surévaluer certaines phrases considérées comme peu admissibles par des êtres humains (effet nommé « *gaming* »). C'est pourquoi la métrique CIDEr-D fut introduite peu après pour corriger cet aspect, avec trois différentes modifications par rapport à CIDEr :

- Suppression de la racinisation pour prendre en compte la forme, la pluralité ou encore le temps des verbes ;
- Ajout d'un facteur de pénalité gaussien fondé sur la longueur des phrases ;
- Limitation du nombre d'occurrences de chaque n-gramme des candidats par le nombre d'occurrences de ce n-gramme dans les références, pour éviter de surévaluer les phrases répétitives.

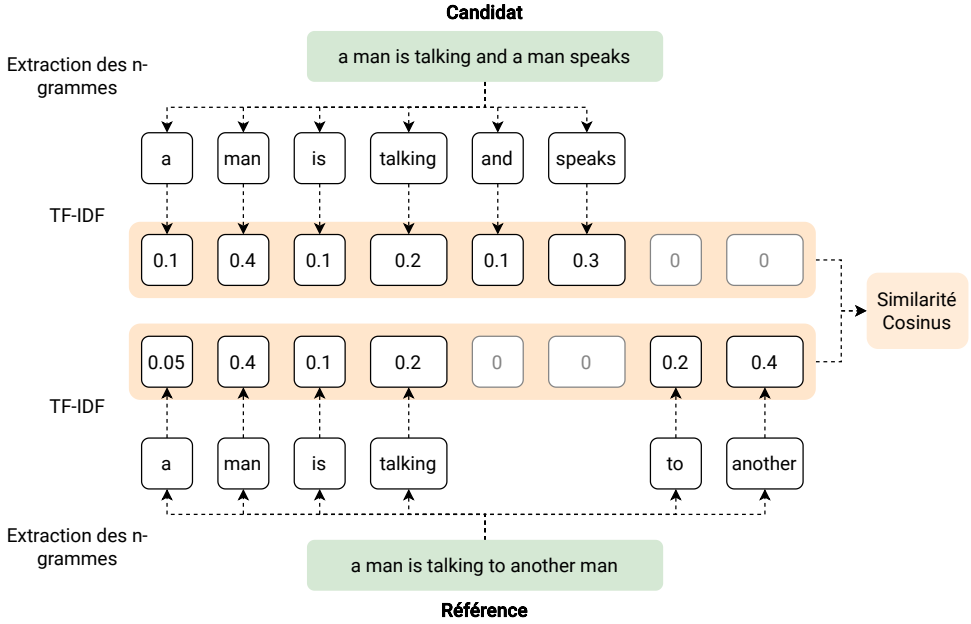


FIGURE 2.1 – Fonctionnement de la métrique CIDEr, uniquement avec des 1-grammes. On récupère en premier lieu les 1-grammes de chaque phrase, puis on calcule leur TF-IDF, qui seront vectorisés et comparés à l'aide d'une similarité cosinus.

Le facteur de pénalité est décrit par l'équation suivante :

$$\text{Pénalité}(c, r) = \exp\left(-\frac{(|c| - |r|)^2}{2\sigma^2}\right) \tag{2.8}$$

avec l'hyperparamètre σ qui vaut par défaut 6. La valeur de la pénalité est exponentiellement plus élevée lorsque la différence entre la longueur des phrases grandit.

CIDEr est la principale métrique utilisée pour comparer les systèmes de DTAI. Bien que ces valeurs soient théoriquement dans l'intervalle [0, 10], l'état de l'art sur certains jeux de données de DTAI et DTAA est souvent en dessous de 1. Et, comme les valeurs sont souvent reportées en pourcentages, ceci peut prêter à confusion. Dans la littérature, la plupart des études ne spécifient pas quelle version de CIDEr est utilisée. Cependant, les codes sources les plus utilisés en DTAI (*Microsoft COCO Caption Evaluation* [18]) et en DTAA (*Audio captioning evaluation metrics* [19]) implémentent tous deux uniquement la métrique CIDEr-D, et cela même si le code source nomme cette métrique CIDEr, ce qui est ambigu. Pour la suite de ce document, nous considérons donc uniquement la version de la métrique nommée CIDEr-D, et non la métrique CIDEr.

2.2.2 SPICE

En 2016, la métrique *Semantic Propositional Image Caption Evaluation* (SPICE, abrégé SC) [Anderson 2016] a été développée pour la DTAI avec une approche très différente de celle de CIDEr. Pour chaque phrase, la métrique détermine la nature des mots présents pour produire un

18. https://github.com/tylin/coco-caption
19. https://github.com/audio-captioning/caption-evaluation-tools

graphe sémantique représentant la scène décrite dans l'image. Plus précisément, chaque phrase est donnée en entrée d'un analyseur syntaxique probabiliste (*Probabilistic Context-Free Grammar dependency parser*) [Klein 2003] pour construire un arbre de dépendance syntaxique. Les mots de l'arbre sont associés à leur nature comme les déterminants (DT) ou les adjectifs (JJ). Les arcs de l'arbre indiquent différentes relations entre les mots tels que préposition (prep) ou objet (pobj).

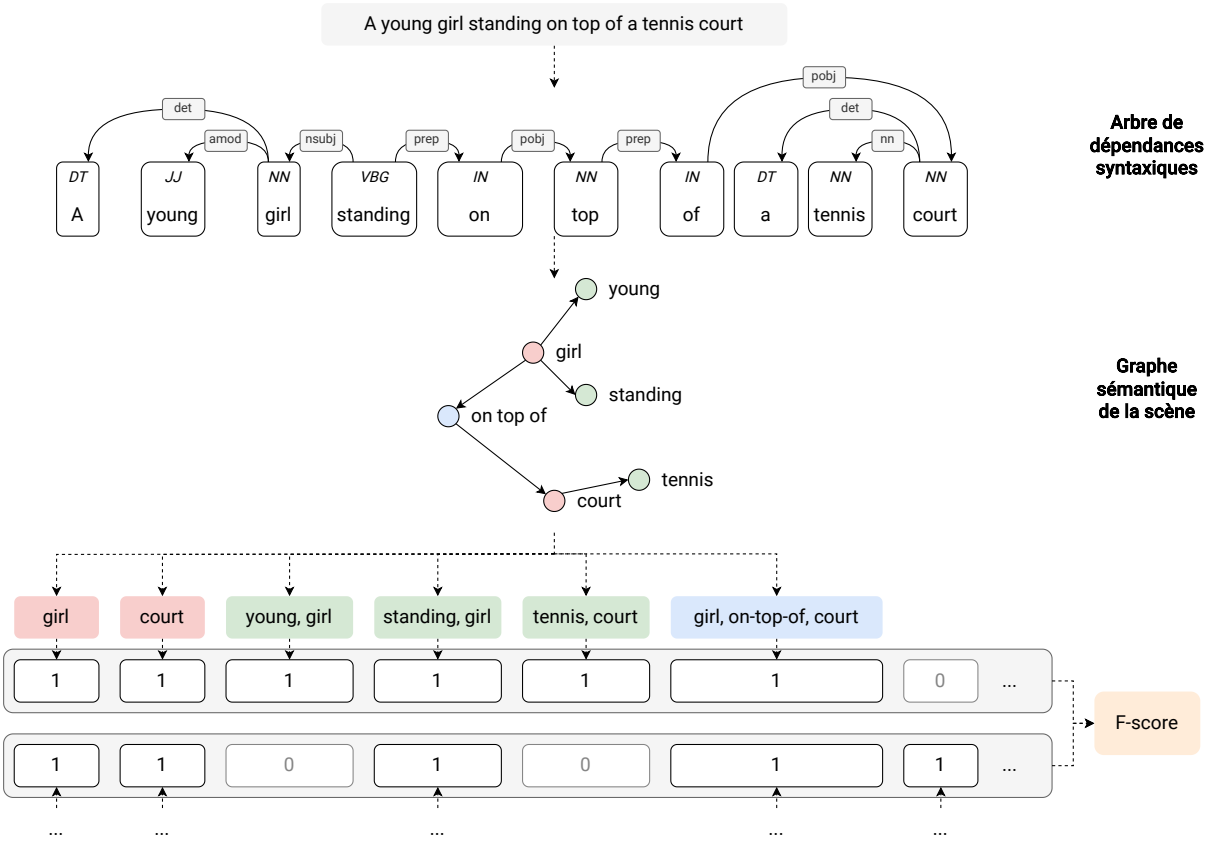


FIGURE 2.2 – Schéma du fonctionnement de la métrique SPICE, avec l'arbre de dépendances syntaxique, le graphe sémantique et la vectorisation du graphe.

Cet arbre sert d'appui pour créer une autre représentation de plus haut niveau introduite par SPICE : le graphe sémantique de la scène. Chaque nœud du graphe correspond à un objet **(rouge)**, un attribut **(vert)** ou une relation **(bleu)**. Les arcs relient un attribut avec un objet ou une relation avec un objet. Finalement, les différents arcs du graphe sont vectorisés pour être mis en correspondance avec celui d'une autre phrase à l'aide d'un F-score. Comme METEOR, SPICE utilise également de la correspondance entre synonymes. Le schéma 2.2 résume le traitement effectué par SPICE pour une phrase. Lorsque plusieurs références par fichier audio sont disponibles, le graphe sémantique du candidat sera comparé à celui de l'union de ceux des références.

Le processus complexe de traitement de texte de SPICE rend son calcul assez coûteux, ce qui explique que la métrique soit peu souvent utilisée comme fonction de coût pour un entraînement fondé sur du RL. De plus, les valeurs de SPICE sont souvent distribuées vers 0 en DTAA [Zhou 2022], ce qui est notamment dû au manque de correspondance entre les différents graphes sémantiques extraits, qui sont en plus plutôt spécifiques à l'image que l'audio [Gontier 2023].

2.2.3 SPIDEr

SPIDEr (abrégé SD) [Liu 2017] est une métrique originellement employée pour optimiser d'un modèle de DTAI sur SPICE et CIDEr-D en même temps avec du RL. SPIDEr est simplement la moyenne des valeurs de CIDEr-D et de SPICE, et elle est censée prendre en compte les avantages de ces deux métriques. Cependant, comme les scores de CIDEr-D sont dans l'intervalle [0, 10] et ceux de SPICE dans [0, 1], les valeurs de SPIDEr sont dans l'intervalle [0, 5,5] et sont donc majoritairement influencées par CIDEr-D. SPIDEr est la principale métrique utilisée pour comparer les systèmes de DTAA, notamment dû à son utilisation régulière dans les compétitions de DCASE.

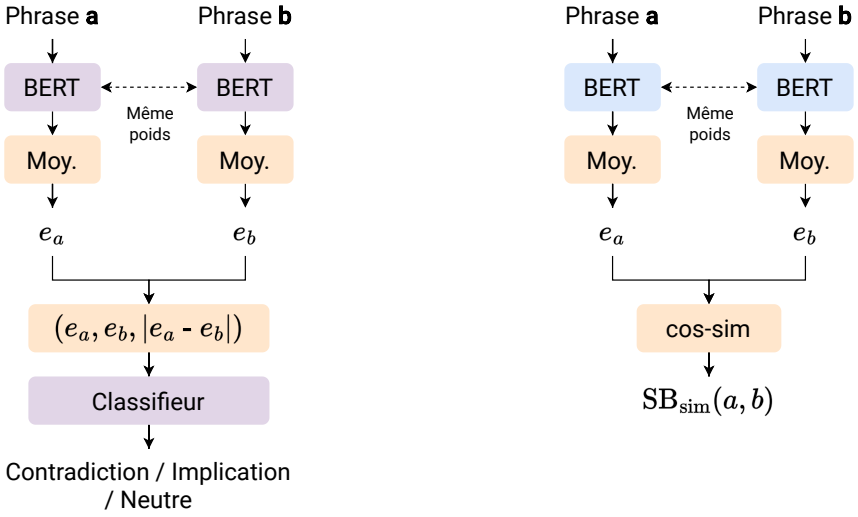
2.3 Description Textuelle Automatique de l'Audio

Lorsque la tâche de DTAA est apparue en 2017, elle a repris les métriques précédemment employées en DTAI, en mettant en avant le CIDEr-D ou le SPIDEr. Plusieurs études [Emmanouilidou 2023, Gontier 2023] ont par la suite mis en doute l'utilisation des n-grammes pour comparer les phrases. Cela est notamment dû au manque de correspondance de certaines phrases différentes mais décrivant le même événement sonore, comme *« A man is speaking »* avec *« Low-pitched voice is heard »*. Ces deux dernières phrases sont par exemple considérées comme peu similaires, même en prenant en compte des synonymes, car elles ne partagent aucun mot en commun. Pour atteindre un plus haut niveau d'abstraction et pour être mieux corrélées avec le jugement humain, de nombreuses nouvelles métriques utilisent des modèles pré-entraînés sur le texte, permettant de faire correspondre le sens d'une phrase plutôt que les mots.

2.3.1 Modèles et métrique SBERT

Les modèles *Sentence-BERT* (SBERT) sont des réseaux de neurones profonds visant à produire un vecteur de taille fixe permettant de comparer le sens d'une phrase avec d'autres. L'architecture de ces modèles est fondée sur celle des modèles BERT, qui sont des modèles de type *Transformer* entraîné sur du texte. Les modèles SBERT moyennent les sorties de l'avant-dernière couche de BERT pour produire un vecteur unique pour chaque phrase. Les modèles SBERT sont entraînés comme des réseaux de neurones siamois. Pendant l'apprentissage, deux phrases sont fournies en entrée du réseau pour produire deux vecteurs e_a et e_b. Ces deux vecteurs sont concaténés entre eux et avec leur différence $|e_a - e_b|$ et donnés à une couche de classification linéaire qui doit prédire la nature de la relation entre les deux vecteurs. Par défaut, le modèle utilisé est le `paraphrase-TinyBERT-L6-v2` [20]. Il a été entraîné sur deux jeux de données de texte : *Stanford Natural Language Inference* [Bowman 2015] et *Multi-Genre Natural Language Inference* [Williams 2018]. Ces jeux de données contiennent des paires de phrases annotées selon trois types de relations : implication, contradiction ou neutralité. Pour comparer deux phrases, la fonction de similarité cosinus (voir équation 2.7) est utilisée entre les deux vecteurs e_a et e_b, car c'est une fonction simple à calculer. La figure 2.3 présente le fonctionnement global de l'entraînement et de l'inférence des modèles SBERT. Dans ce manuscrit, la métrique est nommée *Sentence-BERT similarity* (SB_{sim}), et représente la similarité cosinus utilisant un modèle SBERT.

20. https://huggingface.co/sentence-transformers/paraphrase-TinyBERT-L6-v2



(a) Méthode d'entraînement du modèle SBERT.

(b) Méthode de test sur deux phrases avec le modèle SBERT.

FIGURE 2.3 – Méthodes d'entraînement et de test pour le modèle SBERT.

2.3.2 FENSE

La métrique *Fluency ENhanced Sentence-bert Evaluation* (FENSE, abrégé FNS) [Zhou 2022] repose sur deux idées :

1. l'utilisation des n-grammes pour comparer des phrases ne permet pas de se rendre compte si une phrase est similaire à une autre pour une tâche de description ;
2. les systèmes de DTAA commettent des erreurs de fluence (erreurs syntaxiques ou répétition de n-grammes) qui ne sont pas ou peu pénalisées par les métriques.

Partant de ces deux constats, ils proposent d'utiliser la métrique $\mathrm{SB}_{\mathrm{sim}}$ combinée avec un modèle de détection automatique d'erreurs de fluence. Les auteurs ont entraîné eux-mêmes ce détecteur en générant un corpus de descriptions contenant certains types d'erreurs observés dans les candidats générés par les systèmes de DTAA. Le détecteur d'erreurs de fluence est un modèle fondé sur BERT avec une couche qui moyenne les représentations et une autre de classification entraînée à détecter cinq types d'erreurs :

— Phrase incomplète (« *a woman is giving a speech and* ***a*** ») ;
— Répétition d'événement (« ***music plays*** *followed by* ***music playing*** ») ;
— Répétition d'adverbe (« *sheep bleats* ***nearby*** *several times* ***nearby*** ») ;
— Conjonction manquante (« *people speaking (and) a train horn blows* ») ;
— Verbe manquant (« *food sizzles and a pan* »).

Le modèle possède six neurones en sortie, dont cinq correspondent à chaque type d'erreur et un dernier indique si au moins une erreur est présente. Il a été entraîné en utilisant la fonction de coût d'entropie croisée binaire standard. À l'aide de ce modèle, la métrique FENSE considère qu'une erreur est présente si la probabilité de la sortie du neurone qui détecte au moins une erreur est supérieure à 90%. Si la probabilité est en dessous de ce seuil, le score FENSE d'une phrase sera le même que celui de $\mathrm{SB}_{\mathrm{sim}}$. Sinon, il sera égal à un dixième du score donné par $\mathrm{SB}_{\mathrm{sim}}$. La métrique pénalise donc fortement les erreurs de fluence détectées, ce qui encourage à créer des phrases syntaxiquement correctes et sans trop de répétitions. À noter que si le modèle

de DTAA produit une phrase vide, il n'y a pas d'erreur détectée, ce qui peut parfois donner un score FENSE supérieur à une phrase non vide avec une erreur de fluence. De plus, nous avons remarqué que le détecteur d'erreur n'est pas toujours capable de repérer certaines répétitions d'événement, comme dans *« cars are driving by and a car is passing by. »*. Dans la suite de ce manuscrit, nous reporterons pour certains sous-ensembles uniquement le taux d'erreur de fluence détecté (*Fluency Error Rate*, abrégé FER).

Pour comparer leur métrique par rapport aux autres, les auteurs ont proposé le *FENSE benchmark*, pour estimer la capacité de chaque métrique à évaluer la qualité d'une phrase dans quatre cas différents :

- Humain-correct avec deux phrases écrites par des humains et correspondant toutes deux à l'audio ;
- Humain-incorrect avec deux phrases écrites par des humains, mais l'une ne correspond pas à l'audio ;
- Humain-machine avec une phrase écrite par un humain et une autre par un système automatique, toutes deux correspondant à l'audio ;
- Machine-machine avec deux phrases générées automatiquement et correspondant toutes deux à l'audio.

Chaque paire de phrases fut annotée par des humains qui devaient choisir laquelle des deux phrases ils préféraient avec l'audio correspondant. Chaque audio et référence humaine provient des jeux de données AC-test ou CL-eval. Une métrique est considérée comme plus performante si, pour chaque paire de phrases, elle arrive à donner un score plus élevé à celle préférée par les humains que pour l'autre. La métrique FENSE est comparée aux métriques classiques de DTAI, mais aussi à quelques métriques utilisées en résumé de texte comme BERTScore [Zhang* 2020] et BLEURT [Sellam 2020]. Ces deux dernières métriques emploient toutes deux des représentations issues de modèles de texte pré-entraînés. Cependant, nous ne détaillerons pas ces dernières métriques, car elles n'ont ni été conçues pour la DTAA, ni été utilisées pour comparer des systèmes de DTAA à notre connaissance. Les exactitudes (*accuracies*) de chaque métrique sont données en annexe dans le tableau C.1. Les métriques utilisant des modèles de texte pré-entraînés surpassent la plupart des autres métriques sur les deux jeux de données. La métrique FENSE est globalement supérieure aux autres métriques sur cette évaluation, notamment grâce à sa capacité à choisir la meilleure phrase dans le cas machine-machine. Cette différence s'explique notamment par le détecteur d'erreur de FENSE, car la métrique SB_{sim} qui ne l'intègre pas est bien moins efficace pour ces cas (11,0% de perte d'exactitude). Cela semble signifier que les humains ont évité les phrases qui contenaient des erreurs de fluence pour choisir la meilleure phrase parmi deux phrases générées automatiquement.

La métrique FENSE semble donc très pertinente pour évaluer les phrases de DTAA. Cependant, le modèle utilisé n'a pas été explitement fait pour des descriptions audio, ce qui le rend moins adapté pour certaines caractéristiques sonores. Par exemple, le score FENSE du candidat *« a man is speaking »* avec la référence *« a man is speaking and echoing »* sera 83,5%, alors qu'avec un candidat a priori plus précis comme *« a man is speaking in a large room »* le score sera 69,6%. Il est donc possible que la similarité cosinus des représentations des phrases sous-estiment ou surestiment certaines phrases.

2.3.3 SPIDEr-FL

SPIDEr-FL [21] (SD_{FL}) est une métrique introduite pour la tâche de DTAA de la compétition DCASE en 2023. Elle combine SPIDEr et le modèle de détection d'erreur issu de FENSE de la même manière, en divisant les scores SPIDEr par dix lorsqu'une erreur de fluence est détectée. Cette métrique a été introduite principalement pour pénaliser les systèmes utilisant l'apprentissage par renforcement (décrit dans la section 1.4.2) sur CIDEr-D, qui augmente artificiellement le score SPIDEr, mais amène le système à générer des phrases syntaxiquement incorrectes et répétitives. Cependant, la métrique reste cependant moins efficace que FENSE sur le *FENSE benchmark*, comme on peut le voir dans le tableau C.1 en annexe.

2.3.4 CB-Score

Si la plupart des métriques précédentes se focalisent sur la comparaison des phrases, les auteurs de [Martín-Morató 2022] ont proposé une approche différente, se focalisant uniquement sur les événements sonores. La métrique *Content-Based evaluation of captions* (CB-Score) extrait les différents événements sonores définis par l'ontologie des classes d'AudioSet (AS) à partir des phrases à l'aide d'une suite de traitements de texte (*tokenization*, lemmatisation, correspondance des synonymes et des hyperonymes). Une fois extraits, les événements sonores sont comparés à l'aide d'un score de pertinence (*relevance score*) :

$$\mathrm{Rel}_i = \frac{\mathrm{Freq}_i}{\sum_{j=1}^{M} \mathrm{Freq}_j} \tag{2.9}$$

avec i l'index d'un événement sonore présent dans le candidat ou les références, Freq_i la fréquence de cet événement i dans les références et M le nombre d'événements distincts dans le candidat et les références. Si K événements sonores sont reconnus dans le candidat c, on récupère le top-K scores de pertinence des événements du candidat c et des références R (respectivement notés $top(c)$ et $top(R)$).

$$\text{CB-Score}(c, R) = \frac{\sum_{i \in top(c)} \mathrm{Rel}_i}{\sum_{j \in top(R)} \mathrm{Rel}_j} \tag{2.10}$$

D'une manière similaire à la métrique de Précision, le CB-Score ne prend en compte que les vrais et faux positifs, mais pas les événements des références manqués dans le candidat. La métrique ignore également les relations entre les événements, les erreurs de fluence ou encore les attributs donnés à ces événements, ce qui limite son utilisation pour comparer des systèmes de DTAA qui doivent aller plus loin que la simple détection d'événements sonores. À notre connaissance, la métrique n'a pas été comparée avec les autres sur le *FENSE benchmark*.

2.3.5 SPICE+

SPICE+ [Gontier 2023] est une autre métrique dédiée à la DTAA, visant à fusionner les avantages des métriques SPICE et FENSE. L'objectif est d'obtenir une métrique plus facilement interprétable à l'aide des graphes de scènes, tout en obtenant un score comparable à celui de FENSE sur le *FENSE benchmark*. Le détail des scores sur ce dernier est donné en annexe dans le tableau C.1, et montre que la métrique est assez proche de FENSE dans toutes les catégories. Pour atteindre leur objectif, les auteurs de SPICE+ ont modifié SPICE pour prendre en compte des expressions plus spécifiques à l'audio que l'image (comme *« rain continues to*

21. https://dcase.community/challenge2023/task-automated-audio-captioning#evaluation

pour », *« group of people »*...), substitué l'analyseur syntaxique probabiliste par un modèle de langage, et ont remplacé la correspondance par synonyme par une distance cosinus fondée sur les vecteurs produits par un modèle SBERT. La métrique utilise également le détecteur d'erreurs de FENSE pour pénaliser les erreurs de fluence.

2.3.6 ACES

La métrique *Audio Captioning Evaluation on Semantics of Sound* (ACES) [Wijngaard 2023] est une métrique similaire sur certains poins à SPICE+, mais visant à s'inspirer de la recherche en sciences cognitives pour se focaliser sur certains éléments descriptifs du son. Ces éléments sont catégorisés en plusieurs types : sources des événements (le « qui » ou « quoi »), actions ou mécanismes liés à la génération du son (le « comment ») et contexte spatial ou temporel (le « quand »). Pour créer une métrique évaluant ces éléments, les auteurs ont annoté les mots du jeu de données CL par ces différentes catégories, puis ils ont entraîné un modèle à les reconnaître automatiquement. Il existe 13 catégories utilisées par ACES, et les auteurs ont également proposé une seconde version d'ACES plus rapide à calculer qui n'en utilise que 10. Elles sont respectivement nommées ACES_{13} et ACES_{10}. Pour chaque candidat et référence, la métrique combine la similarité cosinus entre les représentations des mots issues du modèle de détection de catégories et le F-score entre des catégories du candidat et des références, respectivement notés $L(c)$ et $L(r)$.

$$\text{ACES}(c, R) = \frac{1}{2|R|} \sum_{r \in R} \text{cos-sim}(c, r) + \text{F1}(L(c), L(r)) \tag{2.11}$$

Cette métrique permet ainsi de mieux comprendre ce que les systèmes de DTAA sont capables de reconnaître et de décrire dans les candidats. Cependant, elle est bien moins efficace sur le *FENSE benchmark* que FENSE ou SPICE+, car elle n'intègre pas de détecteur d'erreurs de fluence et n'utilise pas un modèle de texte aussi grand que SBERT. Comme pour les autres métriques, les scores détaillés sont en annexe dans le tableau C.1.

2.3.7 SBF

Récemment, une nouvelle métrique nommée *Similarity-Based F-score* (SBF) a été proposée dans l'étude [Mahfuz 2023a]. Celle-ci combine plusieurs idées des métriques précédentes, en cherchant à détecter les événements sonores présents dans les candidats et références pour ramener le problème à une tâche de classification audio. D'une manière similaire à SPICE+, la métrique emploie des modèles de textes pré-entraînés (`paraphrase-TinyBERT-L6-v2` et `all-MiniLM-L6-v2`) sur les propositions (*phrases*) extraites des descriptions à partir de la catégorie grammaticale des mots (*part-of-speech*). Les modèles sont aussi exploités pour obtenir une représentation de chaque événement sonore issue de l'ontologie d'AS, en utilisant les noms des classes. Chaque proposition des phrases est associée avec un vecteur de représentation à taille fixe qui sera comparé avec les vecteurs qui représentent les événements sonores par une fonction de similarité cosinus et d'un seuil prédéfini nommé `tag_t`. On obtient ainsi la liste des événements sonores d'AS décrit par une phrase, qui seront comparés à l'aide de leurs représentations à l'aide d'un second seuil appelé `sim_t` pour définir les vrais positifs, faux positifs et faux négatifs des événements sonores d'un candidat selon une référence. Les auteurs ont également ajouté un filtre pour supprimer les redondances des événements trouvés à l'aide d'un troisième seuil désigné par `rep_t`.

Par exemple, le tableau 2.1 montre comment un candidat et une référence et leurs propositions et leurs événements sonores extraits. Dans cet exemple, on obtiendra le vrai positif *« Bell »*, le faux positif *« Bird »* et le faux négatif *« Conversation »*. On utilisera ensuite la formule classique du F-score pour calculer le score, qui sera 0,5 dans cet exemple.

TABLE 2.1 – Exemple d'événements sonores extraits par la métrique SBF.

Type	**N°**	**Contenu**
Phrases	C1	A bell is ringing while birds are chirping in the background
	R1	A bell rings while people talk in a courtyard.
Propositions	C1	a bell is ringing ; birds are chirping in the background
	R1	a bell rings ; people talk in a courtyard
Événements	C1	Bell ; Bird
	R1	Bell ; Conversation

Les trois seuils `tag_t`, `sim_t` et `rep_t` ont été trouvé expérimentalement et sont respectivement égaux 0,4, 0,45 et 0,45, pour correspondre au mieux au jugement humain sur le *FENSE benchmark*. Nous donnons les résultats de la métrique sur ce dernier dans le tableau C.1 en annexe, avec deux modèles de texte différents. Comme pour ACES, la métrique n'est pas aussi performante que le FENSE sur la majorité des scores. Ceci est probablement dû au manque de détecteur d'erreur de fluence et au fait que la métrique ne s'occupe que des événements sonores et non de leurs relations ou de la structure de la phrase. On peut cependant noter que la métrique est légèrement meilleure que FENSE dans le cas Humain-machine pour AC-test (92,1% au lieu de 91,6%).

2.3.8 S2V

Une autre métrique nommée s2vscore (S2V) [Bhosale 2023] se focalise également sur les événements sonores décrits dans les candidats et références, mais d'une manière différente de CB-score ou de SBF. Contrairement à toutes les autres métriques, cette dernière n'utilise pas des représentations issues d'un modèle de texte pré-entraîné, mais des représentations issues d'un modèle de détection d'événements sonores contrôlée par du texte (*Text-to-Audio Grounding*). Pour rappel, cette tâche a pour but de créer des systèmes capables de localiser un événement sonore décrit par une phrase dans un fichier audio. Un tel modèle peut produire des représentations mélangeant des informations audio et texte en même temps, ce qui permet de faire correspondre des événements similaires d'un point de vue d'un auditeur mais différents dans les descriptions (par exemple, de la pluie forte et un bruit de ventilateur). Le modèle est entraîné sur le jeu de données AudioGrounding [Xu 2021d], qui contient 4590 fichiers audio.

En premier lieu, cette métrique extrait les propositions des descriptions à l'aide d'une bibliothèque nommée Textacy qui repose sur les catégories grammaticales des mots. Nous notons le nombre de proposition extraires P_c pour un candidat et P_r pour une référence. Puis, les propositions des phrases sont données séparément au modèle de détection d'événements sonores avec l'audio, pour produire des séquences de probabilité d'occurrence dépendant de la taille de l'audio (de taille T_a). Chaque séquence sera comparée avec celles de l'autre phrase par une

fonction de similarité cosinus pour former une matrice de similarité, de taille $P_c \times P_r$. Pour finir, un F-score sera appliqué sur cette matrice puis moyenné pour obtenir le score final S2V. La métrique pondère donc la durée de chaque événement sonore dans le fichier audio, et permet également de faire correspondre des descriptions qui décriraient un événement sonore similaire à celui présent dans les références.

Pour évaluer leur métrique par rapport aux autres, les auteurs ont appliqué un test ressemblant vaguement au *FENSE benchmark*, que nous nommons le *S2V benchmark*. Ce dernier utilise les taux de corrélation entre des paires de descriptions positives et négatives. Les paires positives sont extraites de CL à partir des multiples références de chaque audio, et les paires négatives sont générées en choisissant une description d'un fichier audio différent. Les résultats détaillés sont donnés dans le tableau C.2 en annexe. Leur métrique est comparée avec celles héritées de la DTAI et avec la métrique BERTScore. Elle obtient une meilleure corrélation que toutes les autres métriques sur les paires positives et une corrélation légèrement inférieure à celle du BERTScore sur les paires négatives.

2.4 Diversité du texte

La majorité des métriques précédentes mesurent la fidélité des phrases générées par rapport aux références de chaque fichier audio. Cependant, ces métriques ne tiennent pas ou peu compte du niveau de langue des phrases produites, qui peuvent être parfois assez génériques, répétitives ou vagues. Plus marginalement, quelques études [Mei 2022c] monitorent également d'autres aspects liés à la qualité du texte produit, comme la taille du vocabulaire de mots (au niveau du corpus des candidats générés). Le vocabulaire notée « **Vocab** » dans ce manuscrit, mais est parfois appelée « Uniq » ou « #Words » dans la littérature. Cette taille peut être comparée à celle des références humaines, qui sont bien souvent beaucoup plus variées. D'autres métriques se focalisent sur la diversité des n-grammes des candidats, comme la métrique **div-n** [Li 2016] (aussi appelée distinct-n) qui calcule le ratio entre le nombre de n-grammes uniques et le nombre total de n-grammes. Il existe plusieurs versions de diversité qui varient en fonction des études, et la version utilisée pour les systèmes de DTAA se focalise sur la diversité des phrases indépendamment les unes des autres. **Self-BLEU** [Zhu 2018] (aussi appelé mBLEU pour *mutual BLEU*) propose d'utiliser la métrique BLEU pour étudier la similarité des candidats générés entre eux. Contrairement à BLEU, la métrique est donc meilleure si le score est plus faible. Elle peut être appliquée pour un système de DTAA qui génère plusieurs candidats par fichier audio, à l'aide d'un algorithme de recherche en faisceau ou d'une méthode de décodage stochastique.

2.5 Score humain

Toutes les métriques présentées ici évaluent les systèmes de génération de texte d'une manière différente, avec des intervalles de valeurs variés en théorie, mais aussi en pratique. En effet, le score maximal de chaque métrique est atteint lorsque le candidat est le même que l'une des références et que toutes les références sont égales entre elles, mais il n'est pas nécessairement atteint pour un candidat écrit par un humain.

Pour estimer le score que donnerait un système qui générerait des phrases du niveau d'un être humain, nous pouvons croiser les références de chaque audio entre elles. Comme chaque ensemble de test contient cinq phrases par audio, l'une des cinq phrases est exclue aléatoirement

pour l'utiliser en tant que candidate tout en gardant les quatre autres comme références. Le processus est appliqué pour tout un ensemble de test pour calculer les scores globaux de chaque métrique. Il peut également être répété plusieurs fois pour utiliser plusieurs phrases différentes en tant que candidat, puis moyenné pour chaque métrique. Cette valeur est donc nommée le score « humain », mais il faut noter qu'il est parfois également appelé le score de cross-référence ou encore le score de consensus, à différencier du CB-Score. Après avoir échantillonné une référence r d'une liste de références R d'un fichier audio, le score humain pour une métrique sera donc défini par l'équation 2.12 :

$$\text{Humain}(r, R) = \text{Métrique}(r, \{R \setminus r\}) \tag{2.12}$$

TABLE 2.2 – Scores humains des métriques héritées de la DTAI pour les différents sous-ensembles de validation et de tests d'AudioCaps (AC) et de Clotho (CL) (en %). Les métriques sont, dans l'ordre : BLEU1 (B1), BLEU2 (B2), BLEU3 (B3), BLEU4 (B4), ROUGE-L (RG), METEOR (ME), CIDEr-D (CD), SPICE (SC) et SPIDEr (SD).

Jeu	**B1**	**B2**	**B3**	**B4**	**RG**	**ME**	**CD**	**SC**	**SD**
AC-test	65,4±1,2	48,7±1,2	37,2±1,0	28,9±0,8	49,2±0,8	28,8±0,4	90,1±2,1	21,7±0,6	55,9±1,2
AC-val	63,3±2,4	48,0±2,4	36,9±2,5	28,5±2,6	53,9±1,3	29,8±0,9	111,8±6,8	25,9±0,6	68,9±3,6
CL-eval	65,5±0,4	49,7±0,6	39,5±0,7	32,1±0,7	50,9±0,2	30,5±0,2	90,3±1,2	23,1±0,2	56,7±0,5
CL-val	63,1±1,2	47,0±1,3	36,6±1,2	29,2±1,1	48,0±1,4	28,9±0,7	84,3±2,9	21,7±0,3	53,0±1,6

TABLE 2.3 – Scores humains des métriques supplémentaires pour les différents sous-ensembles de validation et de tests d'AudioCaps (AC) et de Clotho (CL) (en %). Les métriques sont, dans l'ordre : Similarité SBERT (SB_{sim}), Taux d'erreur de fluence (FER), FENSE (FNS), SPIDEr-FL (SD_{FL}), Vocabulaire (Vocab) et Taille des phrases (#Sent).

Jeu	**SB_{sim}**	**FER**	**FNS**	**SD_{FL}**	**Vocab**	**#Sent**
AC-test	68,2±0,3	0,5±0,1	68,0±0,2	55,7±1,1	944,0±18,6	10,3±0,1
AC-val	66,3±0,6	0,6±0,5	66,0±0,7	68,6±3,7	687,8±20,1	8,3±0,2
CL-eval	57,9±0,2	0,8±0,3	57,4±0,2	56,3±0,6	1818,2±17,1	11,3±0,1
CL-val	57,0±0,4	0,7±0,3	56,6±0,3	52,7±1,4	1726,2±28,7	11,4±0,1

Les scores humains de plusieurs métriques sont donnés dans les tables 2.2 et 2.3, avec les jeux de test et de validation contenant 4560 phrases pour AC-test, 2320 pour AC-val, 5225 pour CL-eval et 5225 pour CL-val. Nous avons calculé les scores de chaque métrique pour laquelle nous disposons du code source public, ainsi que des métriques Vocab et #Sent qui sont très simples à calculer. Les valeurs du tableau révèlent que les phrases humaines sont loin d'atteindre les valeurs théoriques maximales de chaque métrique. Par exemple, le score SPIDEr atteint 55,9% sur AC-test, alors que sa valeur théorique maximale est 550%.

Les scores des métriques héritées de la traduction et du résumé automatiques sont globalement assez similaires entre chaque sous-ensemble. CL-eval contient un peu plus de long n-grammes en

commun, mais c'est le sous-ensemble AC-val qui contient les sous-séquences communes les plus similaires. Pour les métriques de DTAI, la métrique CIDEr-D varient énormément sur chaque jeu, allant jusqu'à 111,8% sur AC-val, et il influence grandement le score SPIDEr. Les scores SPICE sont cependant assez similaires entre chaque jeu. Le score SPIDEr sur AC-val est plus élevé que sur les autres sous-ensembles (68,9%), notamment car les phrases sont plus courtes (8,3 mots par phrase), ce qui augmente drastiquement le score SPIDEr si un n-gramme long se retrouve dans plusieurs références d'un même audio.

De plus, nous avons reporté les tailles des vocabulaires et la taille moyenne des phrases dans le second tableau. Ces valeurs sont plus élevées pour CL-eval que AC-test, malgré un nombre similaire de fichiers audio (environ 1000 chacun), ce qui semble indiquer que le test de CL est plus divers que dans AC. On peut également noter que les scores SB_{sim} et FENSE sont plus élevés sur AC que CL, ce qui pourrait être dû au processus d'annotation qui a encouragé les humains à diversifier les différentes descriptions de chaque audio. Contrairement à ce que l'on pourrait s'attendre, le taux d'erreur de fluence n'est pas nul, notamment à cause de faux-positifs liés au modèle de détection d'erreur, avec certaines phrases comme *« insects buzz and fly »* ou *« rain is falling down on a tin roof »*.

2.6 Bilan

Dans ce chapitre, nous avons listé et détaillé les différentes métriques utilisées pour comparer les systèmes de DTAA. Nous avons décrit celles héritées de la traduction automatique, de la synthèse de texte, de la DTAI mais aussi celles spécifiquement conçues pour la DTAA. Nous avons également décrit quelques métriques dédiées à la génération de texte, qui sont encore marginalement utilisées. Bien que plusieurs d'entre elles semblent corrélées, les métriques prennent en compte des aspects très différents des phrases, comme les synonymes pour METEOR, la sémantique pour SB_{sim} ou la fluence des phrases pour FENSE. Malgré tout, la métrique SPIDEr reste cependant la principale métrique de comparaison des systèmes de DTAA dans la littérature. Le tableau 2.4 donne un récapitulatif des différentes métriques présentées dans ce chapitre.

La recherche pour évaluer les systèmes de DTAA reste très active, avec de nouvelles métriques qui se focalisent sur d'autres aspects comme l'interprétabilité (SPICE+) ou sur les événements sonores (CB-Score). L'apparition du *FENSE benchmark* permet aussi d'évaluer la capacité de ces métriques à discriminer des phrases similaires, pour mieux se rapprocher du jugement humain. Cependant, ces nouvelles métriques ne prennent pas encore en compte certains aspects des phrases générées, comme la diversité du vocabulaire, qui pourrait être intéressante pour évaluer le niveau de langue des systèmes de DTAA.

En DTAA, il est commun de calculer le score de nombreuses métriques qui ont été présentées dans ce chapitre. Cependant, il semble que plusieurs de ces métriques soient fortement liées, et certaines d'entre elles prennent en compte des aspects similaires comme le recouvrement de n-grammes entre les candidats et références. Il est donc légitime de s'interroger sur quels métriques sont les plus pertinentes à conserver pour l'évaluation. La métrique SPIDEr semble nécessaire pour comparer les systèmes avec la littérature, mais elle ne prend pas en compte de nombreux aspects du texte, comme les erreurs de fluence par exemple. De plus, nous verrons dans le chapitre 4 qu'elle peut être assez sensibles aux mots présents dans les descriptions. La métrique FENSE semble être la meilleure métrique d'après le *FENSE benchmark*, grâce aux

Table 2.4 – Tableau récapitulatif des métriques utilisées ou conçues pour la DTAA. Les acronymes utilisés sont : DNN pour *Deep Neural Network*, Trad. pour Traduction, entr. pour entraîné, Repré. pour représentations (*embeddings*) et n-gram pour n-gramme.

Nom	**Année**	**Origine**	**Intervalle**	**DNN pré-entr.**	***Open Source***	**Fondé sur**
BLEU	2001	Trad.	[0 ; 1]	Non	Oui	N-gram
ROUGE-L	2004	Résumé	[0 ; 1]	Non	Oui	LCS
METEOR	2014	Trad.	[0 ; 1]	Non	Oui	Unigramme
CIDEr-D	2015	DTAI	[0 ; 10]	Non	Oui	N-gram
SPICE	2016	DTAI	[0 ; 1]	Non	Oui	Graphe scène
SPIDEr	2017	DTAI	[0 ; 5,5]	Non	Oui	N-gram+graphe scène
SB_{sim}	2021	DTAA	[-1 ; 1]	Oui	Oui	Repré.
FER	2021	DTAA	[0 ; 1]	Oui	Oui	Fluence
FENSE	2021	DTAA	[-1 ; 1]	Oui	Oui	Repré.+fluence
SPIDEr-FL	2023	DTAA	[0 ; 5,5]	Oui	Oui	N-gram+graphe scène+fluence
CB-Score	2022	DTAA	[0 ; 1]	Non	Non	Événements sonores
SPICE+	2023	DTAA	[-1 ; 1]	Oui	Non	Repré.+graphe scène+fluence
ACES	2023	DTAA	[0 ; 1]	Oui	Oui	Repré.+catégories
SBF	2023	DTAA	[-1 ; 1]	Oui	Non	Événements sonores
S2V	2023	DTAA	[0 ; 1]	Oui	Non	Repré.
Vocab	N/A	DTAA	[0 ; $+\infty$[	Non	Oui	Mot

représentations texte et au détecteur d'erreur de fluence. Cependant, il faut rappeler que ces représentations n'ont pas été conçues pour des descriptions audio, et on peut se demander si ils représentent bien certaines caractéristiques sonores précises. Une métrique liée à la diversité des phrases comme Vocab peut également être utilisée pour donner un aperçu de la diversité d'un système de DTAA, car ni SPIDEr ni FENSE ne semble couvrir cet aspect. À l'heure de l'écriture de cette thèse en novembre 2023, de nouvelles métriques ont été développées et pourraient se rajouter à cet ensemble. En effet, SPICE+ pourrait remplacer FENSE et permettre une meilleure interprétabilité grâce au graphe de scène adapté à l'audio, tout en gardant les représentations texte et le détecteur d'erreur. Puis, une métrique se focalisant sur les événements sonores pourrait également être utilisée, comme SBF qui permet de comparer les modèles de DTAA sur cet aspect uniquement et même de les confronter aux systèmes de classification audio sur un jeu comme AS. Dans le chapitre 4, nous porterons notre attention sur la métrique SPIDEr, qui reste la principale métrique utilisée pour comparer les systèmes de DTAA.

Chapitre 3

Création d'un premier système de DTAA

Dans ce chapitre, nous décrivons en détail la création de nos premiers systèmes de DTAA, qui s'inspirent des différents systèmes et architectures proposés dans la littérature en 2020 et 2021. Tout d'abord, nous présentons le *Listen-Attend Spell* (LAS), un encodeur-décodeur composé de cellules récurrentes et à l'origine conçu pour la RAP. Puis, nous décrivons le CNN10-teller, qui remplace une partie du modèle précédent par un encodeur convolutif pré-entraîné sur une tâche de classification d'événements sonores. Ensuite, nous remplaçons le décodeur récurrent par un décodeur *Transformer* pour créer le modèle CNN10-trans, qui est le plus performant de nos systèmes de ce chapitre. Nous détaillons également le processus de prétraitement des données audio et texte, ainsi que les hyperparamètres de notre modèle. De plus, nous rapportons les résultats des architectures qui ont été testées, et comment ces modèles se comparent à l'état de l'art de 2021. Nous terminons par quelques exemples de sorties de ces modèles, pour faire nos premières conclusions qualitatives sur la pertinence des phrases produites.

Sommaire

3.1 Architectures

3.1.1 *Listen, Attend and Spell*

Pour créer notre propre système de DTAA, nous nous sommes inspirés des architectures utilisées en transcription de parole qui traitent également les deux modalités audio et texte. En premier lieu, nous avons utilisé le modèle *Listen-Attend Spell* (LAS) [Chan 2016], conçu à l'origine pour transcrire la parole audio en séquence de caractères. Il est composé d'un encodeur audio (nommé *Listener*), qui est un réseau pyramidal BiLSTM qui produit une représentation de l'audio sous forme d'une séquence. Cette dernière est ensuite exploitée par le décodeur (nommé *Speller*) qui est un réseau LSTM utilisant un mécanisme d'attention. Ce modèle a été implanté par mon codirecteur de thèse Thomas Pellegrini, dans le cadre d'une participation à la première édition de la tâche de DTAA dans DCASE en 2020 [Pellegrini 2020], avant le début de ma thèse.

3.1.2 *Listener*

Le terme « pyramidal » du *Listener* est ici employé pour désigner la réduction de la séquence provoquée par la concaténation des états cachés de la séquence en sortie des BiLSTM. Autrement dit, chaque état caché d'une cellule BiLSTM est concaténé avec celui de son voisin, diminuant la taille de la séquence en sortie par deux pour chaque couche de l'encodeur. Le fonctionnement du *Listener* est illustré sur le schéma 3.1. En pratique, chaque cellule est composée d'un *Dropout* [Srivastava 2014] avec une probabilité de 10%. À l'exception de la première, chaque cellule prend en entrée un vecteur de taille 256 pour produire en sortie un vecteur de taille 128, permettant de garder la même dimension en sortie qu'en entrée. La première couche prend en entrée une séquence de vecteurs comprenant chacun des 64 coefficients log-Mel.

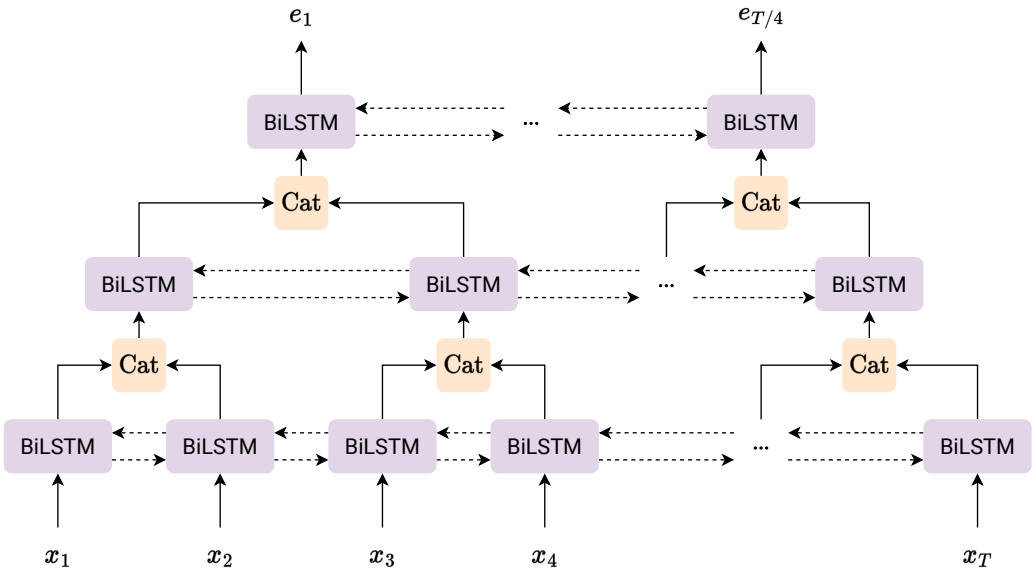


FIGURE 3.1 – Fonctionnement du *Listener* possédant trois couches BiLSTM pour une entrée x de taille T et une sortie e de taille $\frac{T}{4}$.

3.1.3 *Speller*

Le *Speller* est un décodeur récurrent auquel est attaché un mécanisme d'attention. Il est composé d'une couche de représentation (*Embedding*), suivie de deux couches LSTM unidirectionnelles, d'une couche linéaire, d'un mécanisme d'attention et d'une autre couche linéaire qui produit la distribution de probabilité du prochain mot. Le modèle ingère en entrée le mot précédent w_{i-1} et la représentation audio produite par l'encodeur, pour produire en sortie le mot suivant w_i ainsi que le contexte c_i et les états cachés de chaque cellule LSTM. Un mécanisme d'attention standard (voir section 1.2.3) prend en entrée une projection de la représentation audio ainsi qu'une projection du mot précédent pour produire le contexte c_i qui représente l'information de l'audio pour le décodeur. Le schéma 3.2 montre le fonctionnement global du *Speller*. Pour la DTAA, nous avons renommé ce décodeur en *Teller*, qui semble plus approprié, et le modèle complet est nommé par la suite *Listen, Attend and Tell* (LAT).

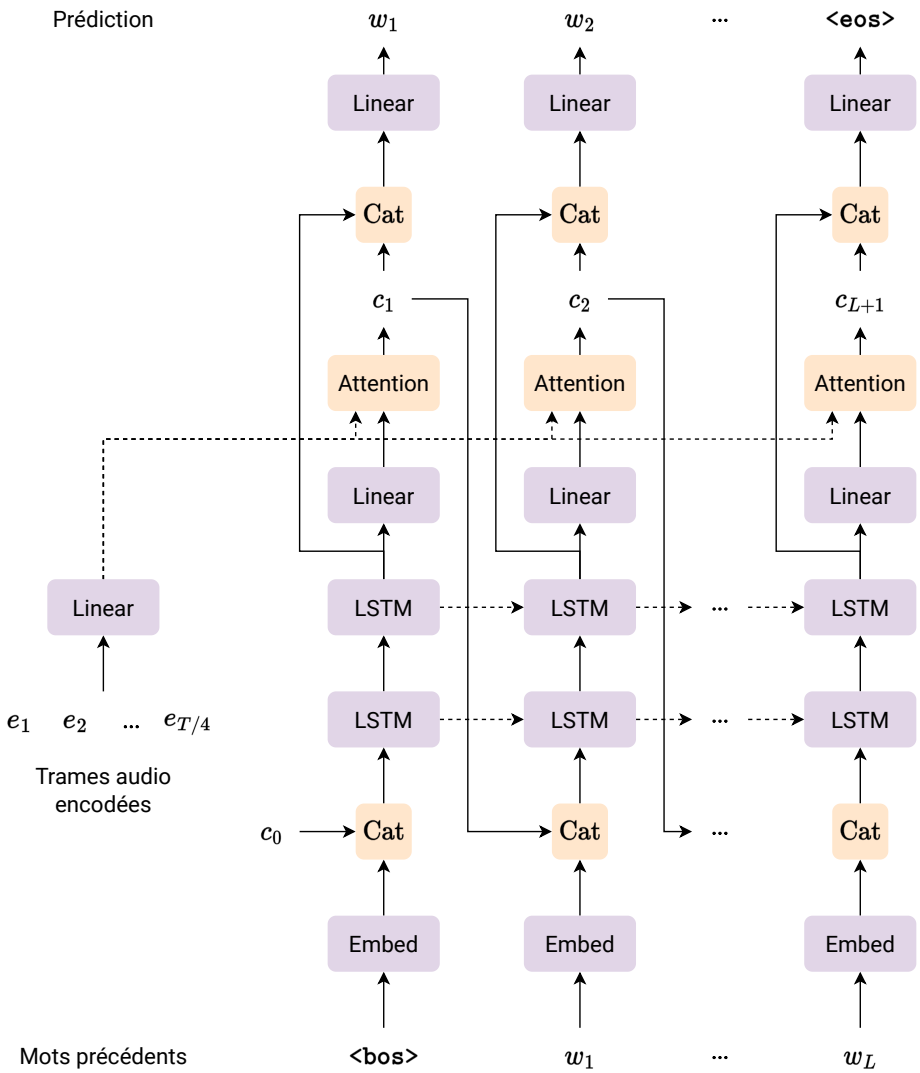
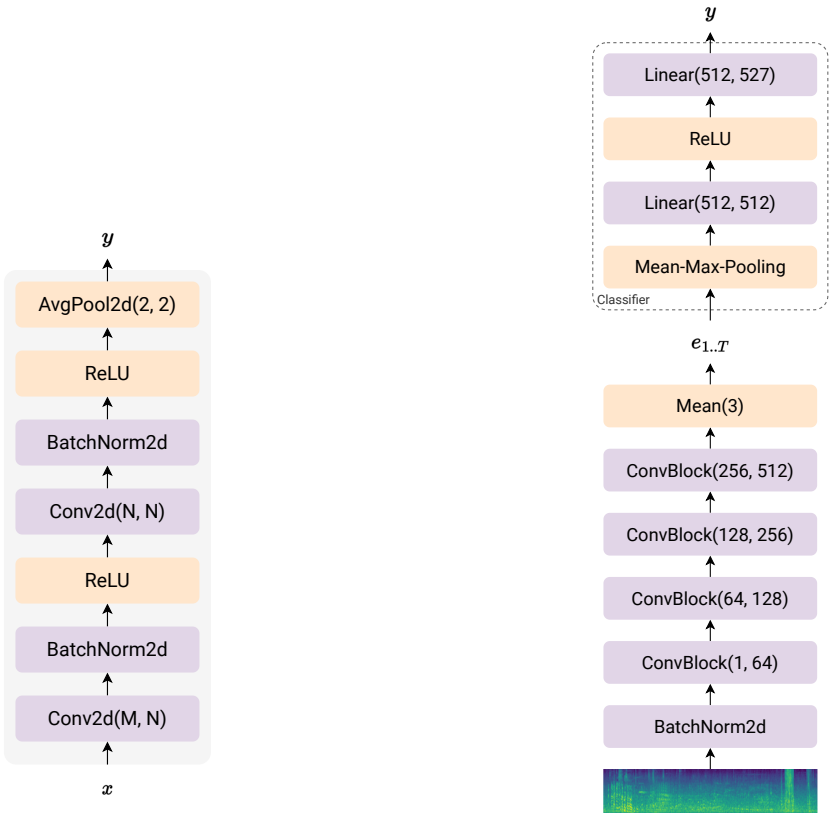


FIGURE 3.2 – Fonctionnement du *Speller* possédant deux couches LSTM pour une phrase w de taille L.

3.1.4 CNN10-teller

En 2019, l'étude sur la classification audio nommée PANN a proposé un ensemble de 20 modèles entraînés sur AudioSet (AS) pour classer des fichiers audio. Les modèles diffèrent en taille, en architecture, en vitesse et en performance, et sont accessibles librement. Comme la tâche de DTAA est étroitement liée à la tâche de classification d'événements sonores, beaucoup de systèmes ont utilisé des modèles pré-entraînés pour traiter l'audio, et notamment les modèles de PANN.

La plupart des modèles de PANN, en particulier CNN6, CNN10 et CNN14, reposent sur un module convolutif typique inspiré de ceux de VGGNet [Simonyan 2014]. Ce module est appelé `ConvBlock`, qui est détaillé par la figure 3.3a. Il contient deux couches de convolutions, avec des couches de normalisation de lot (*Batch normalization*, BN) pour limiter les variances des activations dans les modèles et enfin des fonctions d'activations non-linéaires de type *Rectified Linear Unit* (ReLU) [Nair 2010]. De plus, ces modules modifient le nombre de canaux de M à N et sont suivis d'une couche moyennant les valeurs (*AvgPool2d*) qui divisent la taille de l'entrée par deux. Nous verrons dans un futur chapitre que moderniser ce type de blocs à l'aide de convolutions profondes (*depthwise*) permet d'améliorer grandement les performances en classification audio et en DTAA.



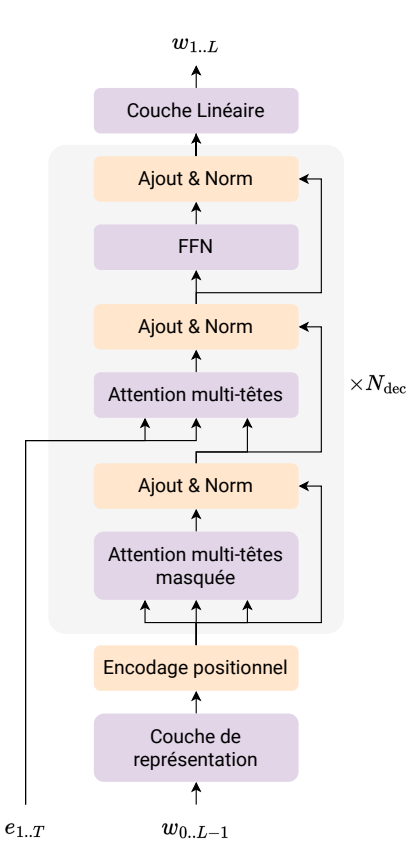
(a) `ConvBlock` des modèles de PANN. (b) CNN10.

FIGURE 3.3 – Modules et modèles convolutifs de PANN. Mean(i) désigne la moyenne selon le i-ème axe du tenseur d'entrée.

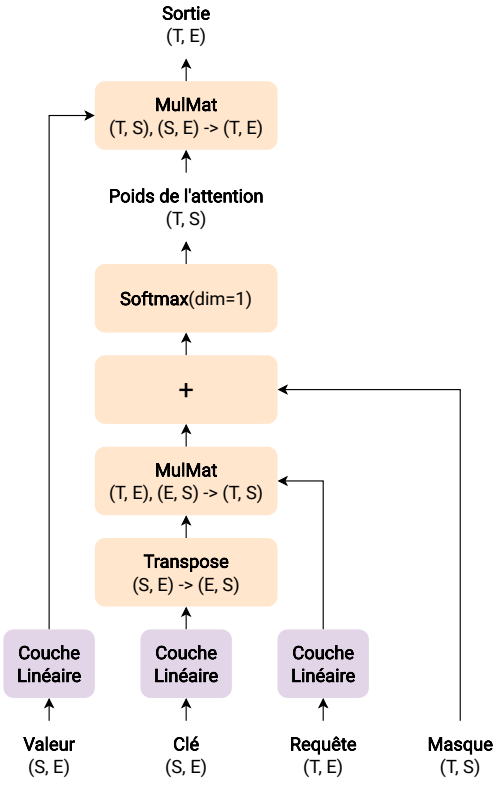
En 2021, nous avons décidé de réutiliser le modèle CNN10 décrit par la figure 3.3b. Il est relativement petit en nombre de paramètres (6 millions) et obtient une performance convenable sur AS avec 0,380 de précision moyenne (*mean Average Precision*, mAP). Le modèle est composé d'une séquence de 4 `ConvBlock`, d'une moyenne sur l'axe des fréquences puis d'une partie de classification qui est supprimée pour la DTAA. Nous obtenons donc en sortie une séquence de représentations de taille $T \times 512$, avec T l'axe temporel réduit (62 trames pour 10 secondes d'audio).

3.1.5 CNN10-trans

Le *Transformer* est une architecture de réseaux de neurones conçue pour traiter des séquences. Contrairement aux RNN, elle est uniquement fondée sur des mécanismes d'attention standard SDPA, sans aucune récurrence. Le *Transformer* est à l'origine un DNN de type encodeur-décodeur, chacun composé d'une séquence de blocs. Comme illustré dans la figure 3.4a ci-dessous, ces blocs intègrent deux mécanismes d'attention multitêtes (*Multi-Head Attention*, MHA) suivies par un *Feed-Forward Network* (FFN).



(a) Décodeur du *Transformer*.



(b) Schéma de l'attention multitêtes masquée (MHA). MulMat est l'abréviation de multiplication matricielle.

FIGURE 3.4 – Schéma du décodeur de type *Transformer* et du module MHA.

MHA est défini par l'équation 3.1, qui calcule en parallèle N_h attentions, permettant à chaque sortie de se focaliser sur plusieurs éléments de l'entrée. Dans le cas du premier MHA de chaque bloc du décodeur, celui-ci utilise pour ses trois entrées une représentation des mots précédents de

la phrase. Un masque est ajouté aux poids de l'attention produits pour éviter que le modèle ne puisse se reposer sur les mots futurs pour prédire le mot courant. Nous parlerons donc d'Attention Multi-têtes Masquée. Comme les trois entrées de cette attention sont les mêmes, cette couche peut aussi être appelée **Auto**-Attention Multi-tête Masquée. Aucun masque n'est appliqué pour le second MHA qui prend en entrée la séquence de représentations audio, car le modèle peut utiliser l'entièreté de la séquence audio pour prédire le mot suivant. Le schéma 3.4b montre comment le masque est appliqué pour contrôler certains poids de l'attention. Plus précisément, le masque contient des 0 ou $-\infty$ et il est ajouté aux valeurs de l'attention, avant la fonction d'activation softmax.

$$\mathrm{MHA}(Q, K, V) = W_o \ \mathrm{Cat}_{i=1}^{N_h} \ \mathrm{SDPA}(W_{Q,i}Q, W_{K,i}K, W_{V,i}V) \tag{3.1}$$

FFN est composé de trois couches : une linéaire qui augmente la dimension des représentations, puis une non-linéaire (comme ReLU) et enfin une dernière également linéaire qui remet la représentation à sa taille d'origine.

$$\mathrm{FFN}(x) = W_2 \max(W_1 x + b_1, 0) + b_2 \tag{3.2}$$

L'absence de récurrence temporelle dans les *Transformer* permet de les entraîner beaucoup plus rapidement que les RNN, en parallélisant totalement le traitement des séquences pour chaque élément i. Ces systèmes sont rapidement devenus très communs en traitement automatique des langues mais aussi en DTAA, supplémentant la partie encodeur ou décodeur des réseaux. Pour notre système de DTAA, nous utilisons uniquement la partie décodeur du réseau, décrite par le schéma 3.4a. Pour simplifier les expériences, nous utilisons l'implémentation officielle du *Transformer* de PyTorch [22], car une partie du traitement est optimisé en C++.

3.2 Mise en place expérimentale

Comme de nombreux systèmes de DTAA, nous utilisons des spectrogrammes log-Mel composés de 64 bandes de fréquences. La fenêtre utilisée est celle de Hamming [Hamming 1962], avec des fréquences allant de 50 à 14 000 Hz. Pour notre premier modèle LAT, nous utilisons une fenêtre de 64 ms et un recouvrement de moitié de 32 ms. Après l'introduction du modèle de PANN, nous utilisons la même taille de fenêtre de 32 ms et un recouvrement de 10 ms. Pour accélérer les entraînements, les spectrogrammes sont pré-calculés sur disque et stockés dans des fichiers de type *Hierarchical Data Format* pour accélérer leur lecture. L'emploi de spectrogrammes utilisant des fenêtres plus courtes ou d'un pas plus grand peut grandement accélérer l'entraînement et l'inférence le modèle, mais dégradera les performances de plusieurs points.

Les phrases sont mises en minuscules, la ponctuation et les caractères spéciaux sont supprimés à l'aide d'expressions régulières. Les phrases sont découpées en mots à l'aide du *tokenizer* anglais de la bibliothèque de traitement de texte spaCy [Montani 2021] (`en_core_web_sm`). Pour chaque jeu de données, nous calculons à l'avance l'ensemble des mots du vocabulaire sur le sous-ensemble d'entraînement. Le nombre de mots uniques obtenu est égal à 4 370 pour CL et 4 823 pour AC. De plus, le système possède quatre mots spéciaux ajoutés à chaque vocabulaire, qui sont supprimés lors du post-traitement de l'inférence pour retrouver les mots de la phrase :

22. https://pytorch.org/docs/stable/generated/torch.nn.Transformer.html

— *Begin-Of-Sentence token* (`<bos>`) est le mot indiquant le début d'une phrase, qui sert de première entrée au décodeur.
— *End-Of-Sentence token* (`<eos>`) est le mot indiquant la fin d'une phrase, qui sert notamment à arrêter la génération d'une phrase pendant l'inférence.
— *Pad token* (`<pad>`) est le mot servant à remplir les phrases pour en faire un lot. Il est ignoré par la fonction de coût CE et est égal au vecteur nul pour la couche de représentation d'entrée du décodeur.
— *Unknown token* (`<unk>`) est le mot servant à représenter des mots absents du jeu d'entraînement, comme certains mots de la validation d'AC. Il n'est pas utilisé pour les données d'entraînement, mais permet de calculer la fonction de coût sur la validation.

Lorsque plusieurs descriptions sont disponibles pour un fichier audio d'entraînement (comme pour CL), nous sélectionnons aléatoirement l'une d'entre elles. Dans nos expériences, une époque d'entraînement consiste donc à passer une fois sur chaque fichier audio, mais pas nécessairement sur chaque paire audio-texte. Durant ces premières expériences, nous avons utilisé l'optimiseur Adam, comme la majorité des systèmes de DTAA. Nous avons également appliqué une fonction diminuant le pas d'apprentissage durant l'entraînement, définie par l'équation 3.3 :

$$\mathrm{lr}_k = \frac{1}{2}\left(1 + \cos\left(\frac{k\pi}{K}\right)\right)\mathrm{lr}_0 \tag{3.3}$$

avec k l'indice de l'époque et K le nombre total d'époques d'entraînements.

Cette fonction décroissante permet de réduire le pas d'apprentissage et d'éviter au modèle d'osciller autour d'un optimum vers la fin de l'entraînement. Pour éviter l'effondrement du système dans certaines initialisations du réseau, nous avons également limité la norme ℓ_2 des gradients par une valeur fixe (*gradient clipping*) [Pascanu 2013]. Dans ces premières expériences, nous utilisons la fonction de coût sur l'ensemble de validation à la fin de chaque époque d'entraînement pour sélectionner le meilleur modèle. Comme plusieurs références sont disponibles pour chaque fichier audio sur les sous-ensembles de validation, nous calculons la moyenne les coûts de toutes les références associées à chaque audio.

Nous avons également utilisé l'augmentation SpecAugment (section 1.4.4.1) pour masquer une partie des spectrogrammes, avec deux bandes sur l'axe temporel avec une taille maximale de 64 trames et une bande sur l'axe de fréquence avec une taille maximale de deux fréquences. Cela permet de légèrement réduire l'effet de surapprentissage du modèle. Un lissage sur les étiquettes (*Label Smoothing*, LS) [Szegedy 2016] est également appliqué, et perturbe les vecteurs *one-hot* représentants les vérités terrain pour limiter la confiance maximale du modèle. Cette augmentation s'applique notamment dans un cadre de classification mono-étiquette et diminue la probabilité de la classe de la vérité terrain par une valeur p_{LS}. Les probabilités des autres classes seront mises uniformément à la valeur $p_{\mathrm{LS}} \Big/ \Big(|C| - 1\Big)$ pour garder une distribution dont la somme vaut 1. Cette augmentation permet surtout d'empêcher le modèle d'être trop confiant dans ses prédictions, et ainsi d'éviter de surévaluer certaines phrases ou mots.

Les couches de représentations du LAT utilisent toutes des vecteurs de taille 64 pour le *Listener* et le *Speller*. Le décodeur *Transformer* utilise une dimension de représentation de 256, quatre têtes d'attention, six blocs de décodeur standard empilées (N_{dec}) et un *Dropout* global P_{drop} fixé à 20%. La dernière couche affine projette les représentations à 256 dimensions vers une sortie

de la taille du vocabulaire de l'ensemble de données. La fonction d'activation *Gaussian Error Linear Unit* (GELU) [Hendrycks 2016] est utilisée dans le décodeur et la dimension intermédiaire de la couche FFN est égale à 2048. Les représentations positionnelles sont fixes et initialisées avec une fonction sinusoïde standard. Les valeurs de ces hyperparamètres ont été optimisées par une recherche par grille, ou sont celles utilisées par défault dans PyTorch. Nous avons également remarqué qu'un décodeur trop grand (avec une dimension supérieure à 512) ou trop profond (avec plus de 8 blocs) ne converge pas sur nos jeux de données.

Pour entraîner notre décodeur *Transformer*, nous utilisons la méthode d'apprentissage forcée (*teacher forcing*), qui consiste à donner au modèle les mots précédents de la vérité terrain pour prédire le mot suivant. Nous avons également expérimenté d'échantillonner les mots précédents dans ceux que le modèle prédit (*scheduled sampling*) [Bengio 2015], mais cela n'a pas apporté de gain pour le *Transformer*. Le LAT est entraîné avec 10% de ces propres prédictions pour le rendre plus robuste à ses propres erreurs, comme dans l'étude d'origine du modèle.

Nous utilisons pendant l'inférence la recherche en faisceau (*beam search*, cf. section 1.4.5.2) pour générer nos phrases. Nous avons également expérimenté trois méthodes de génération stochastiques : l'échantillonnage par top-k [Fan 2018], l'échantillonnage par noyau (section 1.4.5.3) et l'échantillonnage typique [Meister 2022]. En génération de texte, ces méthodes peuvent en théorie permettre de diversifier le vocabulaire limité des phrases. Cependant, nous avons trouvé que ces méthodes dégradent fortement les performances de notre système de DTAA [Labbé 2022].

Sauf mention contraire, nous présentons les scores sur les sous-ensembles de test, mais nos hyperparamètres sont bien choisis en fonction de la validation. Le tableau 3.1 résume les principaux hyperparamètres trouvés sur les deux jeux de données AC et CL. La taille du lot est assez faible dû au manque de mémoire GPU disponible à l'époque. La plupart des valeurs de l'optimiseur sont assez standard, à l'exception du pas d'apprentissage qui est plutôt faible. Ces valeurs sont assez proches de celles de la littérature, à l'exception de la taille du faisceau égale à 9 qui est plus grande que pour la plupart des systèmes sur CL. On peut également noter que la limite de la norme du gradient est plus faible sur CL car le jeu de données est plus diversifié, ce qui provoque de plus grandes valeurs de gradient. Le lissage des étiquettes est plus élevé sur CL (0,2), que sur AC (0,1) car le surapprentissage est plus important sur ce plus petit jeu de données.

Le travail de développement de notre système de DTAA a donné lieu à deux bibliothèques nommées `aac-datasets` [23] et `aac-metrics` [24], toutes deux conçues pour PyTorch. La première propose un ensemble de fonctions et deux classes pour télécharger, lire et pré-traiter les jeux de données de DTAA (AudioCaps, Clotho, MACS et WavCaps). La seconde fournit une liste de fonctions pour installer et calculer facilement toutes les métriques principales de DTAA. Des exemples d'utilisation du code sont donnés dans l'annexe F.

À noter que pour le jeu de données AC, nous avons téléchargé les données depuis YouTube en 2021, et une partie d'entre elles avaient déjà été supprimés par leurs utilisateurs. Notre version d'AC contient donc seulement 46 230 sur 49 838 fichiers d'entraînement, 464 sur 495 fichiers de validation et 912 sur 975 fichiers de test. Bien que ce soit discutable, nous utiliserons dans cette

23. https://github.com/Labbeti/aac-datasets
24. https://github.com/Labbeti/aac-metrics

TABLE 3.1 – Principaux hyperparamètres pour le premier système de référence SR1.

Nom	**Valeur**	
	AC	**CL**
Taille du lot (B)	8	
Optimiseur	Adam	
Pas d'apprentissage (lr_0)	$5 \cdot 10^{-4}$	
β_1	0,9	
β_2	0,999	
ϵ	10^{-8}	
λ_{wd}	10^{-6}	
Taille minimale des candidats (L_{min})	0	
Nb. d'époques (K)	50	
Limite de la norme du gradient	10	1
Lissage des étiquettes (p_{LS})	0,1	0,2
Taille maximale des candidats (L_{max})	52	20
Taille du faisceau (N_{beam})	2	9

thèse ces données pour nous comparer à la littérature, alors que certaines études testent leurs modèles avec plus ou moins de données sur ce sous-ensemble.

3.3 Résultats

Nous présentons ici les résultats qui ont le plus contribué à faire évoluer la performance du système. En l'occurrence, la principale modification a été l'architecture du modèle, avec le remplacement du *Listener* par le modèle pré-entraîné CNN10 de PANN, puis le remplacement du décodeur récurrent *Teller* par un décodeur de type *Transformer*. Nous obtenons donc trois modèles : le LAT, le CNN10-teller et le CNN10-*Transformer* (abrégé CNN10-trans).

Le tableau 3.2 donne le nombre de paramètres et d'opérations de chaque système, calculés par la bibliothèque DeepSpeed [25]. Le nombre d'opérations du LAT n'est pas inclus, car notre version de la bibliothèque ne supporte pas certaines couches BiLSTM de l'encodeur. Nous reportons également les scores de l'état de l'art en 2021 sur AC (PYB) et CL (CNN14-trans†) pour les comparer à nos systèmes. Nous pouvons remarquer que notre modèle est relativement petit par rapport à ceux de l'état de l'art, avec seulement 16,7 millions de paramètres comparés aux 88 et 494 millions. Le CNN10-trans est beaucoup plus large que le CNN10-teller en paramètres, mais leurs nombres d'opérations (*Multiply-Accumulate operations*, MACs) sont assez similaires.

Les tables 3.3 et 3.4 montre les scores des systèmes pour les neuf métriques classiques de DTAA. Nous calculons la performance de chaque architecture sur trois graines différentes pour s'assurer de la performance, et nous reportons l'intervalle de confiance à 95% selon une loi de

25. https://www.deepspeed.ai/tutorials/flops-profiler/

TABLE 3.2 – Caractéristiques des différents modèles. Le nombre MACs est calculé à l'aide d'un fichier audio de dix secondes issu d'AC.

Modèle	**Params. (M)**	**MACs (G)**
PYB [Gontier 2021]	494	N/A
CNN14-trans† [Won 2021]	88	N/A
LAT	1,8	N/A
CNN10-teller	6,2	13,0
CNN10-trans	16,7	13,3

student (*t-distribution*) de nos systèmes. Pour le système [Gontier 2021], les valeurs après le symbole ± correspondent à un écart-type pour trois différentes graines.

TABLE 3.3 – Résultats des premières métriques de DTAA (en %), par jeu de données et architecture. Les meilleurs scores de nos systèmes sont en **gras**.

Test	**Système**	**BLEU1**	**BLEU2**	**BLEU3**	**BLEU4**	**ROUGE-L**
AC	Humain	65,4±1,2	48,7±1,2	37,2±1,0	28,9±0,8	49,2±0,8
	PYB	69,9±0,5	52,3±0,7	38,0±0,8	26,6±0,9	49,3±0,4
	LAT	50,1±2,5	34,5±1,1	23,2±0,2	15,3±0,3	39,3±0,7
	CNN10-teller	59,8±0,7	43,9±1,3	31,6±2,9	22,5±3,6	**47,0**±0,5
	CNN10-trans	**65,7**±2,5	**48,4**±2,4	**34,5**±2,8	**23,6**±3,0	46,7±0,4
CL	Humain	65,5±0,4	49,7±0,6	39,5±0,7	32,1±0,7	50,9±0,2
	CNN14-trans†	56,4	37,6	25,4	16,3	38,8
	LAT	41,6±3,2	23,4±3,8	14,7±3,7	8,6±3,9	28,2±3,2
	CNN10-teller	47,8±1,2	28,3±1,6	17,8±1,5	10,3±1,5	31,8±1,1
	CNN10-trans	**55,5**±1,2	**36,3**±1,0	**24,5**±0,7	**16,2**±0,3	**37,8**±0,2

Notre modèle est beaucoup plus performant sur AC que sur CL, avec un score SPIDEr de 40,1% et 24,7%, respectivement. Il est également plus proche des scores humains sur AC. Ceci est probablement dû au fait que l'encodeur CNN10 a été pré-entraîné sur AS, qui est un sur-ensemble d'AC et donc d'un domaine plus proche. De plus, les descriptions d'AC sont plus simples que celles de CL, avec des phrases plus courtes et un vocabulaire relativement plus restreint (comme nous l'avions vu dans la section 2.5). Nous constatons donc que l'état de l'art d'AC est plutôt proche du meilleur score humain en SPIDEr (9,4% de différence), par rapport à l'état de l'art sur CL qui est plutôt éloigné de l'humain en SPIDEr (28,2% de différence).

Le modèle LAT est le plus petit des systèmes en nombre de paramètres, ce qui explique la performance reste assez faible comparé aux autres systèmes de l'état de l'art, avec 9,5% de SPIDEr comparé à 28,5%. L'introduction de l'encodeur pré-entraîné CNN10 triple la taille du modèle, mais améliore fortement les performances de toutes les métriques, avec un gain relatif moyen de 46,3% de SPIDEr. Enfin, l'ajout du décodeur *Transformer* plus grand et plus récent

TABLE 3.4 – Résultats des autres métriques de DTAA (en %), par jeu de données et architecture. Les meilleurs scores de nos systèmes sont en **gras**.

Test	Système	METEOR	CIDEr-D	SPICE	SPIDEr
AC	Humain	28,8±0,4	90,1±2,1	21,7±0,6	55,9±1,2
	PYB	24,1±0,3	75,3±0,9	17,6±0,3	46,5±0,6
	LAT	18,0±1,1	38,0±7,3	10,8±1,7	24,4±4,5
	CNN10-teller	**23,8**±0,8	57,0±4,1	15,1±1,3	36,0±2,7
	CNN10-trans	22,6±0,2	**63,6**±1,9	**16,5**±0,6	**40,1**±0,7
CL	Humain	30,5±0,2	90,3±1,2	23,1±0,2	56,7±0,5
	CNN14-trans†	17,7	44,1	12,8	28,5
	LAT	11,2±1,1	12,6±6,0	6,4±1,5	9,5±3,7
	CNN10-teller	12,9±0,9	19,0±1,0	8,2±1,3	13,6±1,1
	CNN10-trans	**17,1**±0,2	**37,6**±0,3	**11,9**±0,3	**24,7**±0,2

apporte un nouveau gain substantiel, avec une augmentation de 11,4% de SPIDEr. Nous pouvons cependant noter que les métriques ROUGE-L et METEOR ne sont pas améliorées par ce nouveau décodeur, mais la différence reste peu significative au vu des intervalles de confiances. Par ailleurs, ces intervalles ont été réduits pour le modèle CNN10-trans, notamment en CIDEr-D sur AC, indiquant que les entraînements sont plus stables avec le décodeur *Transformer* qu'avec le *Speller*.

3.4 Exemples de sorties

Comme vu dans le chapitre 2, les principales métriques utilisées en DTAA ne reflètent pas tous les aspects des textes produits par ces systèmes. Pour mieux étudier ces sorties, nous avons reporté dans le tableau 3.5 quelques exemples de candidats générés pour quatre fichiers audio [26] [27] [28] [29] par nos trois modèles. Les références correspondantes sont données dans l'annexe D.

Les descriptions automatiques du premier fichier audio C1, C2 et C3 montrent que l'événement sonore reconnu peut être différent, avec un bruit de véhicule reconnu par le LAT et une suite d'explosions par le CNN10-trans. Certaines évolutions sont plus mineures, comme les descriptions C7 et C9 où les mots utilisés et leur ordre ont légèrement changé. Nous pouvons également noter que les modèles ont tendance à répéter certains n-grammes, parfois au niveau d'un candidat comme *« explosion »* pour C3, mais aussi au niveau de l'ensemble des candidats avec *« a woman speaks »* comme C6, C9 et C12. Les descriptions contiennent bien souvent un ou deux événements sonores, et restent assez proches des événements sonores catégorisés dans le jeu AS. Le modèle semble être capable de reconnaître plusieurs événements dans le même fichier, mais les descriptions restent assez génériques et peu diversifiées.

26. https://www.youtube.com/embed/6BJ455B1aAs?start=0&end=10
27. https://www.youtube.com/embed/VjSEIRnLAh8?start=30&end=40
28. https://www.youtube.com/embed/_xylo5_IiaM?start=470&end=480
29. https://www.youtube.com/embed/PLHXGDnig4M?start=3&end=13

TABLE 3.5 – Exemple de candidats de nos trois systèmes pour quatre fichiers audio sur AC-test.

Audio	N°	Modèle	Descriptions
6BJ455B1aAs	C1	LAT	*a vehicle engine is revving and then stops*
	C2	CNN10-teller	*wind blows strongly and a gun is shot several times*
	C3	CNN10-trans	*a loud explosion followed by explosions*
VjSEIRnLAh8	C4	LAT	*a woman speaks while a sewing machine runs*
	C5	CNN10-teller	*a woman talks while some liquid flows*
	C6	CNN10-trans	*a woman speaks while food fries*
_xylo5_IiaM	C7	LAT	*a baby cries and a woman speaks*
	C8	CNN10-teller	*a baby cries and a woman speaks*
	C9	CNN10-trans	*a child is speaking and a woman speaks*
PLHXGDnig4M	C10	LAT	*a man speaks and a cat meows*
	C11	CNN10-teller	*a woman speaks and a cat meows*
	C12	CNN10-trans	*a cat meows and a woman speaks*

3.5 Bilan

Ce chapitre avait pour but de présenter nos premiers systèmes de DTAA, inspirés de la littérature naissante sur le sujet en 2020-2021. Nous avons d'abord adopté une approche inspirée des modèles de RAP avec l'architecture LAT, utilisé pour la première fois pour DCASE2020. Puis, nous avons introduit un encodeur pré-entraîné pour aider le modèle à mieux reconnaître les événements sonores, pour finir par remplacer le décodeur récurrent par une architecture plus moderne issue des *Transformer*. Dans le chapitre suivant, nous nommons le modèle CNN10-trans le premier système de référence SR1, qui obtient 40,1% sur AC et 24,7% sur CL en SPIDEr.

Comme prévu, le modèle pré-entraîné apporte une amélioration draconienne au système et l'introduction d'un décodeur *Transformer* surpasse le décodeur récurrent. Notre système CNN10-trans s'est approché de l'état de l'art de 2021 pour certaines métriques comme ROUGE-L, METEOR ou SPICE avec approximativement 1% de différence sur le jeu de données CL, mais il reste moins performant pour les métriques CIDEr-D et SPIDEr avec respectivement 6,5% et 3,8% de différence. Sur AC, notre système est un peu plus éloigné de l'état de l'art avec 6,4% de différence en SPIDEr, mais il utilise un ordre de grandeur de moins de paramètres (16,7 au lieu de 494 millions). Sur les deux jeux de données, nous remarquons que la différence se joue surtout sur le CIDEr-D, qui a un impact notable sur la métrique SPIDEr.

Maintenant que nous avons ce premier système de référence, nous verrons notamment dans le chapitre 5 comment nous avons perfectionné notre système pour les scores de performance sur les métriques principales, mais aussi en nombre de paramètres pour accélérer l'apprentissage et l'inférence.

Deuxième partie

Contributions

Chapitre 4

SPIDEr-max

Comme nous l'avons vu dans le chapitre sur l'état de l'art des métriques, le processus d'évaluation des systèmes de DTAA exploite des métriques de traduction automatique et de DTAI, qui consistent à comparer une phrase candidate à une ou plusieurs phrases de références. Comme l'évaluation de textes générés automatiquement est un problème difficile, plusieurs mesures sont utilisées en parallèle. Nous étudions en particulier la métrique SPIDEr (section 2.2.3), un nom court désignant la moyenne de deux métriques appelées CIDEr-D et SPICE, respectivement décrites dans les sections 2.2.1 et 2.2.2. SPIDEr est utilisé, par exemple, dans les défis annuels du challenge DCASE pour classer les systèmes de DTAA.

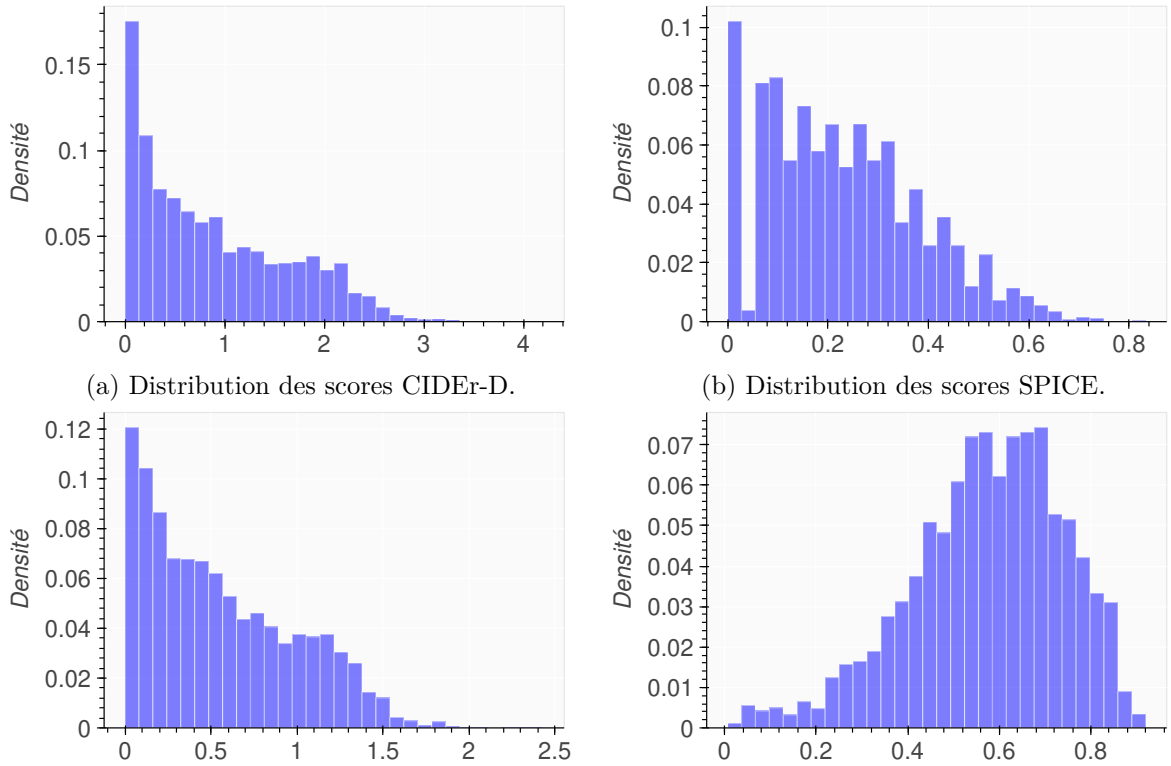
Malgré son emploi important dans la littérature, plusieurs caractéristiques de la métrique semblent assez curieuses. Par exemple, les valeurs de SPIDEr peuvent varier entre 0 et 550%. De plus, les scores humains (section 2.5) sont plutôt proches de 56%, bien loin de la valeur théorique maximale. Nous avons donc décidé d'étudier cette métrique, en utilisant l'algorithme de recherche en faisceau pour générer de multiples phrases par audio, pour étudier les variances de scores lorsque la métrique est soumises à plusieurs candidats similaires. Nous nommons ce calcul SPIDEr-max, et nous comparons ces valeurs avec ceux de SPIDEr.

Sommaire

4.1 Limites de SPIDEr

4.1.1 SPIDEr est majoritairement influencé par CIDEr-D

En premier lieu, nous avons reporté dans les figures 4.1 les distributions des scores de différentes métriques sur CL-eval avec notre modèle SR1 décrit dans le chapitre 3. Nous observons que les valeurs de SPICE (voir figure 4.1b) sont en général bien plus faibles que celles de CIDEr-D (voir figure 4.1a), surtout lorsque l'on regarde l'intervalle de valeur de l'axe des abscisses. CIDEr-D semble être la principale métrique qui ait un impact important sur SPIDEr, ce qui signifie que la fréquence des n-grammes trouvés est la principale source de gains constatés par les systèmes de la littérature. Ces figures montrent également qu'une bonne partie des scores s'approchent de zéro, car de nombreuses phrases ne partagent pas de n-grammes. Nous pouvons constater que les scores de la FENSE sont mieux distribués grâce à l'utilisation des représentations sémantiques (voir figure 4.1d). Cette dernière observation avait également été faite dans une étude sur les métriques de DTAA [Martín-Morató 2022].



(a) Distribution des scores CIDEr-D.

(b) Distribution des scores SPICE.

(c) Distribution des scores SPIDEr.

(d) Distribution des scores FENSE.

FIGURE 4.1 – Distribution des scores de CIDEr-D, SPICE, SPIDEr et FENSE pour les références utilisées pour calculer le score humain sur CL.

4.1.2 SPIDEr varie entre des candidats similaires

Comme la plupart des systèmes de DTAA, nous utilisons un décodeur avec une recherche en faisceau qui permet de générer plusieurs descriptions par fichier audio durant l'inférence. Par défaut, la description la plus probable selon le modèle est sélectionnée pour calculer le score

SPIDEr de notre système. La probabilité P donnée par le modèle est définie par l'équation 4.1 pour une phrase w de longueur L :

$$P(w_{1...L}) = \left(\prod_{i=1}^{L} \mathrm{P}\left(w_i | w_{1..i-1}\right) \right)^{1/L} \tag{4.1}$$

En premier lieu, nous avons décidé d'observer autres phrases produites par la recherche en faisceau, pour repérer les variantes de SPIDEr. Les tableaux 4.1 et 4.2 montrent des exemples de candidats et de références, provenant respectivement de CL-eval [30] et d'AC-test [31]. Nous avons également indiqué les scores SPIDEr et les probabilités P en % données par le modèle de chaque candidat.

TABLE 4.1 – Candidats et références du fichier « `rain.wav` », issu de CL-eval. La valeur la plus élevée par colonne est en **gras**.

N°	**Candidats**	**P (%)**	**SD (%)**
C1	*Heavy rain is falling on a roof*	36,1	56,2
C2	*Heavy rain is falling on* ***a tin roof***	**40,8**	**93,0**
C3	*A heavy rain is falling on a roof*	36,9	59,4
C4	*A heavy rain is falling on the ground*	35,1	33,5
C5	*A heavy rain is falling on the roof*	34,0	59,4
N°	**Références**		
R1	*Heavy rain falls loudly onto a structure with a thin roof*		
R2	*Heavy rainfall falling onto a thin structure with a thin roof*		
R3	*It is raining hard and the rain hits* ***a tin roof***		
R4	*Rain that is pouring down very hard outside*		
R5	*The hard rain is noisy as it hits* ***a tin roof***		

Dans le tableau 4.1, le candidat le plus probable est aussi celui qui a le score SPIDEr le plus élevé. Cependant, la différence relative de ce score par rapport à celui des autres candidats s'explique par le mot *« tin »*, qui est plutôt rare et augmente le score SPIDEr. Ainsi, les différences observées entre les candidats semblent justifiées, car ce mot ajoute un détail assez précis. Au contraire, dans le second exemple issu d'AC dans le tableau 4.2, le candidat le plus probable est différent de celui qui a obtenu le meilleur score SPIDEr. La phrase sélectionnée a un score SPIDEr bien plus faible 18,9%) que la meilleure selon la métrique SPIDEr (125,9%). Cette large différence s'explique par le 5-gramme *« speaks and a goat bleats »*, qui contient beaucoup de sous séquences de mots en commun avec ceux des références, avec des mots peu fréquents comme *« goat »*.

Lorsque nous avons examiné les candidats générés à l'échelle d'un sous-ensemble de test, nous constatons que la plupart des candidats à la recherche en faisceau sont très similaires. Sur CL par exemple, les scores de cross-référence SPIDEr entre les candidats générés par la

30. https://freesound.org/people/chefroe/sounds/327779
31. https://www.youtube.com/embed/jid4t-FzUn0?start=20&end=30

TABLE 4.2 – Candidats et références du fichier « `jid4t-FzUn0` », issu d'AC-test. La valeur la plus élevée par colonne est en **gras**.

N°	Candidats	P (%)	SD (%)
C1	*A woman speaks and a sheep bleats*	47,5	19,0
C2	*A woman **speaks and a goat bleats***	46,4	**125,9**
C3	*A man speaks and a sheep bleats*	46,4	34,4
C4	*An adult male speaks and a sheep bleats*	45,0	23,1
C5	*An adult male is speaking and a sheep bleats*	**49,1**	18,9
N°	**Références**		
R1	*A man speaking and laughing followed by a goat bleat*		
R2	*A man is speaking in high tone while a goat is bleating one time*		
R3	*A man speaks followed by a goat bleat*		
R4	*A person **speaks and a goat bleats***		
R5	*A man is talking and snickering followed by a goat bleating*		

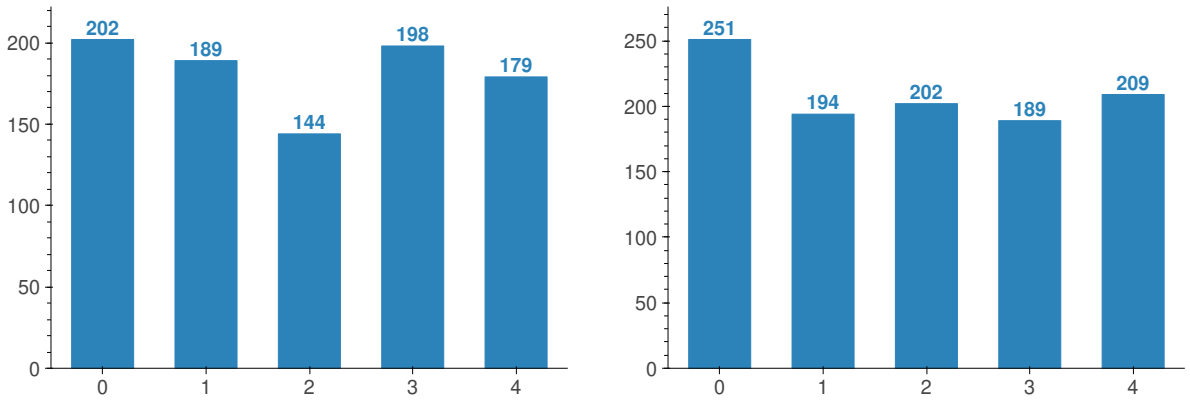
recherche en faisceau atteignent 252,2%. Ils décrivent généralement les mêmes événements sonores avec seulement quelques mots différents. Cependant, le score SPIDEr avec les références peut varier considérablement si un mot rare ou un n-gramme prédit par le modèle se trouve dans les références.

Au vu de ces observations, il est donc légitime de se demander si la sélection par la probabilité la plus élevée permet réellement de meilleurs SPIDEr au niveau d'un corpus. Pour une taille de faisceau égale à 5, l'exactitude (*accuracy*) entre l'index du meilleur candidat selon la probabilité P et selon le score SPIDEr est seulement de 26,5% sur CL, et de 22,6% sur AC. Le coefficient de corrélation entre toutes les probabilités et les scores SPIDEr est de 0,224 pour CL et de 0,259 pour AC. Cela montre que la probabilité maximale P ne correspond pas toujours au meilleur candidat possible, et que le score SPIDEr pourrait être beaucoup plus élevé en sélectionnant mieux le meilleur candidat de la recherche en faisceau.

4.1.3 Pouvons-nous choisir un meilleur candidat automatiquement ?

Sélectionner automatiquement la meilleure phrase parmi les candidats de recherche en faisceau est un problème assez difficile. En effet, la plupart des candidats sont très similaires et décrivent généralement les mêmes événements sonores avec des mots légèrement différents. En premier lieu, nous pourrions nous demander si certains faisceaux donnent de meilleurs scores, c'est pourquoi nous avons reporté l'histogramme des indices des meilleurs candidats d'après le SPIDEr dans les figures 4.2a et 4.2b. Ces figures révèlent qu'*a priori* aucun indice de faisceau ne semble meilleur qu'un autre, que ce soit sur AC ou CL.

Nous avons alors essayé de sélectionner automatiquement le meilleur candidat en utilisant plusieurs méthodes : la taille du vocabulaire généré le plus grand, la longueur de la phrase, et même avec RNN entraîné à choisir la meilleure phrase. Malheureusement, toutes ces approches n'ont pas réussi à mieux classifier les candidats que la sélection par probabilité. Par exemple, les taux de classifications binaires du RNN pour deux candidats aléatoires d'un audio atteignent à



(a) Sur AC-test.

(b) Sur CL-eval.

FIGURE 4.2 – Histogramme des indices des cinq meilleurs candidats.

peine 55%, alors qu'un choix aléatoire vaut 50%. Il est donc assez difficile d'améliorer de manière significative le score SPIDEr global. Nous pouvons supposer qu'il faudrait un système de RAT assez puissant pour repérer la meilleure phrase, ce qui nécessiterait beaucoup plus de calcul. Dans la littérature, une méthode complexe pour mieux sélectionner les candidats a été proposée [Wu 2023b], et décrite dans la section 1.4.5.3. Elle permet de gagner entre 1% et 2% de score SPIDEr absolu, mais elle reste assez coûteuse car elle demande à générer 50 candidats par fichier audio et utilise un modèle de texte pré-entraîné très large (1,5 milliard de paramètres).

Malgré l'échec de trouver une stratégie pour mieux sélectionner un candidat, nous proposons d'étudier l'impact qu'une sélection parfaite aurait sur le score SPIDEr. Plus précisément, nous proposons une métrique, nommée SPIDEr-max, qui considère tous les candidats produits par le modèle pour chaque audio, puis nous sélectionnons le meilleur score SPIDEr à chaque fois.

4.2 SPIDEr-max

4.2.1 Définition

Pour une liste de candidats C et une liste de références R d'un audio, SPIDEr-max est défini par l'équation suivante :

$$\text{SPIDEr-max}(C, R) = \max_{c \in C} \text{SPIDEr}(c, R) \tag{4.2}$$

Cette métrique consiste à ne garder que le plus grand score SPIDEr parmi les scores calculés pour chaque candidat, afin d'éviter de devoir choisir qu'un seul candidat. Une illustration de la différence entre SPIDEr et SPIDEr-max est donnée dans les figures 4.3 et 4.4. Comme pour SPIDEr, les valeurs de SPIDEr-max sont comprises entre 0 et 5,5. Le code source PyTorch pour calculer la métrique est disponible sur GitHub dans notre bibliothèque `aac-metrics` [32].

Dans la littérature, peu de métriques semblent prendre en compte plusieurs phrases candidats par fichier audio. En effet, la sélection de la meilleure phrase est considérée comme faisant partie de la tâche de DTAA. Cependant, l'utilisation de ce type de métrique permet de mieux

32. https://github.com/Labbeti/aac-metrics

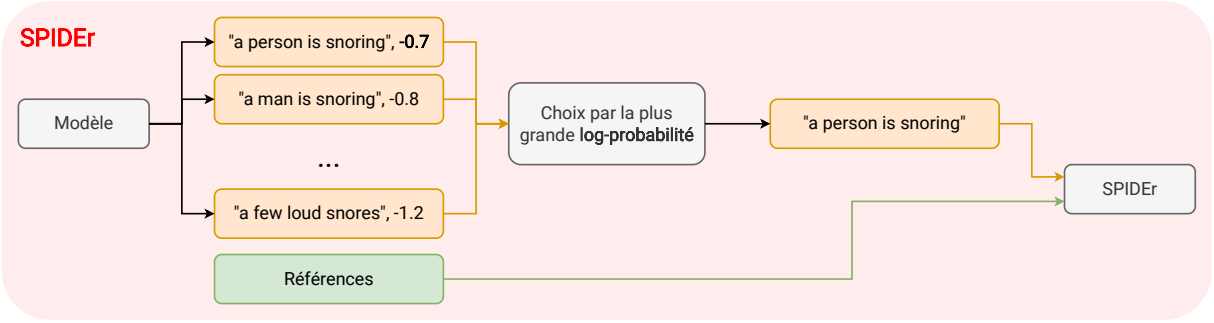


FIGURE 4.3 – Illustration du SPIDEr avec les candidats de la recherche en faisceau.

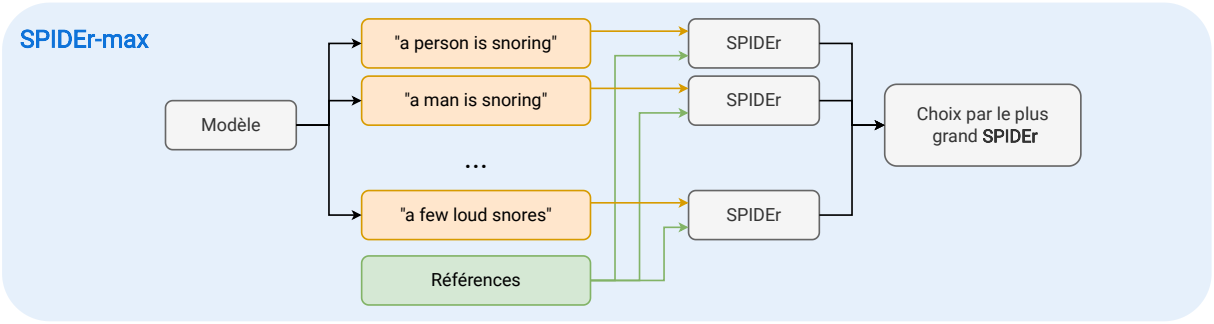


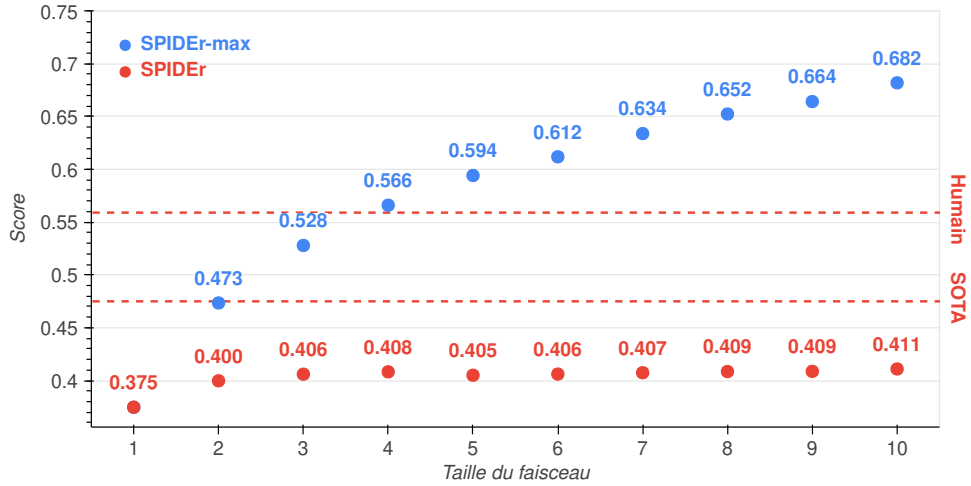
FIGURE 4.4 – Illustration du SPIDEr-max avec les candidats de la recherche en faisceau.

comprendre le comportement des modèles, comme avec la métrique M-SPICE [Hsu 2021]. Cette dernière est une variante de SPICE, et a été introduite pour évaluer la diversité de lorsque plusieurs candidats sont générés par fichier audio. La seule différence avec SPICE réside dans le graphe sémantique de chaque candidat qui fusionné en un seul graphe, exactement comme pour les graphes associés aux multiples références de chaque audio. Les autres étapes de SPICE restent les mêmes. D'une manière similaire, notre métrique nous permet d'étudier le comportement de notre modèle et notamment pour l'impact de la sélection des candidats, ainsi que la variabilité du score SPIDEr.

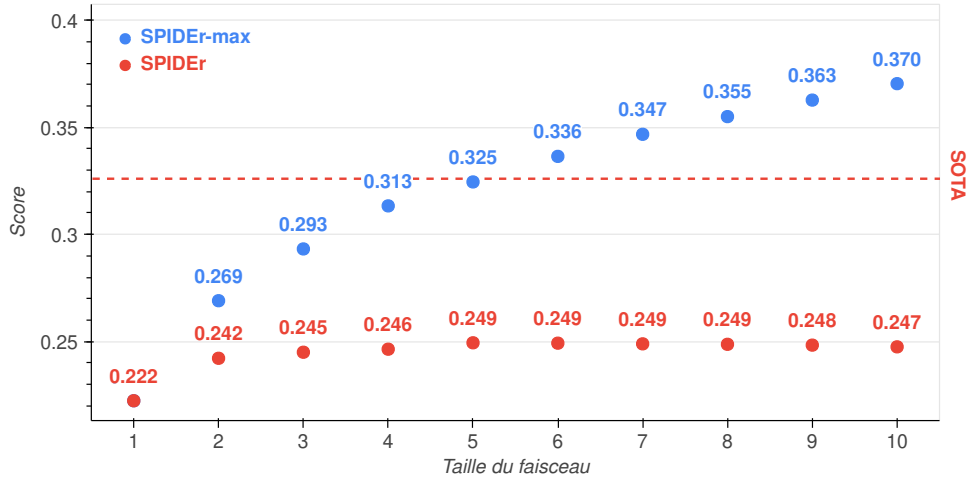
4.2.2 Résultats

Le score de SPIDEr-max dépend fortement du nombre de candidats que nous utilisons, qui ici est égal à taille du faisceau. C'est pourquoi nous présentons les résultats du score SPIDEr et SPIDEr-max avec des tailles de faisceaux de 1 à 10 dans les figures 4.5b et 4.5b pour les jeux de données de test d'AC et de CL, respectivement.

Sur AC, le score SPIDEr-max augmente rapidement au-dessus du score de notre modèle de 40,1% à 47,3% avec seulement une taille de faisceau de deux, ce qui représente une augmentation relative de 18%. Les scores de SPIDEr-max continuent d'augmenter au-dessus du score de l'état de l'art de 2022 (47,5%) et même au-delà du score humain (55,9%). Cette dernière observation montre que, bien que notre modèle produise des descriptions aussi bonnes que l'humain selon le SPIDEr, il ne parvient pas à les sélectionner lorsque nous utilisons la probabilité maximale.



(a) Sur AC-test.



(b) Sur CL-eval.

FIGURE 4.5 – SPIDEr and SPIDEr-max scores en fonction de différentes tailles de faisceaux.

Sur CL, les scores augmentent également avec une taille de faisceau plus élevée, dépassant l'état de l'art (32,6%) mais pas le score humain (56,7%). Cela semble être dû à la performance des modèles sur ce jeu de données, qui est bien plus éloignée du score humain, lui-même causé par la grande difficulté du jeu de données CL, qui a une plus grande diversité de n-grammes.

Nous avons également essayé de calculer le score SPIDEr-max pour une taille de faisceau égale à 100, ce qui a donné un score de 95,3% pour AC et de 53,5% pour CL. Le système peut donc largement dépasser le score humain sur AC, et peut se rapprocher du score humain sur CL. Pour la suite, nous avons décidé de nous concentrer sur une taille de faisceau plus faible, car cela serait plus réaliste dans un scénario réel, où, par exemple, plusieurs descriptions automatiques seraient proposées à un utilisateur.

4.2.3 Pourquoi une telle augmentation de SPIDEr-max ?

Comme nous l'avons vu dans la section précédente, SPIDEr-max augmente rapidement et dépasse même le score SPIDEr humain sur AC. Nous avons également remarqué que la prédiction d'un n-gramme peu fréquent correct semble améliorer considérablement le score d'un candidat, probablement en raison de la métrique CIDEr-D basée sur le TF-IDF des n-grammes. Pour confirmer qu'il existe une relation entre le TF-IDF et les scores SPIDEr et SPIDEr-max, nous avons calculé la corrélation entre les différences de SPIDEr et les différences de TF-IDF. Plus précisément, pour chaque fichier audio, nous avons calculé la différence entre le SPIDEr et le SPIDEr-max et la moyenne des TF-IDF de la meilleure phrase selon le SPIDEr et la meilleure phrase selon la probabilité du modèle. Les corrélations entre ces différences sont presque égales à 1 pour AC et CL (respectivement 0,998 et 0,992), ce qui confirme que c'est bien la fréquence des mots sélectionnés qui augmente radicalement le CIDEr-D et, par conséquent, également le score du SPIDEr-max. Si ces résultats semblent montrer que notre système de DTAA génère des phrases d'un niveau humain, cela montre également que la métrique SPIDEr est en réalité très sensible à la fréquence des n-grammes prédits, et peut facilement donner des scores très variables pour certaines phrases similaires.

4.2.4 Variantes du SPIDEr-max

Pour approfondir l'étude des métriques, nous avons calculé les scores de SPIDEr avec plusieurs autres fonctions que le maximum pour comprendre comment l'augmentation de la taille du faisceau (N_{beam}) impacte l'ensemble des candidats générés. Pour les fonctions minimum, moyenne et aléatoire, nous nommons respectivement ces métriques SPIDEr-min, SPIDEr-moy et SPIDEr-rand. Les scores sont reportés dans les tables 4.3 et 4.4.

TABLE 4.3 – Résultats des variantes du SPIDEr-max sur AC-test (en %). La valeur la plus élevée par ligne est en **gras**.

N_{beam}	1	2	3	4	5	6	7	8	9	10
SPIDEr	37,5	40,0	40,6	40,8	40,5	40,6	40,7	40,9	40,9	**41,1**
SPIDEr-max	37,5	47,3	52,8	56,6	59,4	61,2	63,4	65,2	66,4	**68,2**
SPIDEr-min	**37,5**	30,5	25,8	22,6	20,4	18,6	17,4	16,3	15,6	14,9
SPIDEr-moy	37,5	**38,9**	38,7	38,2	37,7	37,3	37,1	36,8	36,6	36,4
SPIDEr-rand	37,5	**38,5**	38,4	38,1	37,3	37,9	38,0	36,5	35,8	35,3

TABLE 4.4 – Résultats des variantes du SPIDEr-max sur CL-eval (en %). La valeur la plus élevée par ligne est en **gras**.

N_{beam}	1	2	3	4	5	6	7	8	9	10
SPIDEr	22,2	24,2	24,5	24,6	**24,9**	24,9	24,9	24,9	24,8	24,7
SPIDEr-max	22,2	26,9	29,3	31,3	32,5	33,6	34,7	35,5	36,3	**37,0**
SPIDEr-min	**22,2**	20,2	18,2	16,8	15,9	15,1	14,3	13,6	13,0	12,5
SPIDEr-moy	22,2	23,6	23,5	**23,7**	23,6	23,5	23,5	23,4	23,4	23,4
SPIDEr-rand	22,2	23,4	23,3	**23,9**	23,3	23,4	23,7	23,1	22,9	23,3

Si SPIDEr-max montrait un gain significatif du score en sélectionnant la meilleure phrase, SPIDEr-min montre une décroissance des scores très rapide, atteignant 14,9% sur AC-test et 12,5% sur CL-eval. Nous pouvons également noter que SPIDEr-moy et SPIDEr-rand diminuent légèrement avec l'augmentation de la taille du faisceau. Cela semble indiquer que la pertinence des candidats décroit au fur et à mesure.

4.2.5 FENSE-max

En DTAA, même si le SPIDEr est la principale métrique utilisée pour classer les systèmes, il existe de nombreuses autres métriques pour les évaluer. Le FENSE (décrit en section 2.3.2) est très différent de la métrique SPIDEr et il n'est pas fondé sur les n-grammes ou sur la fréquence de ces derniers. Il est donc légitime d'explorer cette métrique de la même manière que le SPIDEr pour savoir si son comportement sera similaire. Nous reportons dans les tableaux 4.5 et 4.6 les scores des variantes de FENSE sur AC et CL, respectivement.

TABLE 4.5 – Résultats des variantes du FENSE-max sur AC-test (en %). La valeur la plus élevée par ligne est en **gras**.

N_{beam}	1	2	3	4	5	6	7	8	9	10
FENSE	59,0	**59,2**	59,1	59,1	58,9	59,1	59,0	59,0	59,0	59,1
FENSE-max	59,0	61,9	63,5	64,4	65,1	65,9	66,6	67,0	67,3	**67,6**
FENSE-min	**59,0**	56,2	53,8	52,0	50,6	49,6	48,7	47,9	47,2	46,6
FENSE-moy	59,0	**59,1**	58,7	58,6	58,4	58,5	58,5	58,4	58,3	58,3
FENSE-rand	**59,0**	58,7	58,7	58,7	58,1	58,8	58,4	58,7	58,5	58,1

TABLE 4.6 – Résultats des variantes du FENSE-max sur CL-eval (en %). La valeur la plus élevée par ligne est en **gras**.

N_{beam}	1	2	3	4	5	6	7	8	9	10
FENSE	45,3	46,1	46,4	46,5	46,6	**46,6**	46,5	46,4	46,2	46,0
FENSE-max	45,3	48,1	49,7	50,9	51,6	52,1	52,7	53,2	53,6	**53,9**
FENSE-min	**45,3**	44,0	42,6	41,8	40,8	40,1	39,3	38,6	37,9	37,5
FENSE-moy	45,3	46,1	46,2	**46,3**	46,3	46,3	46,3	46,2	46,2	46,1
FENSE-rand	45,3	46,1	46,0	**46,6**	46,1	46,2	46,6	46,1	45,6	45,8

Malgré une forte différence entre les deux métriques, les comportements de la métrique FENSE sont globalement assez similaires à ceux de la métrique SPIDEr. Les scores de FENSE-max augmentent et ceux de FENSE-min diminuent assez rapidement. Mais contrairement à SPIDEr-max, FENSE-max ne dépasse pas les scores humains (68,0% pour AC et 57,4% pour CL) pour une taille de faisceau de 10. Cependant, l'augmentation de la valeur de la métrique indique que le meilleur candidat selon le FENSE est tout de même bien supérieur à celui sélectionné par la probabilité maximale. La métrique semble donc moins sensible aux mots employés, mais nous pourrions toujours obtenir des scores si on choisissait plus intelligemment la meilleure phrase de

la recherche en faisceau. Les scores de FENSE-moy. et FENSE-rand. restent cependant assez constants selon la taille du faisceau, indiquant que la métrique semble plus stable que SPIDEr.

4.3 Bilan

Dans ce chapitre, nous avons montré que le score SPIDEr est très sensible aux n-grammes employés dans les candidats générés par notre système de DTAA. Pour cela, nous avons utilisé de multiples candidats par audio, générés par la recherche en faisceau. Les candidats sont très similaires, mais peuvent donner des scores SPIDEr très différents. La sélection du meilleur candidat par la probabilité du modèle est sous-optimale, mais les méthodes automatiques visant à mieux choisir le candidat peinent à obtenir de meilleurs résultats.

Nous avons donc proposé une nouvelle mesure, SPIDEr-max, qui prend en compte plusieurs candidats pour chaque enregistrement audio en sélectionnant le meilleur SPIDEr. Les scores de SPIDEr-max, comparés aux scores humains de SPIDEr, montrent que notre modèle produit déjà des candidats semblables à ceux des humains d'après la métrique, mais échoue à les sélectionner. Cela montre également que la métrique SPIDEr est très sensible aux mots des phrases générées, et peut facilement les surévaluer. L'étude du comportement de la métrique SPIDEr à travers SPIDEr-max a donné lieu à un article lors de la conférence du *DCASE2022 Workshop* [Labbé 2022].

Dans la suite de ce manuscrit, nous avons décidé de montrer les résultats des systèmes sur la métrique SPIDEr, car elle reste la principale métrique utilisée en DTAA. Nous incluons aussi la métrique FENSE, car elle semble moins variable que SPIDEr, et prend en compte la sémantique des phrases et les erreurs de fluence. Enfin, nous reporterons également le nombre de mots unique Vocab qui donne un aperçu de la diversité des phrases. Dans le chapitre suivant, nous verrons comment nous pouvons améliorer notre modèle SR1 selon ces différentes métriques, pour se rapprocher au mieux de l'état de l'art.

Chapitre 5

Vers un meilleur système de DTAA

Dans le domaine de la DTAA, il existe de nombreuses manières d'entraîner, de spécialiser et de tester ces modèles, qui peuvent avoir un impact plus ou moins important sur les performances du système. Dans ce chapitre, nous verrons comment nous avons amélioré notre système SR1, avec pour objectif de créer un système à la fois performant sur les jeux de données publics et efficace en coût d'entraînement et d'inférence.

Nous entamerons ce chapitre par présenter nos expériences autour de l'architecture de notre encodeur audio, en réutilisant un meilleur modèle de PANN que précédemment (CNN14D au lieu de CNN10). Puis, nous détaillerons comment nous avons corrigé certains problèmes liés à la génération des phrases, pour créer un second système de référence SR2. Comme la performance d'un modèle de DTAA semble fortement liée à celle de son encodeur pré-entraîné, nous présenterons un nouvel encodeur nommé ConvNeXt, qui remplacera le CNN14D dans notre système. Nous verrons ensuite comment nous avons amélioré notre système de DTAA à l'aide d'augmentations spécifiques à cette tâche ainsi qu'à l'aide de corrections appliquées sur le jeu de données d'entraînement d'AC. Enfin, nous étudions dans une dernière partie les différents biais liés aux fuites de données présentes entre les différents jeux, et nous étudierons plus particulièrement le cas d'AudioSet (AS) et d'AC.

Sommaire

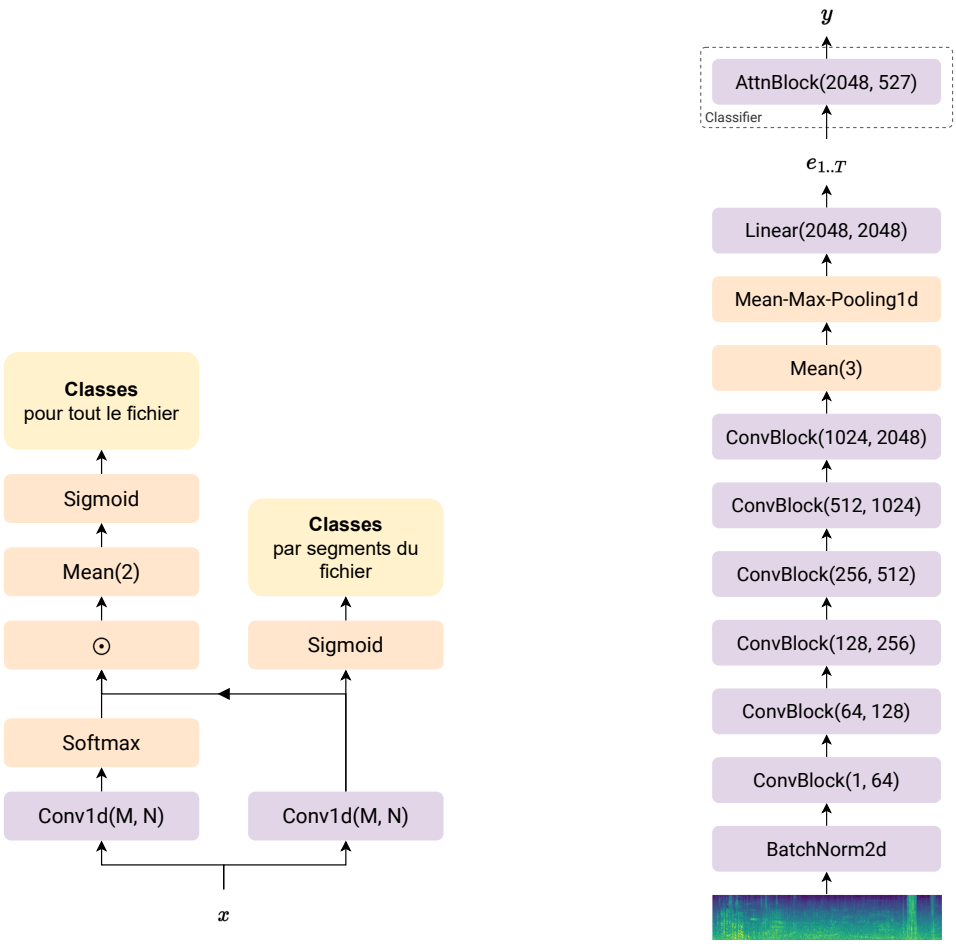
5.1 Réglage d'hyperparamètres

Pour la suite de nos expériences, nous avons augmenté le nombre de graines utilisées pour nos expériences de 3 à 5, car nous avons observé de grands écarts de confiance de SPIDEr. Nous avons également commencé à utiliser l'optimiseur AdamW au lieu d'Adam (décrits dans la section 1.2.5), car il est théoriquement plus efficace et il est également utilisé dans plusieurs des meilleurs systèmes en DTAA [Koizumi 2020c, Gontier 2021, Wu 2023b]. Comme c'est pratique courante avec AdamW, la pénalité sur les poids n'est pas appliquée aux biais du réseau mais seulement aux poids, car les biais ne contribuent pas au surapprentissage ni à la variance des activations du réseau. Ce changement d'optimiseur, en gardant les mêmes valeurs de pas d'apprentissage et de pénalité de poids, n'a pas d'impact significatif sur les résultats de notre modèle.

Nous avons également décidé d'utiliser la métrique FENSE comme critère d'arrêt lors de l'apprentissage plutôt que la fonction de coût, car cela permet de prendre une époque plus lointaine dans l'entraînement et permet de véritablement évaluer la génération du modèle. Ensuite, un autre changement mineur des hyperparamètres se joue sur la taille maximale des candidats qui sera de 30 mots au lieu de 52 pour AC, pour éviter quelques longues phrases répétitives, ce qui permet d'accélérer la validation pour les premières époques d'entraînement. De plus, nous avons pu accéder au supercalculateur Jean-Zay [33], qui dispose de plus de ressources de calcul que notre équipe, et nous permet ainsi d'entraîner le modèle plus longtemps, avec une plus grosse taille de lot. Par souci de comparaison avec les modèles qui suivront, nous avons augmenté le nombre d'époque d'entraînementde 50 à 100 pour AC et de 50 à 400 pour CL, la taille du lot de 8 à 128 et ajouté une accumulation des gradients de 4, ce qui simule une taille de lot de 512. Les résultats du CNN10-trans sont légèrement inférieurs au SR1, à cause du surapprentissage du modèle dû au plus grand nombre d'époques, mais nous verrons que ce problème peut être résolu par la pénalité de poids dans le chapitre 6.

La performance d'un système de DTAA semble fortement dépendre de la performance de son encodeur pré-entraîné. Pour améliorer notre système de DTAA, nous donc avons testé un autre encodeur de PANN : le CNN14-DecisionLevelAtt (CNN14D). Ce dernier a été entraîné à prédire les classes d'événements sonores pour chaque trame et pourrait donc avoir une meilleure représentation temporelle que les autres modèles de PANN qui prédisent la classe au niveau du fichier entier. Le modèle CNN14D adopte la même architecture que le CNN10, mais possède une séquence de 6 `ConvBlock` au lieu de 4. Il contient un mécanisme d'attention nommé `AttnBlock` composé de deux convolutions unidimensionnelles avec un noyau de taille 1. Les sorties par trames sont moyennées sur l'axe temporel pour créer la sortie pour tout le fichier. La figure 5.1 montre l'architecture de l'`AttnBlock` et du CNN14D. Pour la DTAA, nous récupérons la séquence de représentations avant l'`AttnBlock` pour représenter le fichier audio. À noter que nous avons également refait certaines de ces expériences postérieurement avec le modèle CNN14 de PANN, qui donne des résultats de DTAA assez similaires. Le tableau 5.1 résume les caractéristiques des différents encodeurs étudiés, sans leurs couches servant à la classification audio (AT). Dans tous les cas, une couche de projection viendra réduire la taille des représentations produites par chaque encodeur vers la taille attendue par le décodeur (256). CNN10 est un modèle bien plus petit que le CNN14D, mais bien moins performant sur AS en mAP pour la tâche d'AT.

33. http://www.idris.fr/jean-zay/



(a) AttnBlock. (b) CNN14D.

FIGURE 5.1 – Architecture du CNN14D et de son module `AttnBlock`. Mean(i) désigne la moyenne selon le i-ème axe du tenseur d'entrée.

TABLE 5.1 – Performance et caractéristiques des différents encodeurs pour l'AT. La taille des représentations correspond à celle d'un fichier de dix secondes.

Encodeur	**mAP** (↑)	**Params.** (M, ↓)	**MACs** (G, ↓)	**Taille des représentations**
CNN10	0,380	**4,7**	**37,5**	62×512
CNN14D	**0,425**	79,7	52,8	31×2048

Les résultats de nos expériences sur l'encodeur sont présentés dans le tableau 5.2. Sur AC, l'encodeur CNN14D est légèrement plus performant que le CNN10, qu'il soit gelé ou non. Nous pouvons cependant noter que l'apprentissage est beaucoup plus rapide lorsqu'il est gelé. Pour CNN10-trans, le modèle est meilleur si l'encodeur n'est pas gelé, mais ce n'est pas le cas pour le CNN14D-trans. Sur CL, le CNN14D-trans ne surpasse pas le CNN10-trans lorsque les deux encodeurs ne sont pas gelés, mais est légèrement supérieur lorsqu'ils le sont. La meilleure configuration dépend donc beaucoup du jeu de données ciblé, et fait également varier la durée

des expériences. Pour la suite, nous avons décidé de nous focaliser sur AC, car il s'agissait du plus grand jeu de données de DTAA disponible à l'époque, ce qui nous a poussés à sélectionner CNN14D-trans avec l'encodeur gelé.

TABLE 5.2 – Résultats de DTAA des différents encodeurs par jeu de données et architecture (en %).

Test	Encodeur	Encodeur gelé	SD ↑	FNS ↑	Vocab ↑	Durée ↓
AC	CNN10	Non	39,7±1,1	58,0±0,5	406,6±27,3	11h38m
	CNN10	Oui	38,5±1,1	58,0±0,5	**506,6±31,0**	**4h05m**
	CNN14D	Non	40,3±2,2	59,0±1,1	442,2±81,7	15h54m
	CNN14D	Oui	**41,7±1,6**	**59,9±0,8**	471,0±33,7	5h22m
CL	CNN10	Non	**23,5±1,1**	**46,7±0,7**	**875,0±61,8**	14h37m
	CNN10	Oui	21,0±1,2	44,4±0,3	722,8±108,6	**7h11m**
	CNN14D	Non	21,4±1,6	45,2±0,5	685,0±155,3	15h21m
	CNN14D	Oui	22,1±1,1	45,3±0,3	718,6±101,3	8h34m

5.2 Amélioration de la recherche en faisceaux

Durant nos expériences, nous avons observé plusieurs problèmes dans les candidats générés par nos modèles. Pour y remédier, nous avons décidé de nous focaliser sur l'algorithme de recherche en faisceaux en modifiant certains aspects de la génération.

5.2.1 Génération contrainte

Nous avons proposé deux légères modifications sur l'algorithme de recherche en faisceau. Le premier changement consiste à empêcher le système de générer des phrases trop courtes, voire vides. La prédiction du modèle avant la fonction d'activation softmax (*logit*), est modifiée à $-\infty$ pour la balise `<eos>` pendant les L_{min} premiers mots. La seconde modification vise à interdire au modèle de produire certains mots déjà prédits dans la phrase actuelle afin de limiter les répétitions d'événements (comme dans *« a man speaks and a man speaks »*). Cela est réalisé de la même façon que pour `<eos>` pour les mots concernés.

Nous donnons les résultats de la première modification dans le tableau 5.3. Nous avons également reporté le taux d'erreur de fluence (FER), car il détecte les phrases mal structurées. Bien que la méthode de génération modifiée permette d'éviter quelques cas de phrases vide, cela ne concerne qu'une minorité de candidats (de 0 à 5 phrases en fonction des expériences sur les 912 fichiers d'AC-test). Les scores SPIDEr, FENSE ou Vocab restent donc presque inchangés, jusqu'à atteindre la valeur L_{min} de 4. La performance commence alors à légèrement diminuer, car le modèle commence à décrire des événements non présents ou à en répéter certains. Nous avons cependant noté que cette contrainte pouvait légèrement augmenter le taux d'erreur de fluence FER, car le modèle de détection d'erreur ne considère pas les phrases vides comme erronées. Les phrases générées par le modèle à la place des phrases vides ne sont pas toujours cohérentes comme pour *« crowd applause a »* ou *« a a a »* (avec L_{min}= 3). Nous avons cependant décidé de

garder cette valeur pour la suite de nos expériences, pour que notre modèle produise toujours une réponse.

TABLE 5.3 – Résultats de contrainte L_{min} sur AC (en %).

L_{min}	**SD ↑**	**FNS ↑**	**Vocab ↑**	**FER ↓**
0	41,7±1,6	59,9±0,8	471,0±33,7	2,0±1,1
1	41,7±1,6	59,9±0,7	471,0±33,7	2,2±1,0
2	41,7±1,6	59,9±0,7	471,0±33,7	2,2±1,0
3	41,7±1,6	59,9±0,8	471,0±33,7	2,3±1,1
4	40,9±1,3	59,8±0,7	479,8±38,0	2,4±1,0
5	39,5±1,4	59,6±0,8	496,2±39,1	2,8±1,0

Pour appliquer la seconde stratégie, nous avons d'abord essayé d'empêcher la répétition de tous les mots, puis de seulement les mots non vides (*non-stopwords*). Nous utilisons la liste anglaise de *stopwords* issue de la bibliothèque NLTK [34] (*Natural Language ToolKit*). Cependant, le système peut toujours répéter certains événements en utilisant des mots légèrement différents ou des synonymes comme dans *« a child speaks and children speak »*. Nous avons également remarqué que les événements sonores de certaines phrases pouvaient être altérés par cette méthode, comme la phrase *« a man speaks and a man speaks »* qui est devenue *« a man speaks and a woman talks »*, mais cela reste assez minoritaire.

Le tableau 5.4 donne les résultats des deux stratégies combinées, en fonction du type de répétition autorisée (avec $L_{min} = 3$). Si nous interdisons de répéter tous les mots, nous pouvons voir une perte de SPIDEr, mais un léger gain de FENSE dû à la baisse du FER, lui-même causé par la baisse du taux de répétitions des mots. La diminution du score SPIDEr provient probablement du fait qu'autoriser la répétition de mots permet au modèle de générer une phrase qui a plus de chances de contenir le bon n-gramme utilisé dans les références. La méthode autorisant les *stopwords* obtient des résultats supérieurs, améliorant légèrement le SPIDEr, FENSE et FER. Nous avons donc décidé de retenir cette dernière stratégie, qui reste facile à implémenter et n'augmente qu'assez peu le coût d'inférence du modèle.

TABLE 5.4 – Résultats de méthode d'interdiction de répétitions sur AC (en %).

Répétition autorisée	**SD ↑**	**FNS ↑**	**Vocab ↑**	**FER ↓**
Tous	41,7±1,6	59,9±0,8	471,0±33,7	2,3±1,1
Aucun	39,7±1,4	60,0±0,6	**493,2±30,6**	1,1±0,6
Stopwords	**41,8**±1,5	**60,7**±0,6	484,0±31,8	**0,9**±0,8

34. https://www.nltk.org/howto/corpus.html#word-lists-and-lexicons

5.2.2 Recherche en faisceau par lot

Depuis la mise en place de la validation par la métrique FENSE, l'algorithme de recherche en faisceau est utilisé à la fin de chaque époque d'entraînement sur le sous-ensemble de validation. Cependant, la plupart des implémentations ne traitent qu'un seul exemple à la fois, et non pas un lot entier, ce qui ralentit beaucoup nos expériences. Pour les accélérer, nous avons implémenté une version de cet algorithme plus rapide, qui empile les N_{beam} entrées audio sur la dimension du lot B. L'empilement crée un lot de taille $P = N_{beam} \cdot B$, et permet de calculer toutes les sorties du modèle en une seule passe. La raison de cet empilement est double : le modèle ne supporterait pas une entrée audio de 4 axes ($N_{beam} \times B \times E \times S$) et le nombre de faisceaux de chaque élément du lot doit pouvoir diminuer indépendamment lorsqu'une phrase est terminée, ce qui ne correspond plus à un tenseur si on gardait 4 axes. Il faut donc garder en mémoire les tailles de faisceaux courants et ainsi que les indices de chaque élément dans le lot empilé pour connaître à quel fichier audio correspond chaque élément.

Algorithm 1 Pseudo-algorithme de recherche en faisceau standard

Require: Modèle f, A de taille $B \times E \times S$, N_{beam}, L_{max}, V
  $S \in \mathbb{R}^P$
  $C_{out} = \emptyset$
  $S_{out} = \emptyset$
  for $j \in [1..B]$ **do**
    $A' \leftarrow$ Répétition de l'audio A_j pour obtenir un lot de taille $N_{beam} \times E \times S$
    $C \leftarrow [[\texttt{<bos>}] \mid \forall i = 1..N_{beam}]$
    $N_{beam,j} \leftarrow N_{beam}$
    for $i \in [1..L_{max}]$ **do**
      $p \leftarrow f(A', C)$
      $p_{top}, i_{top} = \text{Top-k}(p, N_{beam,j})$
      Met à jour les candidats C, en fonction des meilleurs prochains mots indiqués par i_{top}
      Met à jour les scores S, en fonction des meilleurs prochains scores p_{top}
      Supprime de C les phrases terminées pour les placer dans C_{out}
      Supprime de S les scores correspondant aux phrases terminées pour les placer dans S_{out}
      Supprime de A' les audios correspondant aux phrases terminées
      Décrémente le nombre de faisceaux actuels $N_{beam,j}$ selon le nombre de phrases terminées
    end for
  end for

Le fonctionnement de la recherche en faisceau standard et par lot sont respectivement donnés par les pseudo-algorithmes 1 et 2. f correspond au modèle de DTAA, qui prend en entrée un lot de représentations audio et un lot des mots précédents de chaque phrase. La variable A correspond aux fichiers audio sous forme de lot de taille $B \times E \times S$, où E et S sont respectivement les dimensions des représentations et de l'axe temporel. Les variables V, N_{beam}, L_{max} correspondent respectivement à la taille du vocabulaire du modèle, la taille du faisceau et la taille maximale des candidats. Pour dupliquer les fichiers audio sur le premier axe des tenseurs, j'utilise la répétition entrelacée, qui est une forme de répétition des éléments d'une liste. Par exemple, répétition entrelacée par deux de la liste [0, 1, 2] résulte en la liste [0, 0, 1, 1, 2, 2]. L'algorithme par lot est approximativement dix fois plus rapide que la version par élément avec notre modèle, et permet de réduire drastiquement les durées d'apprentissage. Plus précisément, les durées sont diminuées

Algorithm 2 Pseudo-algorithme de recherche en faisceau par lot

Require: Modèle f, A de taille $B \times E \times S$, N_{beam}, L_{max}, V
  $A \leftarrow$ Répétition entrelacée de A selon le premier axe pour obtenir un lot de taille $P \times E \times S$
  $I \leftarrow$ Répétition entrelacée de $[0..B[$ pour obtenir P indices
  $C \leftarrow [[\texttt{<bos>}] \mid \forall i = 1..P]$
  $S \in \mathbb{R}^P$
  $C_{out} \leftarrow \emptyset$
  $S_{out} \leftarrow \emptyset$
  for $i \in [1..L_{max}]$ **do**
    $p \leftarrow f(A, C)$
    for $j = 1..B$ **do**
      $N_{beam,j} \leftarrow$ Nombre de faisceaux restants pour l'audio j, indicées par I
      $p_j \leftarrow$ Probabilités pour l'audio j, indicées par I, de taille $N_{beam,j} \times V$
      $p_{top}, i_{top} = \text{Top-k}(p_j, N_{beam,j})$
      Met à jour les candidats C, en fonction des meilleurs prochains mots indiqués par i_{top}
      Met à jour les scores S, en fonction des meilleurs prochains scores p_{top}
      Supprime de C les phrases terminées pour les placer dans C_{out}
      Supprime de S les scores correspondant aux phrases terminées pour les placer dans S_{out}
      Supprime de A les audios correspondant aux phrases terminées
      Supprime de I les indices correspondant aux phrases terminées
    end for
  end for

de 2h30 à 1h45 pour AC et de 6h30 à 2h pour CL sur Jean-Zay avec le modèle SR3 (que nous décrirons dans la suite de ce chapitre dans la section 5.7). Le code source de l'algorithme par lot est également disponible sur GitHub [35].

5.2.3 Pré-calcul des représentations de l'encodeur

Pour le chapitre suivant, nous avions besoin de plus d'espace mémoire pour nos expériences, ce qui nous a poussé à pré-calculer les représentations issues de l'encodeur sur un disque. Cela permet non seulement de ne pas charger en mémoire l'encodeur, mais diminue la place mémoire prise par l'entrée audio (car les spectrogrammes sont plus grands que les séquences produites par le CNN14D). Cependant, ce pré-calcul nécessite de supprimer l'augmentation SpecAugment qui s'applique sur les spectrogrammes, ce qui diminue les performances. Le lissage des étiquettes reste la seule augmentation appliquée.

Table 5.5 – Résultats du CNN14D-trans sur AC selon le pré-calcul des représentations et l'application du SpecAugment (en %). Repr. est ici le diminutif pour Représentations.

Système	Repr. précalculées	SA	SD ↑	FNS ↑	Vocab ↑	Durée ↓
CNN14D-trans	Non	Oui	**41,8**±1,5	**60,7**±0,6	484,0±31,8	5h22m
CNN14D-trans (SR2)	Oui	Non	39,7±0,8	59,6±0,4	**485,8**±54,7	**2h30m**

35. https://github.com/Labbeti/conette-audio-captioning/blob/main/src/conette/nn/decoding/beam.py#L26

Le tableau 5.5 montre ce nouveau système de référence 2 (SR2) sur AC. Même si le score SPIDEr et FENSE diminuent fortement (-2,1% et -1,1% de perte), la durée d'entraînement de chaque expérience est deux fois plus rapide. Le tableau en annexe A.1 détaille les valeurs finales des différents hyperparamètres du SR2.

5.3 Pénalité des poids

Pour la suite de ce chapitre, nous utilisons une valeur différente pour l'hyperparamètre λ_{wd}, qui sera mis à 2. Le chapitre suivant 6 détaille l'impact de cette valeur, qui réduit le surapprentissage de façon draconienne. Il augmente aussi la performance de notre système pour les métriques SPIDEr et FENSE, mais pas pour le vocabulaire produit. Le tableau 5.6 donne le score de cette nouvelle référence sur AC.

TABLE 5.6 – Résultats du CNN14D-trans sur AC par λ_{wd} (en %).

Système	λ_{wd}	**SD**	**FNS**	**Vocab**
CNN14D-trans (SR2)	10^{-6}	39,7±0,8	59,6±0,4	**485,8**±54,7
CNN14D-trans	2	**44,4**±1,3	**61,9**±0,3	375,0±19,6

Nous avons également remarqué que le modèle ne converge pas avec la nouvelle valeur de λ_{wd} si nous entraînons à nouveau la partie encodeur, car cela doit trop modifier ses poids. Cependant, pour nous approcher encore plus de l'état de l'art sur AC, nous pensons que notre modèle nécessite d'utiliser une meilleure représentation de l'audio, ce qui nous a poussé à entraîner notre propre encodeur audio nommé ConvNeXt.

5.4 ConvNeXt

ConvNeXt (CNext) [Liu 2022b] est un réseau de neurones dont l'architecture repose notamment sur des convolutions profondes séparables (*Depthwise Separable Convolutions*, DSC). Il a été à l'origine conçu pour la classification d'image, et obtient des performances similaires aux modèles fondés sur les *Transformer* avec moins d'opérations et de paramètres. Les DSC sont des convolutions elles-mêmes composées de deux convolutions classiques : une convolution « séparable » (*depthwise convolution*, ConvDW) qui traite chaque canal indépendamment les uns des autres, et une convolution « point-à-point » (*pointwise convolution*, Conv1x1) qui mélange les canaux entre eux à l'aide d'un filtre de taille 1×1 (ce qui équivaut à une projection linéaire). Les figures 5.2 et 5.3 illustrent la différence entre une convolution standard et une DSC.

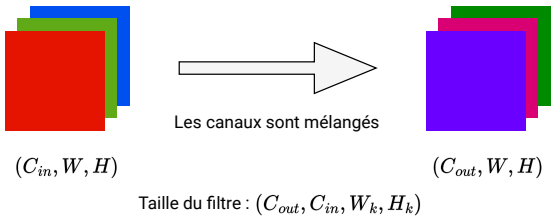


FIGURE 5.2 – Convolution standard avec C_{in} canaux d'entrée et C_{out} canaux de sortie.

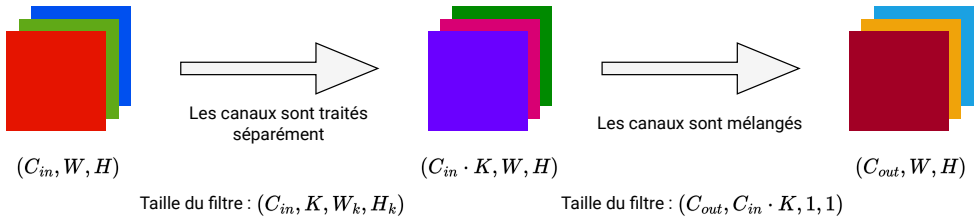


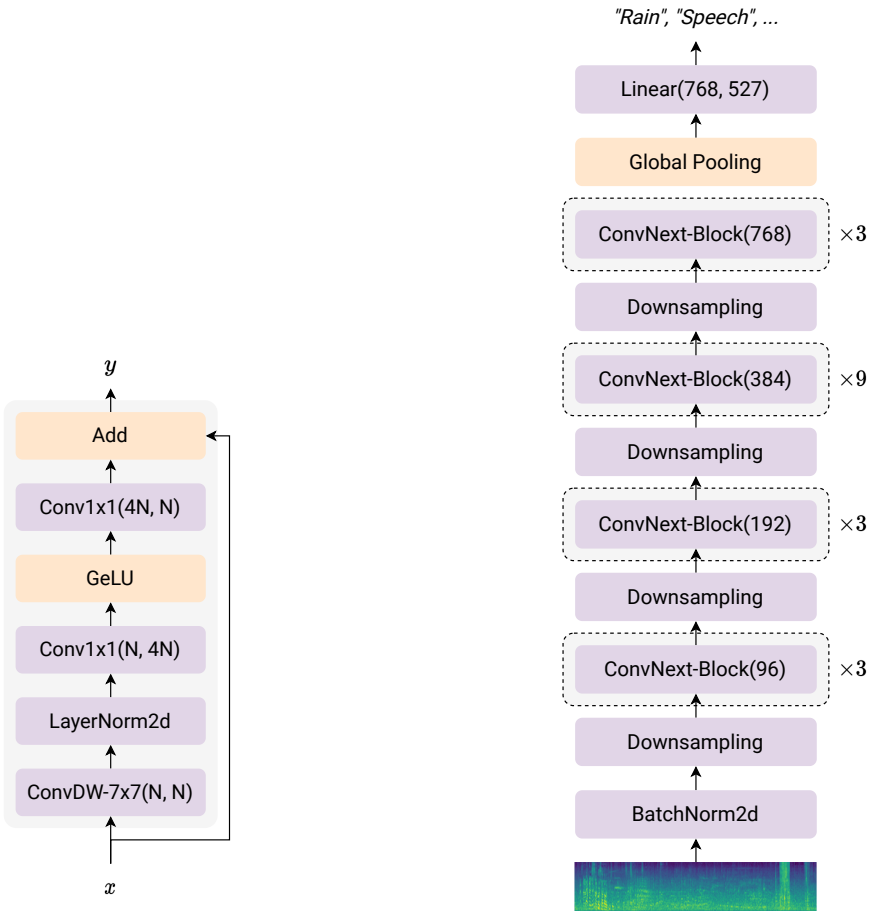
FIGURE 5.3 – DSC avec C_{in} canaux d'entrée et C_{out} canaux de sortie.

L'architecture CNext utilise les DSC dans des blocs décrits par la figure 5.4a. Contrairement aux modèles de PANN, ces blocs ne modifient pas la taille entre l'entrée et la sortie, ce qui permet de les empiler facilement, d'appliquer les connections résiduelles ainsi que certaines augmentations spécifiques comme le *drop path* [Huang 2016]. Chaque bloc commence par une convolution séparable, une normalisation par couche, une convolution point-à-point qui mélange et augmente le nombre de canaux, une fonction d'activation GELU, une seconde convolution point-à-point qui rétabli le nombre de canaux à celui d'origine et enfin une connexion résiduelle entre l'entrée et la sortie. Les blocs sont empilés pour former quatre séquences séparées par des couches de sous-échantillonnage, qui sont des convolutions classiques suivies d'une normalisation. Ces couches réduisent l'axe fréquentiel et temporel et augmentent la taille des représentations, d'une manière similaire aux couches d'*AvgPool2D* des modèles de PANN. Comme pour ces derniers, la fin du modèle moyenne les représentations sur l'axe fréquentiel et calcule la somme entre la moyenne et le maximum sur l'axe temporel. Une dernière couche de classification projette cette représentation finale vers le nombre de classes du jeu de données.

Mon encadrant de thèse Thomas Pellegrini a adapté l'architecture de CNext pour l'entraîner sur une tâche de classification audio sur le jeu AS [Pellegrini 2023]. Plus précisément, il a repris l'architecture ConvNeXt-Tiny, qui est un modèle CNext composé de quatre séquences de 3-3-9-3 blocs de taille 96, 192, 384 et 768, et représenté par la figure 5.4b. Même si le modèle contient moins de paramètres que certains modèles de PANN, il souffre toujours de surapprentissage, ce qui nécessite d'introduire plusieurs mécanismes pour le limiter. Pour cela, il a utilisé l'optimiseur AdamW avec une pénalité de poids importante ($\lambda_{\text{wd}} = 0{,}05$), ainsi que de nombreuses augmentations. On y retrouve le Mixup et le SpecAugment qui sont également utilisés en DTAA, mais aussi le *drop path* qui masque certains blocs de manière aléatoire durant l'apprentissage. La dernière augmentation appliquée est la perturbation de vitesse (*Speed Perturbation*) [Ko 2015], qui rééchantillonne le signal brut à une fréquence légèrement plus élevée ou plus faible, puis tronque ou complète le résultat pour conserver la même taille de signal. Les résultats sur AS de notre modèle ainsi que d'autres modèles comparables sont présentés dans le tableau 5.7. Les modèles comparés sont ceux de PANN (CNN14 et CNN14D), un premier modèle de type *Transformer* nommé *Audio Spectrogram Transformer* (AST), l'état de l'art supervisé lui aussi fondé sur l'architecture *Transformer* (PaSST-S), et une autre implémentation du modèle ConvNeXt-Tiny pour l'audio issu de la bibliothèque Audax [36].

Le modèle ConvNeXt-Tiny surpasse les autres modèles en performance, à l'exception du modèle PaSST-S, pour lequel il obtient le même score (0.471 de mAP). Cependant, notre modèle est plus petit que PaSST-S en nombre de paramètres, avec seulement 28,2 millions comparés à 87 millions.

36. https://github.com/SarthakYadav/audax



(a) Architecture d'un bloc du CNext. (b) Architecture globale du modèle ConvNeXt-Tiny.

FIGURE 5.4 – Illustration de l'architecture du CNext et de son module.

TABLE 5.7 – Performance des modèles de classification audio sur AS-eval.

Modèle	**Params. (M)** ↓	**mAP** ↑
CNN14 [Kong 2020]	80,7	0,431
CNN14D [Kong 2020]	80,8	0,425
AST [Gong 2021]	88,0	0,459
PaSST-S [Koutini 2022]	87,0	**0,471**
ConvNeXt-Tiny (Audax)	**28,2**	0,402
ConvNeXt-Tiny (nous)	**28,2**	**0,471**

Cette étude montre que les modèles convolutifs peuvent être aussi ou même plus performants que les modèles de type *Transformer*, si on modernise leur architecture. Le code source de l'inférence du modèle et les poids sont disponibles sur GitHub [37] sur HuggingFace [38].

37. https://github.com/topel/audioset-convnext-inf
38. https://huggingface.co/topel/ConvNeXt-Tiny-AT

Une fois le modèle CNext entraîné, nous pouvons appliquer de l'extraction de représentations comme avec les modèles de PANN. Notre nouveau modèle de DTAA sera nommé CNext-trans dans la suite. Le CNext produit des représentations avec la même résolution temporelle que ceux de PANN, et chaque vecteur a pour taille 768. Comme pour notre précédent système SR2, nous avons gardé le pré-calcul des représentations du CNext pour accélérer nos apprentissages. Les résultats sont présentés dans le tableau 5.8.

TABLE 5.8 – Résultats des modèles CNN14D-trans et CNext-trans sur AC (en %). Les meilleurs scores de nos systèmes sont en **gras**.

Modèle	**SD**	**FNS**	**Vocab**
Multi-TTA [Kim 2022]	47,5	N/A	N/A
CNN14D-trans	44,4±1,3	61,9±0,3	375,0±19,6
CNext-trans	**47,1**±1,5	**63,8**±0,4	**409,6**±21,4

Le modèle CNext-trans surpasse CNN14D-trans pour les trois métriques avec un gain important de 1,7% de SPIDEr, un gain de 1,7% de FENSE et 26,0 mots uniques supplémentaires. De plus, le CNext-trans arrive à atteindre un score SPIDEr avec 47,1%, ce qui est très proche de l'état de l'art Multi-TTA qui obtient 47,5%. Ceci confirme une fois de plus que la performance de l'encodeur audio conditionne fortement la performance du système de DTAA qui l'utilise, et que le modèle CNext fournit de bien meilleures représentations. Pour aller encore plus loin, nous avons décidé d'explorer les augmentations de données pour la DTAA, car notre système actuel n'en utilise qu'une seule.

5.5 Augmentation de données

5.5.1 Mixup

La première augmentation que nous étudions est le Mixup, que nous avons déjà détaillé dans la section 1.4.4.2. Cette augmentation reste en général assez indépendante du type de données traitées, et pourrait permettre au modèle de mieux reconnaître des superpositions d'événements. Toute la difficulté d'appliquer cette augmentation réside dans l'ajout des phrases, qui ne peuvent pas être sommées aussi facilement que des vecteurs *one-hot* pour la classification mono-étiquettes. Nous avons sélectionné quatre variantes de Mixup qui pourrait s'appliquer dans le contexte de la DTAA. En premier lieu, nous testons le **MixGen** et le **Mixup-in**, tous deux déjà décrits dans la section 1.4.4.2. Les deux autres variantes étudiées ici sont le **Mixup-sep** et le **Mixup-cat**. Dans la suite de cette section, nous employons les mêmes notations que dans la section 1.4.4.2, que nous rappelons ici. Nous notons une paire audio-texte (x_1, y_1) qui peut être mélangée avec une autre paire du lot notée (x_2, y_2). x_{mix} désigne la somme de x_1 et x_2, pondérée par λ et $1-\lambda$. Les suffixes *prev* et *next* indiquent respectivement les mots précédents et suivants nécessaires durant l'entraînement. Par exemple, si x_1 est « `<bos>` *a man speaks* `<eos>` », $x_{1,prev}$ désigne « `<bos>` *a man speaks* » (l'entrée du modèle) et $x_{1,next}$ désigne « *a man speaks* `<eos>` » (la sortie attendue du modèle).

Mixup-sep est la variante du Mixup proposée dans la première implémentation officielle sur CIFAR-10 par Facebook [39]. Son fonctionnement est décrit par les équations 5.1. f correspond au modèle qui prend en entrée un audio x et la liste des mots précédents de la phrase pour produire un *logit* noté z. Il a pour avantage de mélanger les valeurs des fonctions de coût et non les étiquettes elles-mêmes, ce qui contourne le problème de l'ajout des phrases.

$$\begin{aligned} z_1 &= f(x_{mix}, y_1) \\ z_2 &= f(x_{mix}, y_2) \\ \mathcal{L} &= \lambda \,\mathrm{CE}(z_1, y_1) + (1-\lambda)\,\mathrm{CE}(z_2, y_2) \end{aligned} \tag{5.1}$$

Mixup-cat consiste à concaténer à la fois les audios, les mots précédents ainsi que les mots suivants durant l'entraînement. Il permet au modèle d'apprendre à reconnaître des fichiers audio plus longs, tout en lui apprenant à générer des phrases plus longues. De plus, le modèle pourra donc apprendre à mieux reconnaître et décrire des séquences d'événements si les phrases sont concaténées dans le bon ordre. Cette méthode diffère de MixGen, car les deux fichiers audios sont concaténés au lieu d'être sommés. Pour éviter que le modèle ne s'entraîne que sur des exemples concaténés, nous appliquons cette augmentation selon une probabilité p pour chaque exemple du lot.

$$\begin{aligned} x_{1,2} &= \mathrm{Cat}(y_1, x_2) \\ y_{1,2,prev} &= \mathrm{Cat}(y_{1,prev}, y_{2,prev}) \\ y_{1,2,next} &= \mathrm{Cat}(y_{1,next}, y_{2,next}) \\ z_{1,2} &= f(x_{1,2}, y_{1,2,prev}) \\ \mathcal{L} &= \mathrm{CE}(z_{1,2}, y_{1,2,next}) \end{aligned} \tag{5.2}$$

À noter que dans nos expériences, nous avons décidé de conserver le pré-calcul des représentations audio par l'encodeur, ce qui signifie que nous mélangeons les représentations et non les spectrogrammes ou signaux bruts. Les résultats des différentes variantes du Mixup sont présentés dans le tableau 5.9. Le MixGen n'apporte pas de gain sur notre modèle, et dégrade même beaucoup les performances. Nous pouvons supposer que notre version était trop naïve et demanderait à faire en sorte que les deux fichiers audios sommés soient tous les deux audibles, comme pour le ChatGPT-Mixup (décrit dans la section 1.4.4.2). Surprenamment, le Mixup-in asymétrique améliore légèrement les performances, avec un score SPIDEr de 48,8% et un FENSE de 64,0%. Nous avons brièvement cherché deux autres valeurs pour α, mais elles n'ont pas donné de meilleurs résultats. Le Mixup-sep dégrade légèrement les performances, probablement pour la même raison que le MixGen. Enfin, le Mixup-cat n'apporte pas non plus de gain, et dégrade même les performances si l'augmentation est trop présente. Cependant, nous pouvons noter que cela peut grandement contribuer à la diversité du modèle, en augmentant le vocabulaire de 409,6 à 583,0 en moyenne.

Nous avons également essayé quelques variantes autour du Mixup-in pour essayer de mieux comprendre pourquoi mélanger seulement les entrées améliorait les performances. Nos variantes enlèvent soit le mélange de l'audio, soit celui des représentations des mots précédents en entrée du décodeur, ou bien mélangent en plus les étiquettes. Cependant, ces variantes dégradent légèrement les performances, comme le montre les résultats donnés en annexe dans le tableau E.1. Pour

39. https://github.com/facebookresearch/mixup-cifar10

TABLE 5.9 – Résultats des variantes du Mixup pour le modèle CNext-trans sur AC-test.

Aug.	α	**Asym.**	**p**	**SD**	**FNS**	**Vocab**
Aucune	0	Oui	0	47,1±1,5	63,8±0,4	409,6±21,4
MixGen	0,4	Non	1	30,1±1,1	60,4±0,6	464,6±35,8
Mixup-in	0,4	Non	1	45,7±0,7	62,3±0,5	280,2±5,2
Mixup-in	0,4	Oui	1	**48,8**±0,8	**64,0**±0,2	397,6±19,7
Mixup-in	0,2	Oui	1	48,3±1,2	**64,0**±0,2	406,2±19,6
Mixup-in	0,8	Oui	1	47,8±1,5	63,7±0,4	396,4±25,5
Mixup-sep	0,4	Non	1	46,5±1,4	63,6±0,4	405,6±15,5
Mixup-cat	N/A	Non	1	35,0±1,2	63,8±0,6	**583,0**±45,2
Mixup-cat	N/A	Non	0,1	46,7±0,9	63,6±0,3	395,0±51,9
Mixup-cat	N/A	Non	0,2	46,8±1,5	63,7±0,5	404,4±27,0
Mixup-cat	N/A	Non	0,4	46,4±1,8	63,4±0,4	387,0±17,3

la suite de ce chapitre, nous avons donc sélectionné le Mixup-in asymétrique qui ne s'applique que sur les entrées du décodeur, avec $\alpha = 0{,}4$ et $p = 1$. Comme ce Mixup mélange les entrées et notamment l'audio sans modifier les étiquettes, on peut s'attendre à ce que le modèle soit moins sensible aux bruits additifs.

5.5.2 Mixup+SpecAugment

Une autre augmentation que nous avons décidé d'ajouter à notre système est le SpecAugment (SA), car elle reste assez indépendante du type de tâche. Cependant, nous ne l'appliquons pas sur le spectrogramme, mais sur la séquence de représentations produite par le modèle CNext. De plus, comme les fichiers audios peuvent être de longueur variable, nous avons modifié le code pour que la taille des zones masquées dépende d'un ratio de la taille du fichier audio sur les axes temporels et les axes des représentations. Nous nommons cette variante le SA-ratio, et nous avons recherché quelques valeurs pour trouver des ratios sur les tailles de masques entre 0% et 10% et un nombre maximal de bandes égal à 2 pour les deux axes. Les résultats sont présentés dans le tableau 5.10. L'augmentation obtenue est bien moins importante que lorsque nous l'utilisions sur le CNN14D-trans, car notre modèle souffre moins de surapprentissage. Nous pouvons donc constater un léger gain sur les trois métriques, bien qu'il ne soit pas très significatif.

TABLE 5.10 – Résultats du Mixup+SA du modèle CNext-trans sur AC-test.

SA-ratio	**SD**	**FNS**	**Vocab**
Non	48,8±0,8	64,0±0,2	397,6±19,7
Oui	**48,9**±1,0	**64,2**±0,3	**408,0**±25,0

Par souci de comparaison avec les modèles suivants, nous avons augmenté le nombre de têtes d'attention du décodeur de 4 à 8, mais cela n'a aucun impact sur les scores du modèle.

5.6 Correction de l'entraînement AudioCaps

Malgré un processus d'annotation incluant une correction des descriptions par des annotateurs humains, le jeu de données AC-train contient de nombreuses erreurs que nous avons observées durant nos expériences. Nous avons donc décidé de catégoriser les plus flagrantes et d'en corriger une partie manuellement pour améliorer la qualité de ce sous-ensemble d'entraînement. Pour repérer les mots suspects, nous nous fondons sur les mots les moins fréquents et vérifions s'ils sont valides. Cela permet de facilement repérer des erreurs typographiques, mais aussi des informations visuelles ou parlées qui ne devraient pas se trouver dans les descriptions. Les différents types d'erreurs trouvées sont listés ici :

— Annotation incorrecte comme *« No audio no video »*, *« Broken link not working »* ;

— Informations visuelles comme *« A man **in his 40s** is making a salad a demonstrating his cooking abilities »* ;

— Informations liées à la parole comme *« Woman describing **a cooperative survey about Wisconsin frogs and toads** »* ;

— Entités nommées comme *« A telephone rings and **mickey mouse** speaks »* ;

— Typographique comme *« Womam »* au lieu de *« Woman »* ou *« Continous »* au lieu de *« Contin**u**ous »* ;

— Mots non homogènes comme *« mid sized »* avec *« midsized »* ou *« ring tone »* avec *« ringtone »*.

En fonction de l'erreur, la description est modifiée si possible, sinon elle est supprimée. Les erreurs typographiques sont les plus communes. Nous avons également supprimé les phrases trop longues contenant plus de 40 mots, car elles sont très répétitives. Après toutes ces corrections, 968 phrases ont été modifiées, 28 phrases ont été supprimées parmi les 49 838 descriptions d'origine de l'ensemble d'entraînement. Les descriptions de ce nouveau jeu sont téléchargeables et utilisables dans notre bibliothèque `aac-datasets`, mais également disponibles en version brute sur GitHub [40]. Les résultats du modèle entraîné sur AC-train avant et après ces corrections sont donnés dans le tableau 5.11. Malgré toutes ces corrections, les résultats du système n'ont été que très légèrement améliorés, et la différence reste inférieure à l'intervalle de confiance. Cependant, nous avons observé une réduction importante de la taille du vocabulaire du modèle, qui a diminué de 4724 à 4590 mots uniques.

Table 5.11 – Résultats du CNext-trans sur AC avec les références corrigées ou non (en %).

Correction	SD	FNS	Vocab
Non	48,9±1,0	64,2±0,3	**408,0±25,0**
Oui	**49,5**±1,3	**64,3**±0,3	393,0±44,5

40. https://raw.githubusercontent.com/Labbeti/aac-datasets/main/data/train_v2.csv

5.7 Biais et fuite de données

La tâche de DTAA est une tâche complexe qui nécessite généralement un grand nombre de paires de descriptions audio. Comme mentionné dans la section 1.4.1.3, la plupart des systèmes utilisent l'apprentissage par transfert ou de l'extraction de représentations par un modèle pré-entraîné pour l'AT, généralement sur AS. Cependant, nous avons remarqué que l'ensemble des données AC est en réalité un sous-ensemble de l'entraînement d'AS, ce qui signifie qu'un encodeur entraîné sur AS a déjà vu les fichiers audio d'AC, même ceux de validation et de test. Cela implique donc que l'encodeur connaît déjà les événements sonores de la partie de validation et de test d'AC. La figure 5.5 résume ce biais (ou fuite) de données que nous voulons mettre en évidence. Sur cette figure, la première phase consiste à pré-entraîner un réseau sur une tâche de classification sur AS-train, la seconde à réutiliser une partie de ses poids (l'encodeur) pour réaliser la DTAA sur AC-train, et la troisième à tester le modèle de DTAA appris sur le jeu AC-test.

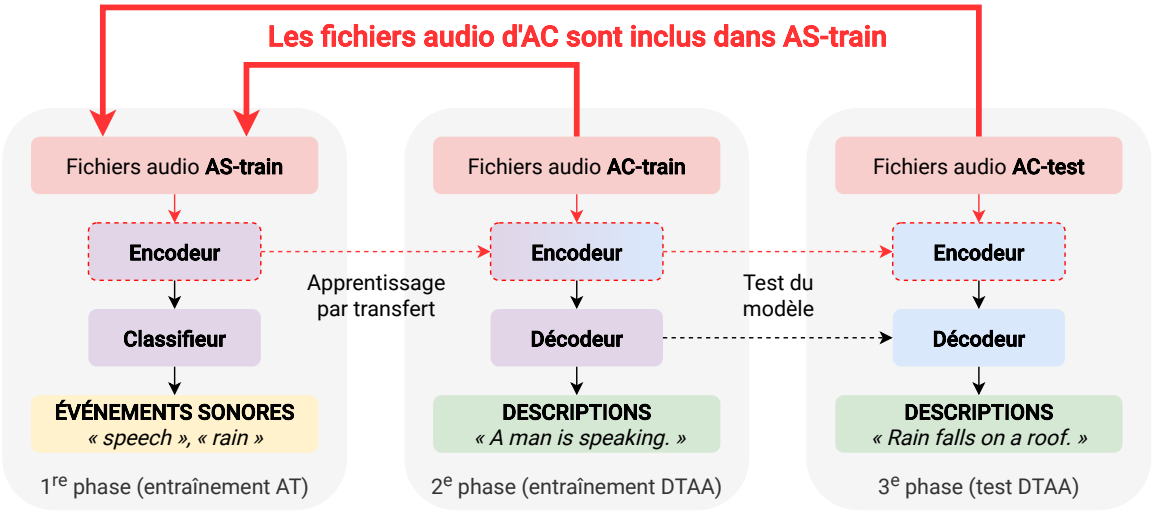


FIGURE 5.5 – Illustration de la fuite de données par apprentissage par transfert ou extraction de représentations entre AC et AS.

Ce biais concerne toutes les tâches audio-texte (DTAA, RAT...) qui exploitent AS et AC dans leur procédure d'entraînement. De la même manière, il concerne également tous les cas où deux ensembles de données utilisés se chevauchent. En outre, nous avons étudié les proportions de chevauchement avec d'autres ensembles de données qui partagent une source commune. CL et l'ensemble de données d'AT FSD50K [Fonseca 2021] proviennent tous deux de FreeSound, et présentent une faible intersection d'environ 5%. L'ensemble de données WC contient des fichiers provenant de FreeSound et d'AS, qui partagent des sources communes avec CL et AC. Le biais lié à l'utilisation de ce grand ensemble de données signifie que les données de test peuvent être partiellement ré-étiquetées dans la partie entraînement par des descriptions différentes. Cela ajoute un autre biais, plus grave que l'utilisation de modèles pré-entraînés, qui doit être pris en compte lors de l'évaluation des modèles sur CL et AC. Nous avons détaillé les proportions de sources de chevauchement entre les ensembles de données CL, AC, AS, WC et FSD50K dans le tableau 5.12. Nous fournissons également en annexe I le détail des intersections entre les différents sous-ensembles de chaque jeu de données.

TABLE 5.12 – Intersection entre les sources des jeux de données de DTAA et d'AT (en %).

Jeu A	Jeu B	|A ∩ B|/|A|	Étiquettes du jeu B
AC	AS-train	100,00	Classe
CL	FSD50K-train	5,38	Classe
AC	WC	17,62	Description
CL	WC	89,05	Description

Pour étudier l'impact de la fuite de données, nous avons entraîné les modèles CNN14 et CNext sur le jeu de données AS, sans aucun fichier audio d'AC. À noter que nous avons choisi le modèle CNN14 au lieu du modèle CNN14D ici, car ce dernier ne convergeait pas correctement dans nos expériences sur AS. Nous comparons en premier lieu la performance de ces modèles pour la tâche AT sur l'ensemble de validation et de test d'AS et AC dans le tableau 5.13.

TABLE 5.13 – Résultats mAP pour différents modèles sur AS-test, AC-val et AC-test. Les valeurs en **gras** ou soulignées indiquent respectivement notre meilleur score biaisé et non biaisé par colonne. AS^{-AC} se réfère au jeu AS sans données issues d'AC. CNN14* indique que nous réutilisons les poids de nos entraînements du CNN14 et non ceux de PANN.

Modèle	Poids	Entraînement	AS-test	AC-val	AC-test
CNN14	PANN	AS	0.431	0.717	0.647
CNN14*	Nous	AS	0.441	0.755	0.727
CNext	Nous	AS	**0.471**	**0.774**	**0.749**
CNN14*	Nous	AS^{-AC}	0.434	0.642	0.537
CNext	Nous	AS^{-AC}	0.465	0.669	0.565

L'impact de l'exclusion des données d'AC avec les modèles CNN14 et CNext montrent une légère baisse de mAP de 0,7% et 0,6% sur le test d'AS, ce qui reste marginal pour la tâche AT. Cependant, lorsque que nous regardons la performance de ces systèmes de AC-val et AC-test, nous observons une large différence comparée aux données de test d'AS, confirmant que le modèle est beaucoup plus performant sur les données sur lesquelles il a été entraîné. Lorsque nous enlevons les données d'AC, nous observons une plus grande baisse de performance de 11,3% pour CNN14 et 18,4% pour CNext. Ces derniers résultats semble indiquer que les modèles de DTAA utilisant un encodeur entraînés sur AS et testés sur AC sont surestimés. Pour valider cette hypothèse, nous avons donc réutilisé les poids de ces différents encodeurs sur AC pour étudier cet impact. Nous ajoutant également les résultats sur CL, pour vérifier que le modèle a bien généralisé sur des données différentes.

Par souci de lisibilité, le nombre de paramètres de notre système avec ceux de l'état de l'art est donné séparément dans le tableau 5.14, avec leurs références. Le symbole † de CNN14-trans† désigne le modèle proposé dans l'étude [Won 2021], pour éviter de le confondre avec notre propre CNN14-trans qui n'utilise pas les mêmes hyperparamètres. À l'exception de ce modèle CNN14-trans† [Won 2021], les modèles de la littérature sont bien supérieurs en taille aux nôtres, dépassant tous les 100 millions de paramètres.

TABLE 5.14 – Nombre de paramètres des différents systèmes de l'état de l'art.

Test	Système	Params. (M)	
		appris	gelés
AC	HTSAT-BART [Mei 2023]	171	0
	Multi-TTA [Kim 2022]	108	0
	PYB [Gontier 2021]	494	0
CL	BEATs-BART [Wu 2023b]	127	1500
	CNN14-BART [Mei 2023]	219	0
	CNN14-trans† [Won 2021]	8	75,5
AC/CL	CNN14-trans	12,3	75,5
	CNext-trans (SR3)	12,0	28,2

Les résultats des différents encodeurs sont présentés dans le tableau 5.15. Enc* indique le potentiel biais de l'encodeur. WC* se réfère au jeu WC sans intersection avec les sous-ensembles de validation et de test d'AC et de CL. Les références de chaque modèle de l'état de l'art sont données dans le tableau 5.14. Le modèle CNext-trans dont l'encodeur est pré-entraîné sur AS^{-AC} est appelé SR3, et servira de référence pour les chapitres 7 et 8. Les hyperparamètres détaillés de SR3 sont donnés en annexe dans le tableau A.1.

Pour AC, nous rapportons les résultats de systèmes récents de la littérature, dans lesquels nous avons identifié un biais présumé dû à leurs encodeurs pré-entraînés, puisqu'ils n'ont pas mentionné que les fichiers AC ont été retirés d'AS pendant le pré-entraînement AT de leurs encodeurs. À notre connaissance, il n'y a pas de système dans la littérature de DTAA qui prenne en compte ce biais. En ce qui concerne nos systèmes, le modèle CNext-trans a surpassé CNN14-trans dans tous les contextes, comme nous nous y attendions étant donné sa meilleure performance dans l'AT. Notre modèle CNext-trans biaisé a surpassé l'état de l'art précédent de 1,0% en valeur absolue, sans utiliser de données externes et avec un ordre de grandeur de moins de paramètres entraînables (12,3 contre 171 millions). Cependant, le modèle CNext-trans non biaisé, qui n'a pas utilisé les fichiers audio AC lors de la préformation (AS^{-AC}), a connu une baisse de points de 2,9% en valeur absolue, ce qui représente seulement une baisse relative de -5,9%. Cela indique que la fuite de données dans ce modèle pré-entraîné a encore un impact sur la performance pour la tâche de DTAA, lié à la réduction de la performance AT du modèle (qui était une baisse relative de -21,9%). Les scores FENSE semblent être corrélés avec les valeurs SPIDEr et démontrent que l'encodeur CNext a fourni de meilleures représentations à la partie décodeur, à la fois avec et sans biais de données. Le nombre de mots utilisés varie de 353 à 396 mots, mais n'est pas toujours corrélé avec le SPIDEr.

Sur CL, le modèle CNext-trans continue de surpasser les autres modèles et atteint une performance proche de CNN14-BART, bien qu'il ait moins de données d'entraînement et de paramètres (40 contre 219 millions). Nous avons atteint le score de l'état de l'art pour les modèles qui n'utilisent pas de données externes. De plus, nous observons que la représentation plus riche fournie par CNext permet au décodeur de produire des descriptions avec une plus grande diversité de mots (545 types de mots avec CNN14-trans comparé à 636 avec CNext-trans).

TABLE 5.15 – Résultats de DTAA sur les jeux de données AC-test et CL-eval. Les valeurs en **gras** ou soulignées indiquent respectivement notre meilleur score biaisé et non biaisé. Entr. est ici le diminutif pour Entraînement.

Test	Système	Données		Biais	SD	FNS	Vocab
		Pré-entr.	Entr.				
AC	humain	∅	∅	N/A	55,9±1,2	68,0±0,2	944,0±18,6
	HTSAT-BART	AS	AC+CL+WC*	Enc*	48,5	N/A	N/A
	Multi-TTA	AS	AC	Enc*	47,5	N/A	N/A
	PYB	AS	AC	Enc*	46,5	N/A	N/A
	CNN14-trans	AS	AC	Enc	44,3±0,8	61,5±0,3	383,2±1,6
	CNN14*-trans	AS	AC	Enc	45,6±1,1	62,5±0,2	390,4±22,0
	CNext-trans	AS	AC	Enc	**49,5**±1,3	**64,3**±0,3	393,0±44,5
	CNN14*-trans	AS^{-AC}	AC	Non	43,9±0,8	60,7±0,3	354,0±22,9
	CNext-trans (SR3)	AS^{-AC}	AC	Non	46,6±1,0	63,3±0,4	**396,4**±53,0
CL	humain	∅	∅	N/A	56,7±0,5	57,4±0,2	1818,2±17,1
	BEATs-BART	AS	AC+CL	Non	32,6	N/A	N/A
	CNN14-BART	AS	AC+CL+WC*	Non	31,0	N/A	N/A
	CNN14-trans†	AS	CL	Non	28,5	N/A	N/A
	CNN14-trans	AS	CL	Non	26,5±0,5	48,2±0,2	500,2±17,4
	CNN14*-trans	AS	CL	Non	27,4±0,3	49,1±0,2	545,2±20,3
	CNext-trans	AS	CL	Non	29,9±0,6	51,2±0,6	**636,8**±45,1
	CNN14*-trans	AS^{-AC}	CL	Non	25,9±0,5	47,7±0,1	522,6±10,1
	CNext-trans (SR3)	AS^{-AC}	CL	Non	**30,1**±0,5	**51,6**±0,3	628,2±46,2

De plus, nous remarquons que notre score SPIDEr sur AC est proche de celui du score humain avec seulement 6,4% de différence absolue. Cependant, le score SPIDEr sur CL reste beaucoup plus bas que le score humain, avec une différence absolue de 26,2%. Les différences de scores FENSE sont toutes deux assez proches (3,7% et 5,7%), ce qui indique que le modèle produit des descriptions qui sont sémantiquement proches des références, mais qui peinent à contenir les bons n-grammes. Curieusement, nous pouvons remarquer que l'emploi de l'encodeur CNext semble augmenter l'intervalle de confiance du vocabulaire produit par rapport au CNN14, indiquant que les candidats sont plus ou moins diversifiés en fonctions des initialisations.

5.8 Bilan

Dans ce chapitre, nous avons dressé la liste des améliorations faites à notre système de DTAA, en commençant par améliorer la partie encodeur avec les représentations issues d'un modèle de PANN plus performant : le CNN14D. Ensuite, nous nous sommes focalisés sur la partie décodeur, en modifiant et optimisant la recherche en faisceau pour contraindre la génération des mots prédits et pour améliorer la vitesse d'inférence et de validation. Ainsi, nous avons obtenu le modèle CNN14D-trans, nommé SR2 et utilisé dans le chapitre 6. Par la suite, nous avons

introduit un nouvel encodeur, ConvNeXt, qui surpasse les modèles de PANN sur AS grâce à une architecture convolutive plus récente et évoluée. Avec cette extraction de représentations, ce modèle de classification fournit des caractéristiques plus discriminantes que ceux de PANN pour notre modèle de DTAA nommé CNext-trans, le rapprochant beaucoup de l'état de l'art sans données externes sur AC avec 47,1% comparé à 47,5% de SPIDEr. Pour aller plus loin, nous avons expérimenté plusieurs variantes de l'augmentation de données Mixup et SpecAugment, et nous en avons trouvé une combinaison qui améliore les performances de notre modèle qui surpasse l'état de l'art pour atteindre 48,9% de SPIDEr. Puis, nous avons corrigé certaines descriptions du jeu de données AC-train pour améliorer une fois de plus notre modèle, qui obtient 49,5% de SPIDEr et dépasse l'état de l'art avec données externes précédent (48,5% de SPIDEr).

Sur CL, le modèle CNext-trans atteint l'état de l'art sans données externes et sans RL (30,1%), mais pas avec données externes (33,5%). Nous pouvons cependant noter qu'il reste bien plus petit que la plupart des systèmes utilisés dans la littérature. Enfin, nous avons repéré une fuite de données entre les jeux AC et AS, qui signifie que les modèles appliquant de l'apprentissage par transfert ou de l'extraction de représentations pourraient être partiellement entraînés sur des données utilisées durant le test. Pour étudier son impact, nous avons ré-entraîné des modèles sur AS en excluant AC, et nous avons observé une baisse de performance de notre système (de 1,7 à 2,9%). Le modèle de référence utilisé dans certains chapitres suivants est donc le CNext-trans est nommé SR3, dont l'encodeur a été entraîné sans fuite de données avec AC.

Pour la suite de ce manuscrit, nous étudierons l'impact d'une méthode multitâche sur le modèle SR2 dans le chapitre 6, pour essayer d'améliorer la sémantique des phrases produites par ce dernier. Puis, nous verrons comment nous pouvons essayer d'exploiter plusieurs jeux de données d'entraînement en même temps en partant du système SR3. L'objectif sera d'obtenir une bonne performance sur les sous-ensembles de test d'AC et de CL, à l'aide d'un unique système dans le chapitre 7.

Chapitre 6

Apprentissage multitâche pour la DTAA

Pour entraîner les systèmes à faire de la DTAA, l'immense majorité des modèles utilisent la fonction entropie croisée (CE) entre les sorties du modèle et la vérité terrain pour chaque mot de la phrase. Cette méthode reste cependant assez limitée pour la description automatique, car elle demande à ce que le système prédise exactement la phrase de référence, sans prendre en compte les synonymes, un décalage dans les mots ou une tournure de phrase légèrement différente. De plus, les systèmes de génération de texte entraînés avec la CE semblent produire des phrases plus répétitives, génériques et moins diverses en vocabulaire.

Dans ce chapitre, nous proposons d'étudier une méthode d'apprentissage multitâche visant à entraîner le modèle à générer des phrases ayant un sens similaire plutôt que de chercher à produire exactement les mêmes mots. Cet objectif sera atteint à l'aide d'un second terme ajouté à la fonction de coût, nommé *Sentence Embedding Regression* (SER), qui essaie de rapprocher la représentation (*embedding*) de la phrase produite par le modèle avec celle de la référence.

Sommaire

6.1 Motivation

Dans la littérature, quelques études ont proposé d'exploiter des modèles de génération de texte pré-entraîné pour la DTAA. La plupart d'entre elles utilisent un décodeur avec des poids initialisés et provenant d'un modèle de texte pour aider la génération. Cependant, cela semble impacter marginalement les scores à cause de la pauvreté des textes de DTAA par rapport au langage naturel en général. Cependant, d'autres stratégies utilisant des modèles de texte pré-entraînés ont vu le jour [Xu 2021b, Zhang 2023, Mahfuz 2023b, Wu 2023b]. Ces approches sont similaires aux techniques de *Feature-based knowledge distillation*, qui exploitent les représentations latentes d'un modèle pré-entraîné pour conditionner celles du modèle principal. Plus particulièrement, elles proposent souvent d'ajouter un second terme à la fonction de coût qui contraint une représentation intermédiaire de l'encodeur ou du décodeur (souvent avant la couche de classification finale) avec une représentation à taille fixe de la référence.

Nous proposons ici d'explorer ce type d'approche en ajoutant un terme de régression sur les représentations des phrases (*Sentence Embedding Regression*, SER). Ce terme a pour objectif de faire générer au modèle des phrases ayant un sens plus proche de la référence, sans essayer de retrouver les mêmes mots. Par exemple, le tableau 6.1 donne un exemple de candidat pour un fichier audio d'AC-val[41] avec les références associées. Nous remarquons que le score SPIDEr est très faible, car seul le mot *« a »* correspond entre les phrases, alors que leur sens est très similaire. Pourtant, le score SB_{sim} reste relativement élevé. Pour entraîner le modèle à générer des phrases sémantiquement proches, nous utiliserons le modèle de texte pré-entraîné de la métrique SB_{sim}. La principale différence de notre approche par rapport aux autres réside dans le fait que la sortie de l'unité lexicale prédite par notre modèle sera utilisée en entrée du modèle SBERT, et non simplement la moyenne des sorties de notre modèle ou la dernière des représentations de la phrase.

Table 6.1 – Candidat et références du fichier « `y682ml90jGw` » issu d'AC-val, avec les scores associés (en %).

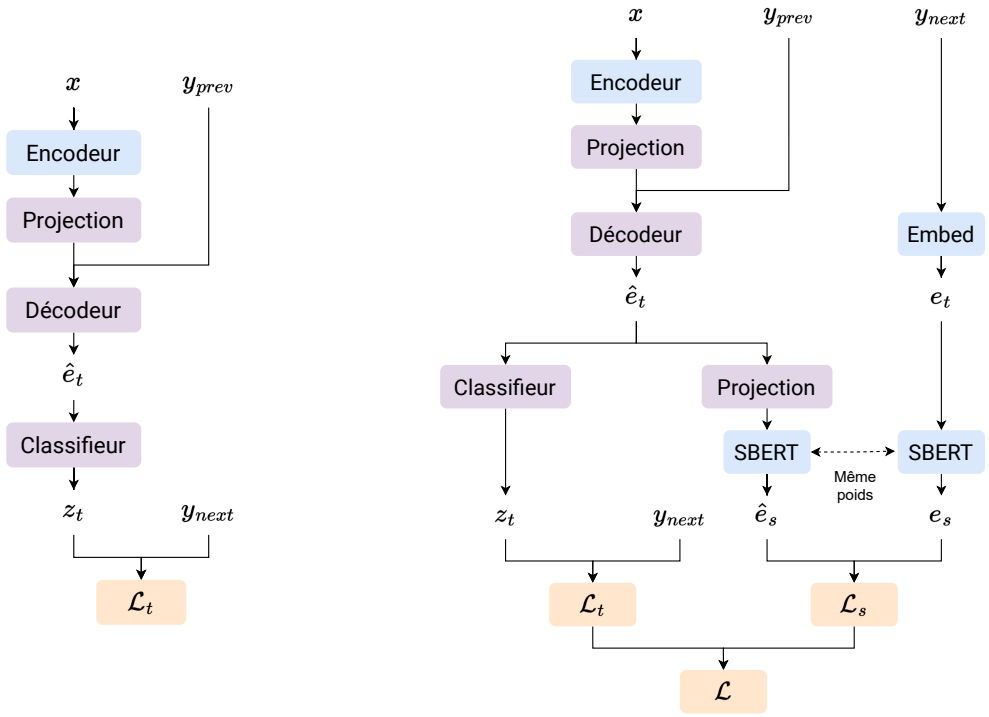
N°	**Candidats**	**SD**	**SB_{sim}**
C1	*A beeping sound is present*	10^{-3}	75,8
N°	**Références**		
R1	*A machine beeps continuously*		
R2	*Electronic beeps occur in a short series*		
R3	*A beep repeats multiple times*		
R4	*A beep sounds several times*		
R5	*Beeps occur continuously*		

6.2 Régression sur les représentations des phrases

Durant l'apprentissage, nous cherchons à entraîner les représentations de chaque unité lexicale suivante du modèle à l'aide du modèle SBERT. La figure 6.1 présente l'ensemble de la procédure

41. https://www.youtube.com/embed/y682ml90jGw?start=11&end=21

et des couches utilisées.



(a) Système de référence sans SER.

(b) Système avec SER.

FIGURE 6.1 – Méthode d'entraînement du système de référence comparé à celui utilisant SER. x correspond à un audio et y à une description. Les blocs **bleus** correspondent à des couches pré-entraînées et gelées, les blocs **violets** à des couches initialisées aléatoirement et apprenantes et les blocs **orange** à des fonctions.

La première différence avec notre système de référence se joue sur l'utilisation du même *tokenizer* que celui des modèles SBERT. En effet, il est nécessaire d'avoir la même taille de phrases que le modèle SBERT utilisé. En l'occurrence, les descriptions sont découpées en unités *WordPiece* (section 1.4.1.2) au lieu de mots : elles forment un vocabulaire de 30 522 unités différentes. À noter que ces unités ne sont pas nécessairement toutes utilisées durant l'apprentissage de DTAA. La seconde modification apportée au système est l'ajout d'une couche linéaire de projection entre les représentations du modèle de DTAA et l'entrée du modèle SBERT, qui n'ont pas la même dimension (respectivement 256 et 768). Cette projection est apprise durant l'entraînement par le second terme de la fonction de coût. Le dernier changement est la suppression de la première couche du modèle SBERT (nommée *Embed*) qui permet de faire correspondre les indices des *tokens* à leurs vecteurs. Cette opération est nécessaire pour permettre au gradient de se rétropropager dans le modèle SBERT et de DTAA. En effet, si on utilisait les indices des *tokens* suivants prédits par le système, l'opération argmax qui récupérerait l'unité lexicale la plus probable ne serait pas différentiable. D'autres techniques existent pour contourner ce problème [Bengio 2013, Jang 2017], modifiant la dérivée du argmax pour permettre la propagation du gradient. La modification du modèle SBERT a nécessité d'altérer son code source pour extraire la première couche, qui ne pouvait pas prendre en entrée des représentations.

Durant la phase d'apprentissage, nous nous servons les caractéristiques audio et les unités lexicales précédentes de vérité terrain pour générer les représentations des unités suivantes nommées $\hat{e}_t$. Ces représentations sont utilisées dans deux parties différentes du modèle. Tout d'abord, elles sont projetées vers des *logits* à l'aide d'un classifieur pour le premier terme de la fonction de coût $\mathcal{L}_t$. Puis, ces représentations $\hat{e}_t$ sont également projetées de 256 à 768 dimensions pour correspondre à l'entrée du modèle SBERT légèrement modifiée. La représentation résultante est comparée avec celle de la vérité de terrain pour le terme de la fonction de coût $\mathcal{L}_s$. Pendant l'inférence, seule la branche du classifieur est active pour générer nos phrases. La fonction de coût finale 6.1 est la somme des deux composantes $\mathcal{L}_t$ et $\mathcal{L}_s$, pondérées par un hyperparamètre λ_{SER} :

$$\mathcal{L} = \mathcal{L}_t + \lambda_{\text{SER}} \cdot \mathcal{L}_s \tag{6.1}$$

Au-delà de ceux déjà présents dans le système, cette méthode ajoute d'importants hyperparamètres : le choix du second terme de fonction de coût $\mathcal{L}_s$, l'architecture et les poids du modèle SBERT et la valeur du coefficient λ_{SER}.

6.2.1 Sélection du modèle SBERT

Les modèles SBERT ont été entraînés sur des corpus de paires de phrases, annotées par des étiquettes qui indiquent l'un de ces trois types de relations : implication, neutralité et contradiction. Il est donc nécessaire de vérifier si ces modèles peuvent bien être utilisés pour évaluer des phrases de DTAA, et plus précisément pour les discriminer. Pour cela, nous créons des triplets de phrases, avec des descriptions positives et une négative. En DTAA, certains sous-ensembles contiennent plusieurs descriptions pour chaque fichier audio, qui peuvent être utilisées pour créer des paires positives. Cependant, il n'existe pas à notre connaissance de jeu de données annoté avec des paires de descriptions négatives. Nous allons donc en générer artificiellement en sélectionnant une description liée à un fichier audio différent. Nous utilisons le sous-ensemble AC-val, qui contient 464 fichiers audio avec cinq descriptions chacun, ce qui permet de créer 4 640 paires positives. Nous associons ensuite dix autres descriptions à chaque paire, ce qui résulte en 46 400 triplets de descriptions. Des exemples de triplets sont donnés dans le tableau 6.2. On remarque que les phrases positives ne partagent qu'assez peu de n-grammes en commun, grâce à la diversité du jeu de donnée CL. Cependant, certaines paires de phrases positives ne donne pas toujours la même information, comme pour les phrases C7 qui décrit de l'eau qui s'écoule et la phrase C8 qui indique une porte qui coulisse.

Afin d'évaluer la pertinence des représentations de chaque modèle, nous pouvons calculer une corrélation de Pearson entre la similarité cosinus attribuée à la paire positive avec la valeur 1 et la similarité cosinus attribuée à la paire négative avec la valeur -1. Une autre manière d'évaluer ces représentations consiste à calculer un score d'exactitude (*accuracy*) en vérifiant que la similarité cosinus de la paire positive est supérieure à celle de la paire négative :

$$\text{Exactitude}(p_{i,a}, p_{i,p}, p_{i,n}) = \mathbb{1}\left(\text{cos-sim}\left(f(p_{i,a}), f(p_{i,p})\right) > \text{cos-sim}\left(f(p_{i,a}), f(p_{i,n})\right)\right) \tag{6.2}$$

Nous comparons les principaux modèles disponibles provenant d'une bibliothèque de modèles de texte pré-entraînés nommée `sentence-transformers` [42] dans le tableau 6.3. Tous ces modèles

42. https://www.sbert.net/docs/pretrained_models.html

TABLE 6.2 – Exemples de triplets utilisés pour sélectionner un modèle SBERT. Pour un triplet i, la phrase source (ancrage) est notée $p_{i,a}$, la phrase positive $p_{i,p}$ et la phrase négative $p_{i,n}$.

N°	Type	Descriptions
C1	$p_{i,a}$	*A rough engine revs and sputters*
C2	$p_{i,p}$	*A car revs loudly a few times while idling*
C3	$p_{i,n}$	*A train chugs nearby while blowing a horn*
C4	$p_{i,a}$	*An infant cries and a woman speaks*
C5	$p_{i,p}$	*A baby fusses and cries as a woman speaks curtly*
C6	$p_{i,n}$	*Someone whistles briefly*
C7	$p_{i,a}$	*Water quietly rushes by while birds chirp in the background*
C8	$p_{i,p}$	*A door slides shut wind blows and birds chirp*
C9	$p_{i,n}$	*A clock sounds an alarm then ticktocks*

TABLE 6.3 – Corrélations et exactitudes des modèles SBERT pour un ensemble de triplets de phrases de DTAA. La meilleure valeur par colonne est en **gras**.

Modèle	Params. (M, ↓)	Corrélation (↑)	Exactitude (%, ↑)
`paraphrase-TinyBERT-L6-v2`	*67,0*	*0,795*	95,5
`all-mpnet-base-v2`	109,5	0,794	96,0
`all-distilroberta-v1`	82,1	0,784	95,7
`all-MiniLM-L12-v2`	33,4	0,791	95,9
`all-MiniLM-L6-v2`	22,7	0,776	95,6
`multi-qa-mpnet-base-dot-v1`	109,5	0,748	94,8
`multi-qa-distilbert-cos-v1`	66,4	0,764	95,1
`paraphrase-multilingual-mpnet-base-v2`	278,0	**0,811**	**96,1**
`paraphrase-albert-small-v2`	**11,7**	0,784	95,4
`multi-qa-MiniLM-L6-cos-v1`	22,7	0,753	94,7
`paraphrase-multilingual-MiniLM-L12-v2`	117,7	0,798	95,8
`paraphrase-MiniLM-L3-v2`	17,4	0,783	95,3
`distiluse-base-multilingual-cased-v1`	135,1	0,738	93,9
`distiluse-base-multilingual-cased-v2`	135,1	0,744	94,2
`all-mpnet-base-v1`	109,5	0,797	96,0
`all-MiniLM-L12-v1`	33,4	0,789	95,8

arrivent très bien à discriminer les paires positives des paires négatives, avec des exactitudes toutes supérieures à 93%. Les corrélations sont également très élevées (supérieures à 0,73), mais ne sont pas toujours liées à la taille du modèle. Par exemple, le modèle le plus petit (11,7 millions de paramètres) obtient une assez bonne performance de 95,4% d'exactitude, alors que certains modèles parfois dix fois plus larges en paramètres obtiennent une moins bonne performance. Le meilleur modèle semble être `paraphrase-multilingual-mpnet-base-v2`, un modèle fondé sur

MPNet [Song 2020]. Cependant, il s'agit également du modèle le plus gros, donc nous avons décidé de nous servir également du modèle `paraphrase-TinyBERT-L6-v2`, car il est utilisé dans les métriques SB_{sim} et FENSE, et son nombre de paramètres est plus réduit.

6.2.2 Sélection du second terme de la fonction de coût

Le second terme de la fonction de coût $\mathcal{L}_s$ doit être une fonction de régression comparant au moins deux vecteurs de représentations. Plusieurs fonctions ont été envisagées : la *CosEmbLoss*, *L1Loss* et *MSELoss*, respectivement fondées sur la similarité cosinus, la distance ℓ_1 et l'erreur quadratique moyenne (*Mean Squared Error*, MSE). Une quatrième fonction nommée la *SmoothL1Loss* a été testée, et applique la *L1Loss* ou *MSELoss* en fonction d'un hyperparamètre β. Elle est définie par l'équation 6.3. L'équation 6.4 décrit quant à elle la version générique de la *CosEmbLoss*, qui peut rapprocher deux vecteurs si $y = 1$ et les éloigner si $y = -1$, ce qui peut être utilisé pour exploiter des paires de vecteurs négatifs. Nous avons essayé a posteriori d'éloigner les représentations par des paires négatives, mais cela a dégradé les performances de notre modèle (-3% de SPIDEr), donc seul $y = 1$ sera utilisé dans la suite.

$$\text{SmoothL1Loss}_\beta(e_a, e_b) = \begin{cases} \text{MSELoss}(e_a, e_b) \cdot \frac{1}{2\beta} & \text{si } |e_a - e_b| < \beta \\ \text{L1Loss}(e_a, e_b) - \frac{\beta}{2} & \text{sinon} \end{cases} \tag{6.3}$$

$$\text{CosEmbLoss}(e_a, e_b, y) = \begin{cases} 1 - \text{cos-sim}(e_a, e_b) & \text{si } y = 1 \\ \max(\text{cos-sim}(e_a, e_b) - \epsilon, 0) & \text{si } y = -1 \end{cases} \tag{6.4}$$

6.3 Résultats

Après une recherche d'hyperparamètre, nous avons trouvé que le modèle `paraphraseTinyBERT-L6-v2`, combiné avec le terme *SmoothL1Loss* avec λ_{SER} mis à 100 obtenait les meilleurs résultats. Surprenamment, nous avons trouvé que le plus gros modèle fondé sur MPNet n'était pas meilleur que celui que nous avons choisi, et pouvait même dégrader les performances de 1 ou 2% de SPIDEr dans certains cas. Dans le tableau 6.4, nous reportons les résultats de notre modèle de référence, de notre modèle de référence avec les *tokens* de SBERT et du modèle entraîné avec le terme SER avec les hyperparamètres trouvés. Nous comparons nos systèmes avec deux de la littérature (Multi-TTA [Kim 2022] et PYB [Gontier 2021]) ainsi que le score humain sur plusieurs métriques, dont notamment le SPIDEr, FENSE et SB_{sim}. De plus, nous avons également ajouté les résultats d'une étude [Mahfuz 2023b] appliquant une méthode de régression très similaire à la nôtre, et testée sur le même jeu de données. Plus précisément, les auteurs de cette étude appliquent la fonction MSE pour le second terme de leur fonction de coût, utilisent l'encodeur CNN10 de PANN et leurs hyperparamètres sont différents des nôtres (30 époques et 32 éléments par lot). Dans leur cas, la méthode de régression leur permet un gain de 0,5% de SPIDEr et 1% de SB_{sim} par rapport à leur système de référence, qui reste bien moins performant que le nôtre et que l'état de l'art actuel (36,5% de SPIDEr). Leur système est noté CNN10-trans[‡] pour éviter de le confondre avec notre système SR1.

En premier lieu, l'introduction des unités lexicales de SBERT permet à notre système de s'améliorer : de 39,7% à 41,3% en SPIDEr et de 59,5% à 60,2% en FENSE. Puis l'utilisation du terme SER augmente une fois de plus ces scores, menant à 41,8% de SPIDEr et 60,7% de FENSE. Cependant, ce processus demande beaucoup plus de paramètres que le système de référence, et

TABLE 6.4 – Résultats de la méthode SER sur AC-test (en %). Les meilleurs scores de nos systèmes sont en **gras**.

Système	Optim.	λ_{wd}	SD	FNS	SB_{sim}	Params. (M)	
						appris	gelés
Humain	N/A	N/A	55,9±1,2	68,0±0,2	68,2±0,3	0	0
Multi-TTA	AdamW	10^{-6}	47,5	N/A	N/A	108	0
PYB		N/A	46,5			494	
CNN10-trans‡	Adam	N/A	36,5	N/A	54,5	14	0
+Terme SER			37,0		55,5		N/A
CNN14D-trans (SR2)	AdamW	10^{-6}	39,7±0,8	59,6±0,4	60,1±0,5	**12,4**	**79,7**
+Unités SBERT			41,3±1,9	60,2±0,8	60,7±0,9	25,7	**79,7**
+Terme SER			**41,8**±1,5	**60,7**±0,8	**61,4**±0,5	25,9	146,7

reste en dessous du Multi-TTA (47,5%). Nous pouvons également remarquer que, même si le terme SER est censé améliorer uniquement les scores FENSE, le score SPIDEr est également plus élevé. Cela implique que la sémantique des phrases prédites n'est pas la seule chose qui est impactée, et que le modèle a généré plus de n-grammes corrects.

6.3.1 Quel impact le terme SER a-t-il sur le modèle ?

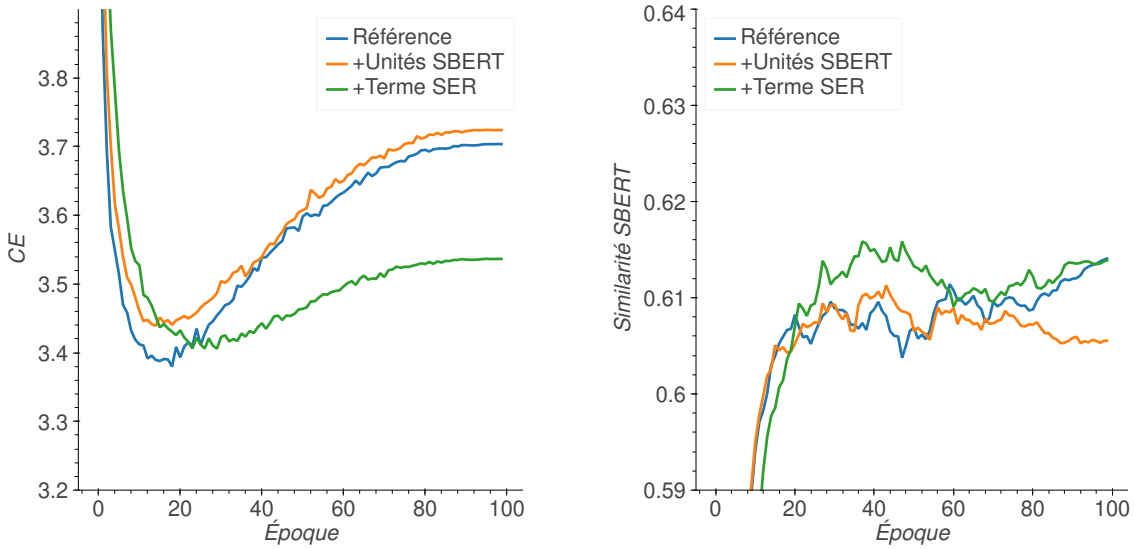
Pour mieux comprendre pourquoi le terme SER a amélioré les performances de notre système, nous avons reporté dans les figures 6.2 les valeurs de la fonction de coût et de SB_{sim} sur AC-val durant l'entraînement. Pour améliorer la lisibilité, les courbes SBERT sont lissées avec un coefficient de 0,75 à l'aide d'une moyenne continue non biaisée.

Nous remarquons sur la figure de la fonction de coût 6.2a que le modèle sur-apprend assez rapidement à partir de l'époque 20, et que le terme SER diminue cet effet. Cela a également été observé dans une autre étude [Çakır 2020] sur l'apprentissage multitâche en DTAA, qui a observé qu'un second terme de la fonction de coût pouvait régulariser le modèle. La courbe de SBERT est également légèrement plus élevée grâce au terme SER pendant la première partie de l'entraînement, ce qui semble indiquer que le modèle a mieux appris la sémantique des phrases générées. Face à ce constat, nous pouvons alors nous interroger sur l'impact qu'aurait le terme SER si nous ajoutions d'autres méthodes de régularisation sur notre modèle.

6.3.2 Régularisation par la pénalité des poids

Pour mieux limiter le surapprentissage, nous avons recherché une meilleure valeur de pénalité de poids (λ_{wd}). Ce dernier contrôle la moyenne continue des poids du modèle pour limiter leur norme, et permet d'éviter qu'elles n'atteignent des valeurs trop élevées, comme décrit dans la section 1.2.5 de l'état de l'art.

Les figures 6.3 détaillent la performance du modèle selon la valeur de λ_{wd} par la métrique SPIDEr, FENSE ou Vocab. Nous avons exploré l'espace des valeurs possibles majoritairement par des puissances de 10. La performance du modèle semble atteindre un optimum avec une valeur de λ_{wd} entre 1 et 5, avec un maximum de FENSE de 62,0% lorsque $\lambda_{wd} = 2$. Cette valeur est plutôt



(a) Valeurs de CE avec λ_{wd}=10^{-6}.

(b) SB_{sim} avec λ_{wd}=10^{-6}.

FIGURE 6.2 – Valeurs de CE et SB_{sim} sur l'ensemble de validation durant 100 époques d'entraînements.

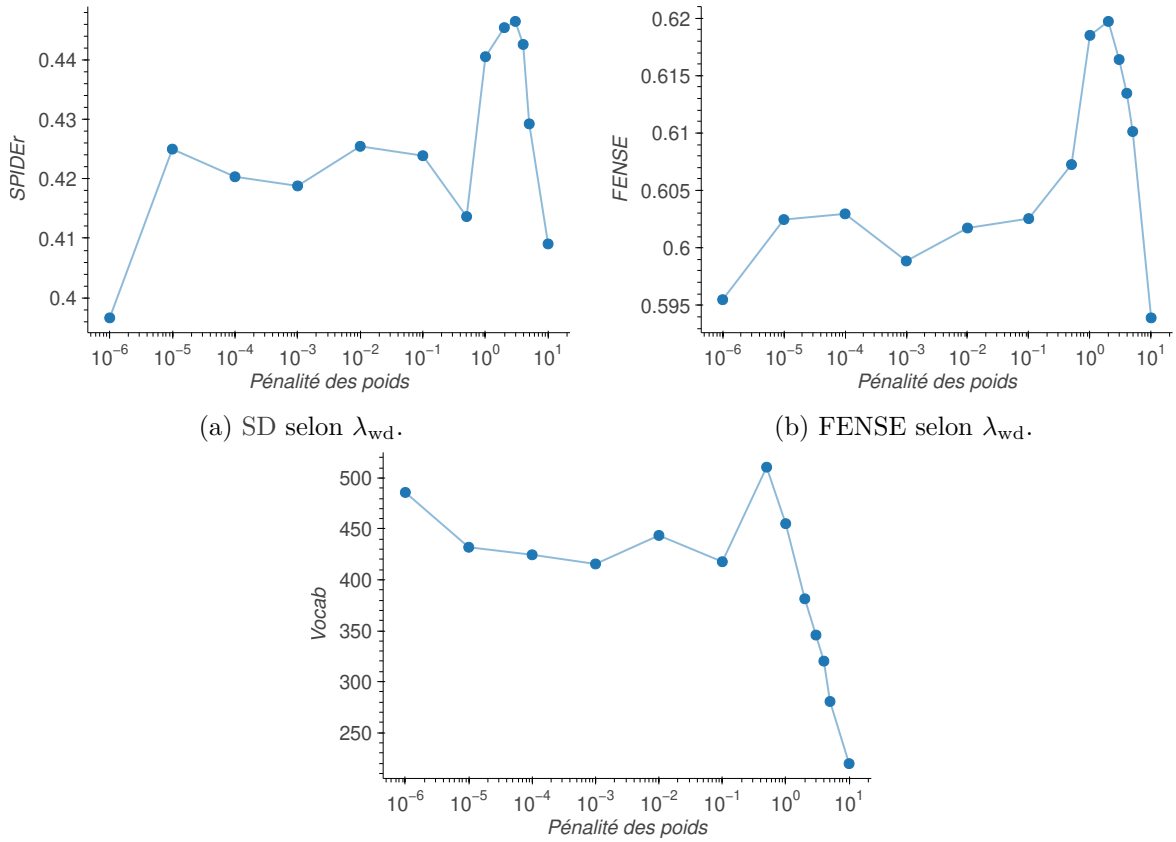
élevée en comparaison de celles utilisées dans la littérature, qui gardent bien souvent la valeur par défaut de l'optimiseur (10^{-6}). Des valeurs au-delà dégradent rapidement la performance du système sur toutes les métriques, et un λ_{wd} supérieur à 10 peine à converger. À noter que nous également essayé la valeur $\lambda_{wd} = 2$ avec l'optimiseur Adam, mais il ne semble pas converger correctement avec des valeurs de pénalités aussi élevées. Nous garderons donc la valeur 2 avec l'optimiseur AdamW pour la suite de nos expériences.

Avec cette nouvelle valeur de l'hyperparamètre λ_{wd}, les courbes de la fonction de coût sur AC-val ne croît plus (voir figure 6.4a), indiquant que le surapprentissage a été complètement empêché par la pénalité des poids. La performance du système de référence est également grandement améliorée avec des scores proches de 63,5% de SB_{sim} sur AC-val au de 61,5%.

TABLE 6.5 – Résultats améliorés de la méthode SER sur AC-test (en %). Les meilleurs scores de nos systèmes sont en **gras**.

Système	λ_{wd}	**SD** ↑	**FNS** ↑	**SB_{sim}** ↑	**FER** ↓	**Vocab** ↑
CNN14D-trans (SR2)	10^{-6}	39,7±0,8	59,6±0,4	60,1±0,5	3,4±0,1	**485,8**±54,7
+Unités SBERT		41,3±1,9	60,2±0,8	60,7±0,9	3,4±0,1	445,0±94,5
+Terme SER		41,8±1,5	60,7±0,8	61,4±0,5	2,7±1,8	465,6±60,1
CNN14D-trans	2	44,4±1,3	61,9±0,3	**62,1**±0,3	0,4±0,1	375,0±19,6
+Unités SBERT		**44,5**±2,0	**62,0**±0,6	**62,1**±0,6	**0,2**±0,1	377,6±13,3
+Terme SER		44,4±1,2	61,9±0,2	62,0±0,2	**0,2**±0,1	348,8±9,6

Si nous combinons la méthode SER avec les nouveaux hyperparamètres, la performance reste équivalente à la nouvelle référence, comme nous pouvons le voir dans le tableau 6.5. En effet,



(a) SD selon λ_{wd}.

(b) FENSE selon λ_{wd}.

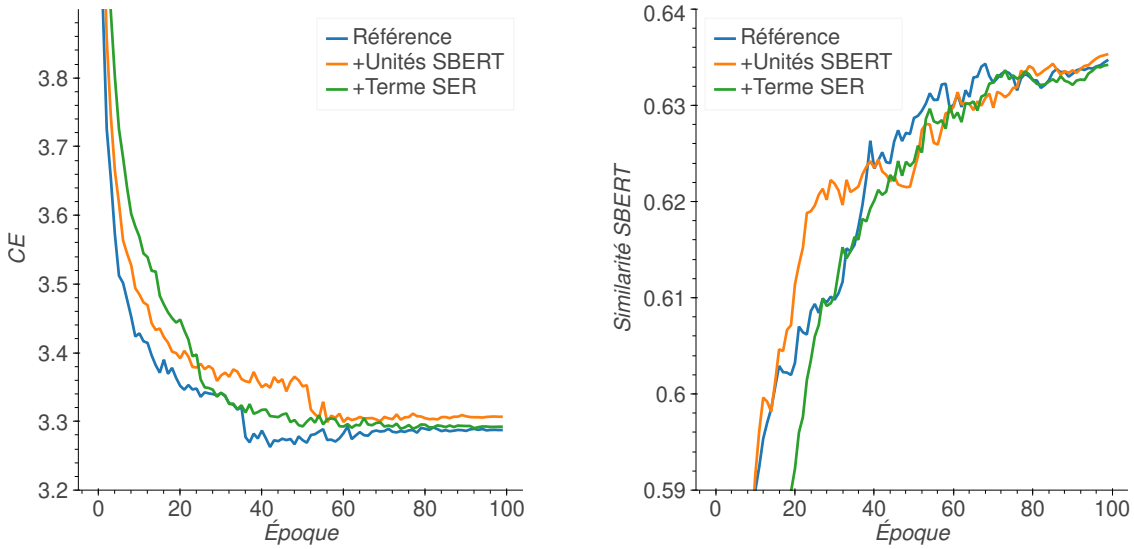
(c) Vocab selon λ_{wd}.

FIGURE 6.3 – Performance du modèle selon la pénalité des poids λ_{wd} avec AdamW.

nous obtenons 44,4% de SPIDEr avec $\lambda_{\text{wd}}= 2$, et le score est le même en ajoutant la méthode multitâche. Le taux d'erreur de fluence est également fortement diminué (3,4% à 0,4%) par la nouvelle pénalité des poids, mais la taille du vocabulaire a diminué (485,8 à 375,0). Nous supposons que c'est dû à une diminution du nombre de répétitions dans les phrases, et à un modèle décrivant mieux les événements sonores présents plutôt que répétant le même événement d'une manière différente. Cependant, ces résultats impliquent également que notre stratégie multitâche n'est donc utile que pour des systèmes mal régularisés, ce qui pourrait expliquer le gain obtenu par des méthodes comme [Mahfuz 2023b]. Ce type de stratégie multitâche et ses variantes est assez fréquent en DTAA, et pourrait n'être qu'un moyen différent de régulariser certains systèmes. Par la suite, nous ne conserverons donc pas le terme SER, mais bien la valeur 2 pour l'hyperparamètre λ_{wd}.

6.4 Bilan

Ce chapitre avait pour but d'explorer une méthode d'apprentissage multitâche pour améliorer la sémantique des candidats générés par notre système à l'aide de modèles de traitement de texte pré-entraînés. En premier lieu, nous avons détaillé comment le modèle pré-entraîné a été intégré à notre apprentissage pour essayer de rendre la représentation en sortie plus robuste. En particulier, nous introduisons un second terme à la fonction de coût, nommé *Sentence Embedding Regression*



(a) Valeurs de CE avec λ_{wd}=2.

(b) SB_{sim} avec λ_{wd}=2.

FIGURE 6.4 – Valeurs de CE et SB_{sim} sur l'ensemble de validation durant 100 époques d'entraînement.

(SER), une tâche de régression entre les représentations du modèle SBERT du candidat et de la référence. Nous avons détaillé comment nous avons choisi le modèle SBERT parmi ceux disponibles, puis nous avons présenté les premiers résultats et le gain obtenu par la méthode SER. Pour mieux comprendre l'impact de cette dernière, nous avons analysé les courbes d'apprentissage et observé que le terme SER semble régulariser le modèle, menant à moins de surapprentissage. Partant de ce constat, nous avons décidé d'intégrer une autre méthode de régularisation plus forte pour améliorer notre système de référence. Nous avons trouvé que l'optimiseur AdamW, avec une valeur de pénalité des poids élevée, améliore grandement les performances. Avec cette nouvelle configuration, l'ajout du terme SER n'apporte plus de gain significatif, indiquant que le précédent gain n'est dû qu'à l'effet de régularisation. Une partie des résultats de ce chapitre a été présentée lors de la conférence internationale *31st European Signal Processing Conference* en 2023 [Labbé 2023c].

Bien que cette stratégie n'ait pas mené à une amélioration avec les nouveaux hyperparamètres, elle pourrait être utilisée pour exploiter d'autres types de données que les descriptions, comme des mots-clés. En effet, comme les modèles SBERT résument l'information texte à un vecteur, il pourrait être intéressant d'essayer d'exploiter d'autres types de texte ne possédant pas nécessairement une syntaxe correcte pour améliorer la représentation du modèle en ajoutant des informations plus précises.

Dans la suite du manuscrit, nous nous refocaliserons sur les deux jeux de données AC et CL, et nous chercherons à créer un système plus généraliste capable de produire plusieurs types de descriptions par fichier audio. Pour cela, nous utiliserons notre système de référence SR3 au lieu de SR2, décrit dans le chapitre précédent 5 à partir de la section 5.3. À noter que le système SR3 adopte une pénalité de poids élevée ($\lambda_{\text{wd}} = 2$), mais pas le terme SER.

Chapitre 7

CoNeTTE

Dans la majorité des études de DTAA, les systèmes sont testés séparément sur les sous-ensembles de test d'AC ou de CL. Même lorsque ces systèmes sont appris sur plusieurs jeux d'entraînement différents, les modèles ne se focalisent en général que sur un seul jeu de test. Peu d'études tentent de créer un système unique capable d'obtenir une performance correcte sur les deux jeux en même temps. Le principal problème semble lié aux descriptions, qui décrivent les événements assez différemment à cause du processus d'annotation de chaque jeu de test.

Dans ce chapitre, nous proposons d'étudier la différence de domaine entre les jeux AC et CL, puis nous présentons notre approche pour créer un système généraliste capable d'obtenir une performance sur plusieurs domaines en même temps. Nous montrerons comment cette différence impacte nos descriptions, et comment ce système général se situe par rapport aux meilleurs systèmes de l'état de l'art.

Sommaire

7.1 Problème

En premier lieu, nous avons décidé de reprendre notre modèle SR3 (du chapitre 5) pour le tester sur les jeux de données de test qui ne correspondent pas à leur jeu de données d'entraînement. Les résultats sont donnés dans le tableau 7.1. Nous observons assez rapidement que les systèmes qui ne sont pas entraînés sur le bon jeu de données obtiennent des scores bien moindres que ceux qui le sont. Cette différence de domaine avait déjà été observée dans une étude qui montrait que les modèles entraînés sur un jeu de données de DTAA puis sur un autre oubliaient ce qu'ils avaient appris [Berg 2021]. Dans notre cas sur AC-test, le modèle entraîné sur CL est deux fois moins performant en SPIDEr, avec 23,1% comparé à 46,6%. Sur CL-eval, la conclusion est similaire avec 14,6% de SPIDEr pour le modèle entraîné sur AC comparé à 30,1% pour celui entraîné sur CL. Cependant, nous pouvons noter que les tailles des vocabulaires produits semblent liées aux jeux de données d'entraînement, ce qui pourrait indiquer que les modèles décrivent les événements sonores différemment. Le système entraîné sur CL produit des phrases bien plus diversifiées que le système entraîné sur AC, et ceux sur les deux jeux de données de test.

Table 7.1 – Résultats du CNext-trans (SR3) pour chaque jeu de test (en %).

Test	Données d'entraînement	SD	FNS	Vocab
AC	AC	**46,6**±1,0	**63,3**±0,4	396,4±53,0
	CL	23,1±0,7	52,1±0,5	**525,2**±29,2
CL	AC	14,6±0,8	46,5±0,6	401,4±49,9
	CL	**30,1**±0,5	**51,6**±0,3	**628,2**±46,2

Pour créer un système de DTAA plus généraliste sur AC et CL, nous avons décidé de concaténer leurs sous-ensembles d'entraînement respectifs. De plus, nous ajoutons plusieurs autres jeux de données d'entraînement : MACS (MA, décrit dans la section 1.3.1.3) et WavCaps (WC, décrit dans la section 1.3.1.4). Comme vu dans la section 5.7, WC contient une partie des sources de données d'AC et de CL, ce qui nous a poussé à exclure les recouvrements afin d'éviter tout biais. En outre, nous avons supprimé les fichiers audio d'une durée supérieure à 30 secondes dans WC, car cela évite certains fichiers très longs qui augmentent la durée d'entraînement. Les 30 secondes correspondent à la durée maximale des fichiers audio de CL. Dans la suite de ce chapitre, le jeu de données WC filtré est noté WC**. Pour MA, la seule modification à appliquer et le sous-échantillonnage des fichiers audio de 48 kHz à 32 kHz pour correspondre à l'entrée du CNext. La concaténation de tous les jeux de données de DTAA (AC+CL+MA+WC**) a permis d'obtenir un total de 316 122 fichiers audio d'entraînement distincts.

Dans la littérature, la plupart des approches [Yuan 2021, Kouzelis 2022, Mei 2023] exploitant plusieurs jeux de données d'entraînement divisent l'apprentissage en deux étapes : la première utilisant tous les jeux de données disponibles et la seconde se spécialisant sur un jeu de données en particulier. Dans notre cas, nous avons décidé de choisir une approche différente, en équilibrant les données en fonction du jeu cible (AC ou CL). Par exemple, pour CL, nous récupérons les 3 840 fichiers de CL puis nous sélectionnons au hasard 3 840 autres fichiers dans les autres ensembles de données, ce qui donne 7 680 fichiers audio d'entraînement vus par époque. Pour AC, nous

avons utilisé ses 46 213 fichiers et un nombre égal d'autres fichiers issus des autres jeux pour nous entraîner avec 92 426 fichiers par époque. Cet équilibrage permet de créer un système orienté vers un jeu en particulier, pour éviter que le modèle ne voit trop souvent certaines données issues d'AC ou WC par rapport à celle de CL ou MA. De plus, le jeu de données cible détermine également le sous-ensemble qui sera utilisé pour la validation durant l'apprentissage, ainsi que les valeurs de certains hyperparamètres que nous donnons dans un tableau en annexe A.1.

Les résultats des deux systèmes entraînés sur les quatre jeux de données sont reportés dans le tableau 7.2. Malgré l'ajout d'un nombre considérable de données supplémentaires, les systèmes entraînés avec des données externes ne sont pas supérieurs à ceux entraînés sur chaque jeu de données, avec une légère diminution de 1% pour AC et de 0,6% pour CL. Cependant, ces systèmes sont plus performants en moyenne sur les autres jeux de données sur lesquels ils ne sont pas spécialisés. Par exemple, le système équilibré pour CL sur AC+CL+MA+WC** obtient 36,4% de SPIDEr et 59,1% de FENSE sur AC, alors que le système entraîné que sur CL obtient 23,1% de SPIDEr et 52,1% de FENSE. Comme les systèmes de DTAA sont aussi des modèles de langage, et que la diversité, le style d'écriture et de détails diffère entre AC et CL, il semble légitime de penser que le modèle ne peut obtenir la même performance que les modèles entraînés séparément sur les deux jeux de données en même temps sans surapprendre certaines caractéristiques de l'audio. Nous pouvons alors chercher un moyen de permettre au modèle de s'adapter au style de chaque jeu, en considérant qu'elles correspondent à différentes tâches de description.

TABLE 7.2 – Résultats du CNext-trans (SR3) avec et sans données externes pour chaque jeu de test (en %). WC** se réfère au jeu de données WC sans aucun recouvrement avec tous les sous-ensembles d'AC et CL, et sans les fichiers audio durant plus de 30s.

Test	Données d'entraînement	Données équilibrées	SD	FNS	Vocab
AC	AC	N/A	**46,6**±1,0	**63,3**±0,4	396,4±53,0
	AC+CL+MA+WC**	AC	45,6±1,0	62,7±0,4	396,8±24,4
	CL	N/A	23,1±0,7	52,1±0,5	**525,2**±29,2
	AC+CL+MA+WC**	CL	36,4±0,4	59,1±0,4	376,0±12,6
CL	AC	N/A	14,6±0,8	46,5±0,6	401,4±49,9
	AC+CL+MA+WC**	AC	19,1±0,4	48,4±0,4	472,4±21,5
	CL	N/A	**30,1**±0,5	**51,6**±0,3	**628,2**±46,2
	AC+CL+MA+WC**	CL	29,5±0,4	51,0±0,0	589,2±41,9

7.2 Solution

Pour aider le modèle à générer différents types de phrases, nous pouvons lui donner explicitement en entrée une valeur indiquant le style types de phrases voulu. Pour cela, nous utilisons une représentation de la tâche (*Task Embedding*, TE), qui sont des représentations faites pour capturer l'information des tâches que doit réaliser un modèle. Par exemple, dans le domaine de la classification, nous pouvons utiliser la TE pour chaque classe, et demander au réseau de produire une réponse binaire oui/non pour savoir si la classe demandée est présente ou non. Cela permet également de rajouter de nouvelles classes plus facilement que pour un réseau standard

qui devrait produire la distribution de probabilité de toutes les classes prévues au départ, et peut parfois améliorer la généralisation du modèle. En génération de séquence, cette approche est assez employée pour unifier plusieurs tâches, comme pour le modèle de reconnaissance de la parole Whisper [Radford 2023]. Ce dernier se sert de plusieurs TE en même temps pour identifier la langue parlée, mais aussi pour définir le résultat attendu (transcription ou traduction de la parole). Dans la littérature de DTAA, une soumission faite pour la compétition DCASE en 2023 [Kadlčík 2023] a eu une idée similaire, en donnant une tâche pour entraîner leur modèle sur AC, CL et AudioSet (AS). Comme AS ne contient pas que des classes d'événements sonores, ils utilisent la concaténation des noms des classes comme description de chaque fichier audio.

7.3 Résultats

7.3.1 Amélioration des résultats avec le *Task Embedding*

Pour exploiter des TE, nous modifions la couche de représentation de l'entrée de notre décodeur, en créant des mots spéciaux qui viennent remplacer le `<bos>`. Pour chaque jeu de données cible, nous créons un vecteur appris qui sera utilisé pour l'initialisation de chaque phrase. Le modèle peut donc apprendre à reconnaître le style de phrase attendu en fonction du jeu de données, et peut ainsi générer plusieurs phrases différentes. Nous obtenons une TE pour les jeux de données AC, CL et MA, et quatre TE pour chaque source du jeu de données WC, car nous avons observé une différence dans le style des phrases selon la source des fichiers audio. Notre modèle CNext-trans entraîné sur les quatre jeux de données et utilisant les TE est nommé CoNeTTE, pour ***Co**nv**Ne**Xt-**T**ransformer with **T**ask **E**mbedding*. Par rapport au SR3, CoNeTTE ne contient que quelques paramètres supplémentaires correspondant aux vecteurs appris des tâches, et aux nouveaux mots des différents jeux de données. Le modèle passe donc de 40,2 à 48,0 millions de paramètres, mais le nombre d'opérations reste assez similaire (20,5 milliards de MACs).

Table 7.3 – Résultats des modèles CNext-trans et du modèle CoNeTTE (en %). WC** se réfère au jeu de données WC sans aucun recouvrement avec tous les sous-ensembles d'AC et CL, et sans les fichiers audio durant plus de 30s.

Test	Données d'entraînement	Données équilibrées	TE	SD	FNS	Vocab
AC	AC	N/A	Non	46,6±1,0	**63,3±0,4**	396,4±53,0
	AC+CL+MA+WC**	AC	Non	45,6±1,0	62,7±0,4	396,8±24,4
	AC+CL+MA+WC**	AC	Oui	**46,8±0,7**	63,0±0,4	379,8±17,1
	CL	N/A	Non	23,1±0,7	52,1±0,5	**525,2±29,2**
	AC+CL+MA+WC**	CL	Non	36,4±0,4	59,1±0,4	376,0±12,6
	AC+CL+MA+WC**	CL	Oui	44,1±1,1	60,9±0,3	331,2±24,0
CL	AC	N/A	Non	14,6±0,8	46,5±0,6	401,4±49,9
	AC+CL+MA+WC**	AC	Non	19,1±0,4	48,4±0,4	472,4±21,5
	AC+CL+MA+WC**	AC	Oui	25,2±1,0	49,8±0,7	467,4±37,9
	CL	N/A	Non	30,1±0,5	51,6±0,3	628,2±46,2
	AC+CL+MA+WC**	CL	Non	29,5±0,4	51,0±0,0	589,2±41,9
	AC+CL+MA+WC**	CL	Oui	**30,5±0,6**	**51,7±0,2**	**639,0±39,5**

Les résultats sont donnés dans le tableau 7.3. Lorsque nous avons introduit la TE, les résultats du CNext-trans se sont légèrement améliorés sur l'ensemble de données équilibré. Plus particulièrement, lorsque nous examinons les performances sur les ensembles de données non équilibrés, les performances ont montré une amélioration significative avec le TE, ce qui indique que le modèle a fait une meilleure utilisation des données externes. En conséquence, la meilleure performance globale du modèle a été obtenue en utilisant CoNeTTE entraîné avec les données équilibrées de CL. Il s'agit également de notre meilleur système sur CL, qui obtient 30,5% de SPIDEr, 51,7% de FENSE et avec la taille de vocabulaire la plus élevée (639,0). La version de CoNeTTE utilisant les données équilibrées d'AC obtient le meilleur score en SPIDEr sur AC avec 46,8%, mais n'est pas le meilleur score en FENSE. Ce même système obtient une performance correcte de 25,2% de SPIDEr et 49,8% de FENSE sur CL, ce qui est bien supérieur aux systèmes entraînés sans TE.

7.3.2 Représentation de tâche : changement de la forme ou du contenu ?

Si l'introduction de la TE améliore globalement le système, il est légitime de s'interroger sur l'impact exact que ce mécanisme a sur les descriptions produites. Si le style d'écriture des phrases est sans aucun doute différent entre les captions d'AC et de CL, la distribution et la nature des événements sonores l'est probablement aussi. Cela signifie que l'usage d'une TE peut avoir un impact à la fois sur le style d'écriture et sur le type d'événements sonores reconnus. Pour étudier ce phénomène, nous avons demandé au modèle CoNeTTE équilibré sur CL de générer des phrases sur AC et CL, mais avec des TE différentes du jeu de test.

Les tableaux 7.4 et 7.5 donnent des exemples de sorties de notre modèle avec ces deux TE différentes sur deux fichiers audio [43] [44]. Ces exemples démontrent que le modèle génère différentes descriptions en fonction de la tâche spécifiée par le TE. Dans le premier tableau, les mots utilisés sont relativement similaires, mais nous pouvons noter que la tâche AC a forcé le mot *« vehicle »* qui est bien présent dans les références. Dans le second tableau, l'événement sonore n'est pas du tout décrit de la même manière. La tâche CL force plus de détails, qui correspondent plus à la manière de décrire ce type de son dans les références.

Table 7.4 – Candidats générés selon deux TE pour le fichier « `35b9BSmN5JM` », issu d' d'AC-test.

N°	Tâche	Candidats	SD	FNS
C1	AC	*A vehicle engine idling and revving*	**28,3**	**59,2**
C2	CL	*An engine is idling and revving up and down*	24,9	57,7
N°	**Références**			
R1	*Loud vibrating followed by revving*			
R2	*Truck in idle mode, door closing, engine revving and accelerating*			
R3	*A wooden thud as an idle car engine runs then accelerates*			
R4	*A motor vehicle accelerates and revs*			
R5	*An engine running*			

43. https://www.youtube.com/embed/35b9BSmN5JM?start=1&end=11
44. https://freesound.org/people/stefansmulovitz/sounds/263656

Table 7.5 – Candidats générés selon deux TE pour le fichier « `Diving Bell 1.wav` », issu de CL-eval.

N°	Tâche	Candidats	SD	FNS
C1	AC	*A musical instrument is playing a note*	3,7	32,5
C2	CL	*A gong is struck and echoes in a steady rhythm*	**19,8**	**58,8**
N°	**Références**			
R1	*A bell is struck by a mallet, and the noise resonates for some time.*			
R2	*A heavy chime is struck and rings loudly at an even tone.*			
R3	*A mallet strikes a bell and the sound resonates for a time.*			
R4	*A single bell is sounded and its reverberations felt all around.*			
R5	*Single bell sound followed by its vibration sound*			

Pour voir plus globalement l'impact de la TE, dans le tableau 7.6, nous indiquons le SPIDEr, le FENSE, la taille du vocabulaire et la longueur moyenne des phrases pour les deux TE et les deux sous-ensembles de test. Par exemple, sur CL-eval, la longueur des phrases est passée de 10,8 à 7,2, et la taille du vocabulaire a chuté de 639,0 à 412,2 mots lors de l'utilisation de la TE dAC par rapport à la TE de CL. La forme des phrases semble donc avoir été fortement impactée par la TE donnée en entrée du modèle. Comme on pouvait s'y attendre, le score SPIDEr diminue fortement de 30,5 à 22,7 sur CL-eval, ce qui semble indiquer que le modèle produit des phrases assez différentes. Cependant, lorsque que nous avons calculé le SPIDEr entre les deux sorties sur CL-eval, nous avons un score élevé de 114,8%, ce qui indique en vérité que les phrases contiennent toujours un contenu similaire.

Table 7.6 – Résultats du modèle CoNeTTE sur AC et CL, avec deux différentes TE.

Test	Tâche	SD	FNS	Vocab	#Sent
AC	AC	**44,1**±1,1	**60,9**±0,3	331,2±24,0	7,5±0,2
	CL	30,7±0,8	57,6±0,4	**517,2**±35,4	**10,8**±0,1
CL	AC	22,7±0,6	49,3±0,2	412,2±26,3	7,2±0,2
	CL	**30,5**±0,6	**51,7**±0,2	**639,0**±39,5	**10,8**±0,2

Nous avons également reporté dans le tableau 7.7 les différentes proportions de mots présentes dans les vocabulaires d'entraînement, pour savoir si les TE modifient les mots employés. Nous pouvons assez facilement voir que chaque TE force bien le modèle à produire des mots qui correspondent plutôt au jeu de données de la TE en question, et ce quel que soit le jeu de test. Par exemple sur CL-eval, l'utilisation de la TE d'AC produit des phrases dont 97,9% des mots sont issus de AC-train, alors que 94,5% des mots proviennent de CL-dev.

Afin d'explorer plus amplement la différence entre les différentes TE, nous avons reporté en annexe H les détails des distributions des 15 premiers unigrammes racinisés, des trigrammes racinisés et des natures des mots dans les candidats et les références d'AC et CL. Nous pouvons observer que le nombre de prépositions et de conjonctions était respectivement deux fois plus

TABLE 7.7 – Proportions de vocabulaire des candidats générés en fonction du jeu de données d'entraînement (en %).

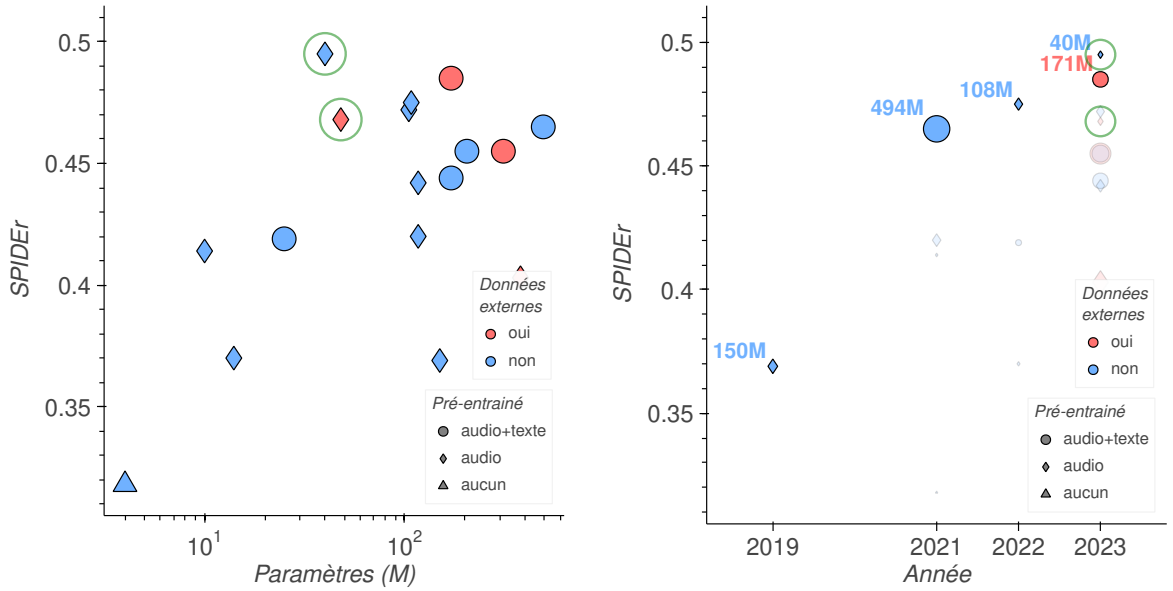
Test	Phrases	Tâche	Mots issus de	
			AC-train	**CL-dev**
AC	Candidats	AC	**99,9**±0,2	98,2±0,2
	Candidats	CL	98,1±0,4	**99,6**±0,4
	Références	N/A	92,2	84,0
CL	Candidats	AC	**97,9**±0,5	99,6±0,3
	Candidats	CL	94,5±0,8	**100,0**±0,0
	Références	N/A	71,8	100,0

élevé sur AC-test et CL-eval lors de l'utilisation de la TE de CL par rapport à la TE d'AC. Cela semble à nouveau suggérer que les descriptions adoptent différentes formulations. Les TE semblent également pousser certains trigrammes spécifiques comme *« followed by »* pour la TE d'AC, et *« in the background »*, pour celle de CL. En outre, les trigrammes correspondant à des types d'événements sonores ne sont pas toujours dans le même ordre lorsque nous utilisons des TE différentes sur le même test, ce qui semble indiquer que les TE ont également un impact sur la reconnaissance de ces événements. Par exemple, le trigramme *« bird-are-chirp »* est présent sur AC avec la TE de CL en 9^{e} position, alors qu'il n'est même pas présent dans le top 15 lorsque nous nous servons de la TE d'AC. Pour se faire une idée des événements sonores qui diffèrent entre AC et CL, nous avons donné en annexe les figures G.1 représentant les distributions des dix classes d'événements sonores les plus fréquentes selon les vraies annotations pour AC et selon le modèle CNext (entraîné sur AS$^{\text{-AC}}$) pour CL. Nous pouvons constater qu'AC contient beaucoup plus de parole que CL, alors que ce dernier contient plus de sons d'oiseaux et de pluie. Malgré tout, les événements sonores reconnus selon chaque TE semblent être assez similaires sur un jeu de test, mais une étude plus approfondie serait nécessaire pour le confirmer.

7.3.3 Efficacité du système

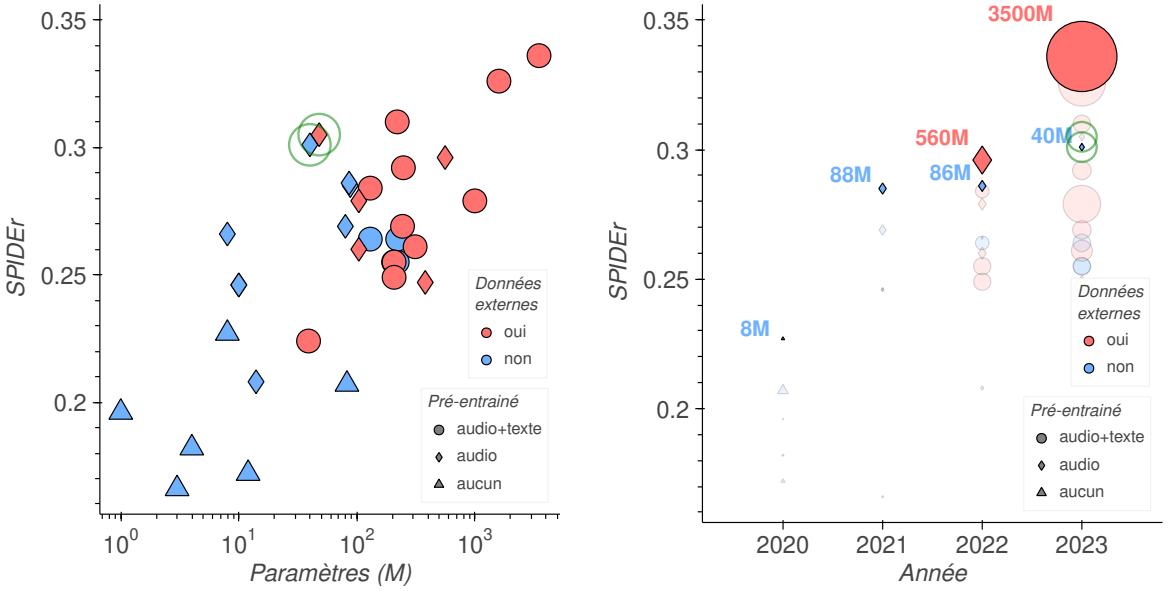
Le modèle CoNeTTE est le système générique le plus efficace et polyvalent pour la DTAA que nous ayons conçu. Le système ne peut pas être directement comparé à l'état de l'art sur AC, notamment car les autres systèmes sont possiblement biaisés (c.f. chapitre 5). Il se place tout de même plutôt bien en quatrième position avec 46,8% de SPIDEr et reste 3 à 5 fois plus petit que les modèles plus performants. De plus, il se positionne assez bien comparé à l'état de l'art sur CL, avec 30,5% de SPIDEr alors que l'état de l'art sans méthode d'ensemble de modèles qui atteint 32,6%, avec 40 fois plus de paramètres. Nous avons donné dans les figures 7.1 et 7.2 la place de CoNeTTE ainsi que notre modèle de référence SR3 biaisé et entraîné sur les jeux de données séparés.

Nous observons assez rapidement que notre modèle biaisé SR3 est à la fois plus petit et plus performant que la plupart des systèmes. CoNeTTE est également efficace et reste plus petit que la majorité des systèmes qui obtiennent des performances similaires, alors que ces derniers sont possiblement biaisés sur AC. Sur CL, le modèle est plus léger que ceux de la littérature avec seulement 42 millions de paramètres. Il reste cependant de la marge pour l'améliorer, notamment



(a) SPIDEr par nombre de paramètres sur AC. (b) SPIDEr par année sur AC.

FIGURE 7.1 – État de l'art de DTAA sur AC, par nombre de paramètres et année. Nos systèmes CoNeTTE et SR3 sont entourés en **vert**.



(a) SPIDEr par nombre de paramètres sur CL. (b) SPIDEr par année sur CL.

FIGURE 7.2 – État de l'art de DTAA sur CL, par nombre de paramètres et année. Nos systèmes CoNeTTE et SR3 sont entourés en **vert**.

en essayant d'exploiter des modèles de texte pré-entraînés comme le font la plupart des autres modèles. Il serait intéressant d'étudier ces modèles pour savoir si le potentiel gain obtenu vaut l'introduction de centaines de millions de paramètres. Dans les deux cas, le système SR3 fait partie des optimums de Pareto de ces graphiques. Et le système CoNeTTE est le seul système

capable de générer des phrases dans les deux styles de descriptions, avec le meilleur score moyen sur les deux jeux de données.

7.4 Démo

Une interface simple pour tester CoNeTTE est disponible sur HuggingFace Spaces [45]. Elle permet à un utilisateur de téléverser des fichiers audio, ou bien d'enregistrer un fichier audio depuis le micro et de l'envoyer en entrée du modèle. Une illustration de cette interface est donnée dans la figure 7.3. Les fichiers audio en entrée sont automatiquement ré-échantillonnés à 32 kHz. Pour cette démo gratuite, le modèle fonctionnne sur deux CPU seulement, ce qui explique le temps de réponse variable et parfois élevé pour certains fichiers audio. Il est également possible de sélectionner certains hyperparamètres liés à la génération de la phrase : la représentation de la tâche (AC, CL, MA, WC-FSD, WC-SB, WC-AS, WC-BBC), le fait d'autoriser certaines répétitions (*stopwords*, aucune, toutes), la taille du faisceau et le nombre minimal et maximal de mot par phrase. Il est également possible de tester le modèle CoNeTTE localement à travers un terminal ou du code Python grâce à la bibliothèque `conette` [46].

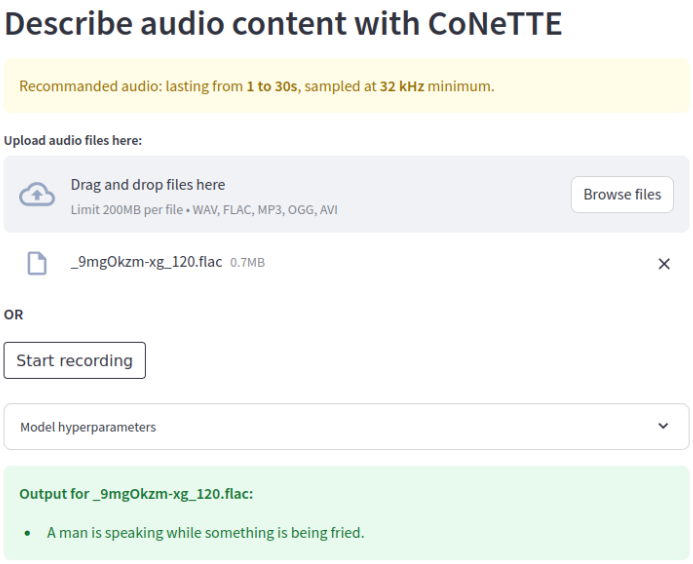


FIGURE 7.3 – Capture d'écran de l'interface permettant de tester CoNeTTE.

7.5 Bilan

Dans ce chapitre, nous avons présenté CoNeTTE, notre modèle de DTAA généraliste visant à obtenir une bonne performance pour tous les jeux de données. En premier lieu, nous avons observé que les modèles entraînés sur AC ou CL n'obtiennent pas une performance acceptable sur l'autre jeu de données. Ce problème peut être atténué en concaténant divers jeux de données

45. https://huggingface.co/spaces/Labbeti/conette
46. https://github.com/Labbeti/conette-audio-captioning

d'entraînement et en équilibrant un jeu de données, au détriment d'une légère baisse de SPIDEr et FENSE sur le jeu de données équilibré.

Pour remédier à cette difficulté, nous avons proposé de voir le problème sous l'angle de plusieurs tâches de génération, en introduisant une représentation de chaque jeu de données. L'intégration de cette tâche nous permet d'obtenir un gain de performance global de notre système, surpassant largement les systèmes entraînés séparément en moyenne. Cependant, l'équilibrage des fichiers d'entraînement ainsi que le choix du jeu de validation ont un impact important selon le jeu de données cible, ce qui signifie qu'il reste à créer une procédure plus généraliste qui ne dépendrait pas d'un jeu cible. Pour analyser l'effet de ces tâches, nous avons expérimenté et comparé les sorties pour les mêmes jeux de données selon deux tâches. Nous avons constaté que le système produisait différentes phrases d'après le vocabulaire, la longueur, la nature des mots et les n-grammes. Cela implique que les tâches affectent bien le style des descriptions générées. Cependant, la distribution de certains n-grammes correspondant à des événements sonores semble indiquer que cela affecte également la reconnaissance des événements sonores dans les candidats, ce qui pourrait indiquer que le modèle hallucine certains événements.

Le travail de chapitre a donné lieu à un article soumis au journal *IEEE/ACM Transactions Audio Speech and Language Processing*. Une version de prépublication est actuellement disponible sur arXiv [Labbé 2023a]. Pour aller plus loin, il serait pertinent de continuer à rendre l'apprentissage moins dépendant d'un jeu de données cible, pour obtenir la meilleure performance sur tous les sous ensembles de test. De plus, il serait pertinent d'étudier les nombreux autres jeux de données décrit dans la section 1.3.2, qui pourraient proposer d'autres styles de descriptions.

À présent que nous avons un système robuste pour la DTAA, nous allons analyser d'autres cas d'utilisations liés à cette tâche. En premier lieu, nous étudierons la tâche de Recherche Audio-Texte (RAT), qui consiste à récupérer un fichier audio ou texte à partir d'une entrée de l'autre modalité. Nous montrons dans le chapitre 8 comment notre système peut être utilisé pour réaliser cette tâche sans entraînement supplémentaire, et comment il se place par rapport aux systèmes dédiés. Puis, nous étudions d'autres paradigmes d'apprentissage différents de l'apprentissage supervisé pour tenter d'exploiter un ensemble de données non annoté pour améliorer notre modèle. Nous nous focalisons dans le chapitre 9 sur les apprentissages dits « semi-supervisés » (*Semi-Supervised Learning*, SSL). En premier lieu, nous conduisons une étude de certaines méthodes SSL pour une tâche simple de classification audio sur plusieurs jeux de données, puis nous tentons de nous servir de l'une de ces méthodes pour la DTAA. Enfin, nous explorons la tâche de DTAA pour des langues différentes de l'anglais dans le dernier chapitre 10. Nous proposons d'abord des systèmes dédiés à chaque langue et entraînés sur des données traduites automatiquement de l'anglais. Puis, nous évaluerons l'un de ces systèmes sur des données françaises d'un jeu ré-annoté manuellement, pour finir par créer un système unique capable de générer des descriptions dans plusieurs langues.

Troisième partie

Ouverture

Chapitre 8

Recherche Audio-Texte

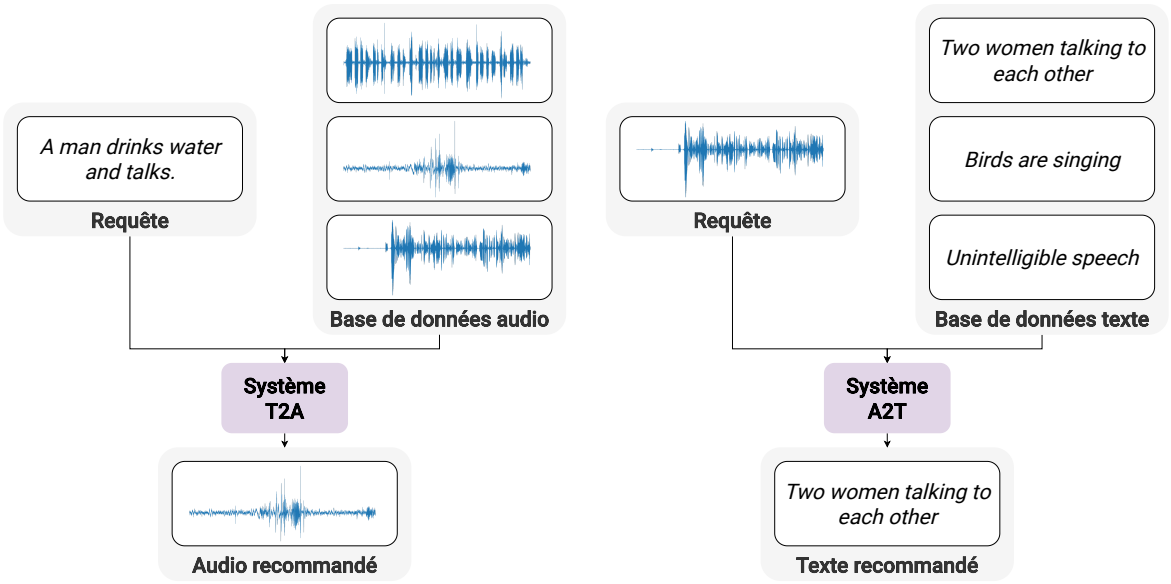
La Recherche Audio-Texte (*Audio-Text Retrieval*, RAT) est une tâche partageant de nombreux points communs avec la DTAA. Elle consiste à concevoir un système capable de récupérer un fichier audio (respectivement texte) dans sa base de données à partir d'une requête exprimée via la modalité texte (respectivement audio). Les systèmes doivent donc apprendre à fusionner les informations audio et texte comme pour la DTAA, et s'entraînent sur des jeux de données similaires. Cependant, les architectures employées sont très différentes, et peu d'études semblent essayer de rapprocher ces deux tâches qui sont pourtant liées.

Dans ce chapitre, nous proposons une méthode utilisant notre système de DTAA pour réaliser la RAT, sans apprentissage supplémentaire. Nous montrons que notre méthode est compétitive avec d'autres modèles de la littérature pourtant dédiés à cette tâche, alors que notre système n'a pas été conçu pour la réaliser. De plus, nous proposons une correction visant à tempérer un biais lié au modèle de langage de notre système, qui surestimait certaines phrases. Et enfin, nous étudions la capacité de notre modèle à distinguer certaines relations entre les événements sonores décrits comme la séquence ou la superposition, au travers de plusieurs expériences visant à modifier les mots indiquant ces relations dans les phrases.

Sommaire

8.1 Contexte

La recherche audio-texte vise à créer des systèmes capables de rechercher dans une base de données un élément à partir d'une entrée. Dans le domaine de la recommandation automatique, cette entrée est nommée « requête », la vérité terrain est dite « pertinente » et l'élément choisi par le modèle est nommé élément « recommandé ». Nous parlerons de recherche texte-vers-audio (*Text-to-Audio Retrieval*, T2A) lorsque la requête est exprimée sous forme de texte et la base de données contient des fichiers audio. Puis, nous nommerons audio-vers-texte (*Audio-to-Text Retrieval*, A2T) lorsque les modalités sont inversées. Les deux sous-tâches de T2A et A2T sont respectivement illustrées par les figures 8.1a et 8.1b. Pour les réaliser, les approches emploient des architectures composées de deux encodeurs, chacun dédié à une modalité. Les deux représentations sont ensuite comparées ou servent d'entrée à un autre réseau pour calculer un score de similarité ou de dissimilarité. Ces tâches demandent donc des paires audio-texte positives, mais peuvent aussi exploiter des paires négatives, où l'audio et le texte ne correspondent pas.



(a) Tâche Texte-vers-Audio (T2A).

(b) Tâche Audio-vers-Texte (A2T).

FIGURE 8.1 – Illustration des deux tâches de recherche audio-texte.

Pour évaluer ce type de système, nous adoptons des métriques très différentes de celles appliquées en DTAA. La principale métrique est le Rappel à k (*Recall at k*, R@k), qui est employé pour mesurer la performance d'un système de recommandation. Pour chaque requête, le système classe tous les éléments de la base de données, et la métrique vaut 1 si au moins l'un des éléments du top-k est pertinent, ou zéro dans le cas contraire, comme décrit par l'équation 8.1. La valeur de k choisie fait donc fortement varier le score obtenu, et dépend de l'application que nous cherchons à réaliser. Pour la RAT, les valeurs utilisées sont 1, 5 et 10, et nous noterons respectivement ces métriques R@1, R@5 et R@10. Une autre métrique employée pour évaluer les systèmes de la compétition 6b de DCASE est la précision moyenne à k (*mean Average Precision at k*, mAP@k). Contrairement aux métriques précédentes, elle prend en compte la position de l'élément pertinent dans le top-k, en calculant l'inverse de son rang. Par exemple, si l'élément pertinent est en position 3 dans le top-10, le score mAP@10 sera 1/3. Si plusieurs éléments

pertinents peuvent être récupérés, le score sera la moyenne des inverses des rangs, multipliés par l'indice de l'élément pertinent. D'autres métriques comme la moyenne ou la médiane normalisée du rang existent, mais elles ne semblent pas être utilisées en RAT.

$$\text{R@k} = \begin{cases} 1 & \text{si au moins un élément pertinent est dans le top-k} \\ 0 & \text{sinon} \end{cases} \tag{8.1}$$

Comme nous l'avons énoncé précédemment, les architectures de RAT emploient deux encodeurs pour créer un système réalisant les deux tâches de T2A et d'A2T. Dans la littérature, la totalité des études emploie un encodeur pré-entraîné pour l'audio comme ceux de PANN [Wang 2023a], PaSST [Primus 2023, Pellegrini 2022] ou encore HTSAT [Mei 2023]. La partie texte est également pré-entraînée, avec bien souvent des modèles fondés sur BERT [Mei 2023, Wang 2023a, Primus 2023, Koepke 2022, Kim 2022, Pellegrini 2022]. Certaines études se focalisent sur l'exploration de différentes fonctions de coût [Mei 2022b, Xin 2023], tandis que d'autres étendent la recherche de données à plusieurs modalités pour améliorer la généralisation [Wang 2023b]. Par ailleurs, une tâche dédiée à la T2A a été proposée en 2022 et 2023 au challenge DCASE et nommée *Languaged-Based Audio Retrieval*[47] (tâche 6b).

Comme pour la DTAA, les deux tâches de recherche audio-texte cherchent à fusionner des informations sur les événements sonores présents dans un fichier audio ou décrits par une phrase. Comme les architectures peuvent donner un score pour n'importe quelle paire audio-texte, les modèles peuvent tout autant réaliser la sous-tâche T2A que A2T. Elles peuvent également s'entraîner sur des jeux de données similaires, même si ceux de DTAA ne contiennent pas de paires négatives. Cependant, bien que les tâches de DTAA et de RAT partagent plusieurs points communs, elles sont bien souvent réalisées par des systèmes différents. En DTAA, quelques études réutilisent parfois certains modèles de T2A pour réaliser de l'apprentissage par transfert [Xu 2022], ou d'autres vont parfois ajouter une fonction de coût de RAT pour améliorer la représentation audio de leur encodeur audio et la performance de leur système de DTAA [Wu 2023b]. Pourtant, nous pensons qu'un système de DTAA capture également des informations audio et texte, ce qui pourrait lui permettre de réaliser des tâches de recherches.

8.2 Utilisation d'un système de DTAA pour la RAT

La première idée pour exploiter un système de DTAA pour la RAT est de générer des candidats pour décrire chaque fichier audio. Chaque candidat est utilisé comme résumé et comparé aux requêtes textuelles ou base de données de texte. Une illustration de ce procédé pour la T2A est donnée par la figure 8.2.

Nous pouvons adopter une métrique comme BLEU, CIDEr-D ou SB_{sim} pour comparer les textes, comme proposé dans [Krishnan 2021], et pré-calculer tous les candidats de la base de données audio, ce qui accélère la recherche par rapport à un système de RAT pur. Cependant, nous avons trouvé de faibles résultats en utilisant cette stratégie, bien en dessous de la plupart des systèmes de la littérature (<3% de R@1 en T2A sur CL). Nous pensons que les systèmes de DTAA ont tendance à produire des phrases moins détaillées et diversifiées que les références humaines, ce qui entraîne une perte d'information lorsqu'ils sont exploités pour résumer le contenu

47. https://dcase.community/challenge2022/task-language-based-audio-retrieval

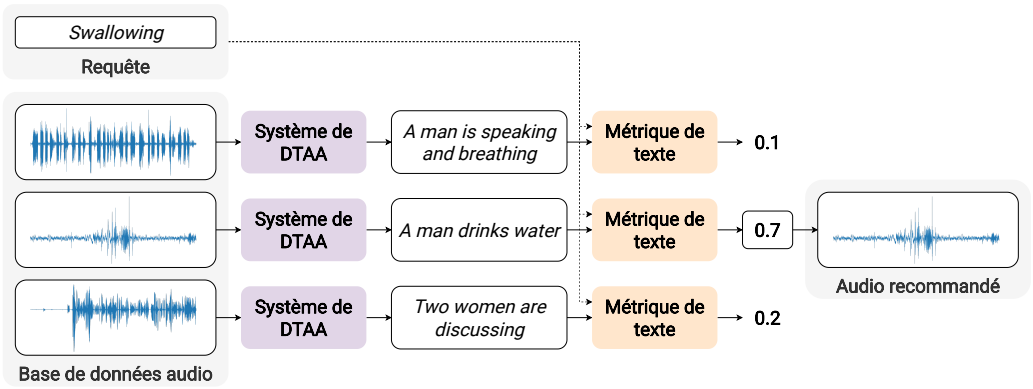


FIGURE 8.2 – Illustration de la tâche T2A par les candidats d'un système de DTAA.

audio en une seule phrase. Typiquement, la taille du vocabulaire généré dans les candidats est égale à 628 mots distincts par rapport aux 1839 mots présents en moyenne dans les références sur CL-eval. Pour mieux explorer cette approche, il serait donc nécessaire de développer des systèmes de DTAA produisant beaucoup plus de détails, en générant plusieurs phrases par audio par exemple.

Une autre approche totalement différente consiste à tirer avantage de la fonction de coût du modèle. Les systèmes de DTAA sont entraînés à prédire le prochain mot d'une phrase en utilisant les mots précédents et le fichier audio. Cela signifie que le modèle prend en entrée un fichier audio et une description, et donc que la CE peut être exploitée pour évaluer une paire. Ainsi, nous nous attendons à ce qu'un système de DTAA soit capable de donner une valeur plus faible pour une paire audio-texte positive que pour une paire négative. La figure 8.3 illustre la manière dont le système de DTAA choisit la meilleure phrase pour un audio particulier et pour la tâche de T2A.

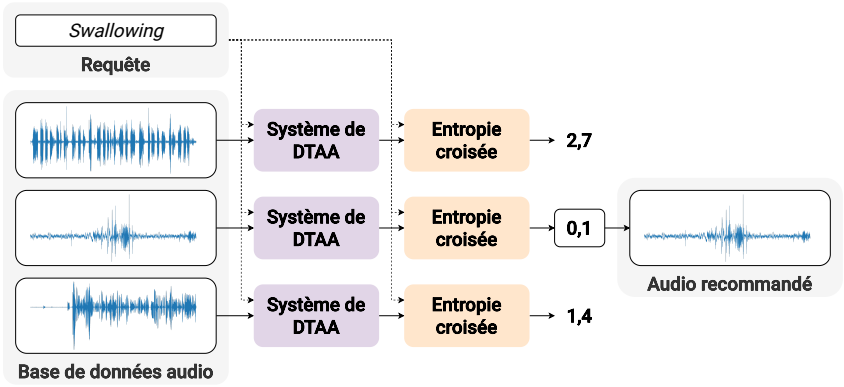


FIGURE 8.3 – Illustration de la tâche T2A par la CE d'un système de DTAA.

Plus précisément, les équations 8.2a et 8.2b décrivent comment les éléments recommandés sont choisis pour les tâches de RAT :

$$\mathrm{T2A}(t, A, f) = \underset{a \in A}{\operatorname{argmin}} \frac{1}{|t| - 1} \sum_{i=1}^{|t|-1} \mathrm{CE}(f(a, t_{1:i}), t_{i+1}) \tag{8.2a}$$

$$\text{A2T}(a, T, f) = \underset{t \in T}{\operatorname{argmin}} \frac{1}{|t| - 1} \sum_{i=1}^{|t|-1} \text{CE}(f(a, t_{1:i}), t_{i+1}) \tag{8.2b}$$

où t correspond à une description (avec les unités lexicales `<bos>` et `<eos>`), T est la liste de toutes les descriptions, a est un fichier audio, A la liste de tous les fichiers audio. f est le système de DTAA qui produit les distributions de probabilités pour les mots suivants étant donné les mots précédents, dans le contexte d'un fichier audio. Pour la suite de nos expériences, nous nous servirons des modèles SR3 du chapitre 5, entraînés séparément sur les jeux de données.

8.3 Résultats

Les résultats pour la T2A et l'A2T sont respectivement donnés dans les tableaux 8.1 et 8.2. Comme pour les résultats de la DTAA, nous avons reporté les méthodes de l'état de l'art sans les méthodes exploitant plusieurs modèles et en indiquant la présence des données externes.

TABLE 8.1 – Résultats de la tâche T2A sur AC et CL. Entr. est ici le diminutif pour Entraînement. Les meilleures valeurs pour chaque jeu, tâche et métrique sont indiquées en **gras**, alors que les meilleures valeurs sans données externes sont soulignées.

Test	Système	Données d'entr.	Params. (M, ↓)	R@1	R@5	R@10
AC	ONE-PEACE [Wang 2023b]	AC+CL+7 autres	2000	**0,425**	**0,775**	**0,884**
	MMT [Koepke 2022]	AC	290	0,361	0,720	0,845
	TAP-PMR [Xin 2023]	AC	185	0,368	0,727	N/A
	SR3 (nous)	AC	**40**	0,382	0,733	0,853
CL	PaSST-N^4 [Primus 2023]	AC+CL+WC	441	**0,261**	**0,553**	**0,693**
	CNN14-BERT [Wang 2023a]	CL	192	0,167	0,410	0,539
	TAP-PMR [Xin 2023]	CL	185	0,171	0,396	N/A
	SR3 (nous)	CL	**40**	0,137	0,349	0,480

T2A. Sur AC, notre système SR3 atteint l'état de l'art des modèles de RAT sans utilisation de données externes à AC avec un R@1 de 0,382 comparé au précédent meilleur système TAP-PMR qui obtenait 0,368. Notre modèle est aussi surprenamment plus petit que la plupart des autres systèmes, notamment car il n'utilise pas de modèle de texte pré-entraîné qui dépassent bien souvent les 100 millions de paramètres. Cependant, notre système reste bien inférieur à l'état de l'art avec données externes ONE-PEACE, qui obtient +4,3% absolu en R@1. Sur CL, notre modèle est bien moins performant que la plupart des autres systèmes dédiés. En effet, la tâche dédié à la T2A de DCASE motive beaucoup de participants à optimiser des modèles sur ce jeu de données, ce qui donne plus de résultats que pour AC. Notre modèle atteint 0,137 de R@1, alors que l'état de l'art sans données externes atteint 0,171 et celui avec données externes atteint 0,261. Nous pouvons également constater que notre système reste celui qui utilise le moins de paramètres.

TABLE 8.2 – Résultats de la tâche A2T sur AC et CL. Entr. est ici le diminutif pour Entraînement. Les meilleures valeurs pour chaque jeu, tâche et métrique sont indiquées en **gras**, alors que les meilleures valeurs sans données externes sont soulignées. L'astérisque * indique les résultats mis à l'échelle par une stratégie min-max décrite dans la section suivante 8.4.

Test	Système	Données d'entr.	Params. (M, ↓)	R@1	R@5	R@10
AC	HTSAT-BERT [Mei 2023]	AC+WC	141	**0,546**	**0,852**	**0,924**
	MMT [Koepke 2022]	AC	290	0,396	0,768	0,867
	Multi-TTA [Kim 2022]	AC	187	0,402	0,740	0,872
	CNN14-TAP-PMR [Xin 2023]	AC	192	0,431	0,733	N/A
	SR3 (nous)	AC	**40**	0,146	0,355	0,493
	SR3 + normalisation (nous)	AC	**40**	0,398*	0,814*	0,919*
CL	CNN14-BERT [Mei 2023]	CL+WC	214	**0,271**	**0,527**	**0,663**
	Triplet-weighted [Mei 2022b]	CL	185	0,169	0,381	0,514
	TAP-PMR [Xin 2023]	CL	185	0,182	0,399	N/A
	SR3 (nous)	CL	**40**	0,038	0,119	0,178
	SR3 + normalisation (nous)	CL	**40**	0,148*	0,404*	0,541*

A2T. D'une manière surprenante pour cette tâche, notre système SR3 est bien moins performant que les modèles de l'état de l'art sur les deux jeux de données. En effet, il atteint 0,146 de R@1 sur AC alors que les modèles s'approchent de 0,400 de R@1. Sur CL, la performance est encore plus faible avec seulement 0,038 de R@1, alors que le meilleur modèle sans données externes TAP-PMR atteint 0,182. La différence est plus importante lorsque l'on compare notre système aux états de l'art avec données externes (HTSAT-BEATs et CNN14-BERT), qui dépassent largement les autres systèmes de plus de 10% de R@1. Cette différence de performance aussi importante pour les tâches de T2A et d'A2T est assez surprenante, puisqu'aucun modèle dédié à la RAT ne semble souffrir d'une telle différence. Nous avons donc décidé d'analyser plus en détail les valeurs de notre système pour mieux comprendre pourquoi ils permettent de réaliser efficacement l'une des tâches mais pas l'autre sans la normalisation, que nous allons décrire dans la section suivante.

8.4 Pourquoi nos scores A2T sans normalisation sont-ils si faibles ?

Comme nous l'avons observé dans la section précédente, nous avons constaté que même si notre système est performant sur la tâche T2A, les résultats sur la tâche A2T sont très faibles par rapport à ceux de l'état de l'art. En observant les descriptions, nous avons constaté que cela est dû à un sous-ensemble de phrases pour lesquelles les valeurs de CE sont presque toujours inférieures à toutes les autres, et ceux pour tous les fichiers audio. En particulier, seules 120 descriptions différentes sont récupérées pour les 1045 requêtes pendant la tâche A2T avec des coûts bruts, mais nous n'avons pas trouvé de fortes corrélations entre ces descriptions et la fréquence de leurs mots ou leur longueur. Nous pouvons voir dans la figure 8.4 que les valeurs des CE de trois phrases peuvent avoir des intervalles de valeurs assez différents, et certaines phrases sont beaucoup plus vraisemblables selon le modèle que d'autres. Les trois phrases sont, de haut

en bas dans la figure : *« A person watches birds near the voluminous flowing river. »*, *« A person hammering a structure with wood and using a table saw »* et *« A metal clanging resonates in the background while a latch bangs against a hard surface. »*, toutes issues de CL-eval.

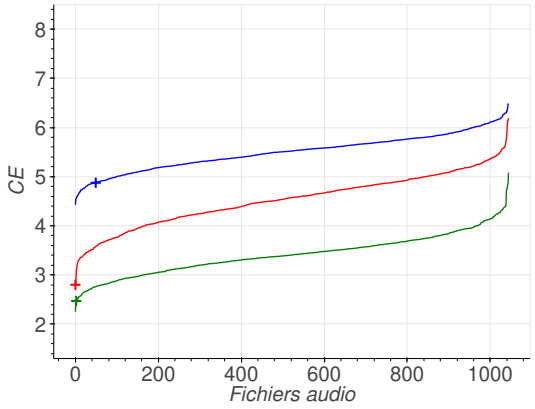
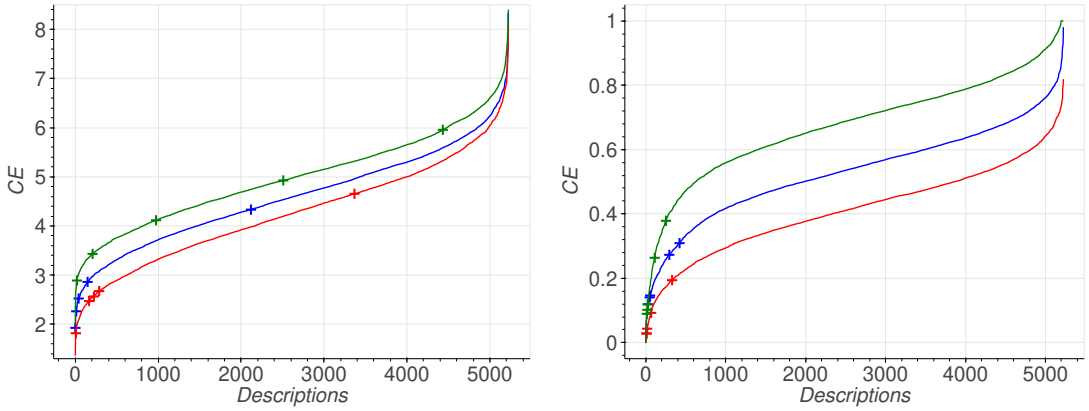


FIGURE 8.4 – Valeurs de CE ordonnées pour trois descriptions selon tous les fichiers audio.



(a) Sans normalisation min-max.

(b) Avec normalisation min-max.

FIGURE 8.5 – Valeurs de CE ordonnées pour trois audios selon toutes les descriptions, sans normalisation (gauche), avec normalisation (droite). Les positions des éléments pertinents sont indiquées par une croix.

Afin de clarifier pourquoi certains mauvais scores affectent que la tâche A2T et non T2A, nous fournissons un exemple dans le tableau 8.3. Cet exemple montre les valeurs de CE pour trois fichiers audio A_i différents avec leurs descriptions correspondantes C_i. Lorsque nous effectuons la tâche T2A, nous sélectionnons l'audio récupéré A_i avec la valeur de coût la plus faible dans la colonne i, ce qui permet d'obtenir un score parfait dans ce cas. Cependant, lorsque nous effectuons la tâche A2T, seule la description T_1 est retrouvée, car sa colonne a une plage de valeurs différente des autres, ce qui explique les résultats médiocres lorsque nous utilisons les valeurs de coût brutes. Pour résoudre ce problème, nous proposons un post-traitement qui normalise chaque « colonne » (c'est-à-dire chaque série de valeurs correspondant à une seule description). En particulier, nous avons essayé une normalisation ℓ_1, normalisation ℓ_2, de standardiser, mais une simple normalisation min-max a donné les meilleurs résultats. Les valeurs normalisées

de l'exemple sont données dans les trois dernières lignes du tableau 8.3. L'impact de cette normalisation sur les coûts pour l'A2T est présenté dans les figures 8.5a et 8.5b. Les trois fichiers audio sont, de haut en bas, « `interference from wireless mouse on am radio.wav` » [48] (**en vert**), « `Deutz-Tractor-Engine-1972.wav` » [49] (**en bleu**) et « `20090712.engine.01.wav` » [50] (**en rouge**) issus de CL-eval. Nous avons également ajouté une règle pour résoudre le problème d'indéterminisme lors de la recherche, qui apparait lorsque deux descriptions récupérées ont le même score minimal (zéro lorsqu'elles sont la valeur minimale de leur colonne). Dans ce cas, nous utilisons les valeurs de coût d'origine pour décider quel élément sera choisi. Les scores finaux d'A2T par normalisation sont donnés dans le tableau 8.2. On obtient un gain de performance importante pour les deux jeux de données, avec +25,2% sur AC et +11,0% sur CL de R@1.

TABLE 8.3 – Valeurs de CE pour trois différents exemples audio et texte. Les valeurs en **gras** et soulignées correspondent respectivement à la meilleure valeur pour la T2A et l'A2T.

Mise à l'échelle		T_1	T_2	T_3
Non	A_1	**1,7**	3,4	8,1
	A_2	2,2	**2,6**	8,5
	A_3	2,0	3,3	**6,5**
Oui	A_1	**0,0**	1,0	0,8
	A_2	1,0	**0,0**	1,0
	A_3	0,6	0,9	**0,0**

8.5 Avantages et inconvénients

Récemment, un article a montré que les systèmes de RAT ne parviennent généralement pas à capturer les relations de haut niveau entre les sons [Wu 2023a]. Plus précisément, les auteurs proposent de remplacer dans les phrases le mot *« after »* par *« before »* et vice versa pour inverser la séquence d'événements sonores décrite et nomment cette modification le Test-Avant-Après (*Before-After Test*, BAT). Un bon système de RAT devrait être capable d'attribuer un score inférieur à une description incorrecte plutôt qu'à une description correcte. Nous pensons que les systèmes de DTAA ou de RAT devraient être capables de capturer ce type de relation en plus des événements sonores eux-mêmes, mais les métriques utilisées ne semblent pas refléter les performances du modèle pour ce type d'analyse. En plus de la perturbation proposée par les auteurs du BAT, nous avons proposé de faire passer le type de la relation de séquence à superposition et réciproquement en remplaçant certains mots ou en inversant l'ordre des mots des phrases. Par exemple, la phrase *« a man speaks **then** a dog barks »* peut devenir *« a man speaks **while** a dog barks »* si on remplace *« then »*, ou devenir *« a dog barks **then** a man speaks »* si on inverse les propositions autour de *« then »*. Nous avons détaillé les différents mots testés dans le tableau 8.4. On nommera donc respectivement `seq2sup` (respectivement `sup2seq`) la perturbation qui remplace une séquence par une superposition (respectivement superposition par une séquence) en substituant des mots de la phrases. La perturbation qui inverse les propositions de la phrase sera nommée `seq` si le mot central représente une séquence et `sup` s'il représente

48. https://freesound.org/people/cognitoperceptu/sounds/78895
49. https://freesound.org/people/Erdie/sounds/34372/
50. https://freesound.org/people/dobroide/sounds/75628

une superposition. La totalité des mots indiquant des relations seront remplacés pour le jeu de test de CL-eval. Les phrases qui ne contiennent pas de relations seront ignorées.

TABLE 8.4 – Mots choisis pour les différentes perturbations.

Perturbation	**Mots**	**Remplacé par**
BAT	before after	after before
`seq2sup`	followed by, and then, then, before, after	as, while
`sup2seq`	as, while	followed by, and then, then, before, after

Pour étudier la capacité du modèle à bien discriminer les phrases, nous utilisons la métrique d'exactitude sur les coûts du modèle pour la référence (phrase positive t_{pos}) et la référence modifiée (phrase négative t_{neg}), définie par l'équation 8.3.

$$\text{Exactitude}(a, t_{pos}, t_{neg}) = \mathbb{1}\left(\text{CE}\left(f(a, t_{pos}), t_{pos}\right) > \text{CE}\left(f(a, t_{neg}), t_{neg}\right)\right) \tag{8.3}$$

Le tableau 8.5 montre que notre modèle est très performant pour discriminer les relations entre les événements sonores, avec 76,8% pour le BAT, ce qui est supérieur au meilleur résultat de l'étude comparée (68,5%). Nous pouvons également constater que notre modèle est très efficace pour d'autres tests qui perturbent les relations, avec 90,6% d'exactitude lorsque nous inversons la proposition avant et après la séquence de mots. Cela pourrait indiquer que notre modèle capture efficacement les relations de séquence et de superposition. Nous avons également remarqué que notre modèle est toujours capable de détecter la description correcte pour le test d'inversion avec des mots superposés. Nous supposons que c'est dû au fait que le premier événement sonore décrit dans ces phrases est le plus facilement audible dans chaque fichier audio, ce qui permet de discriminer les deux phrases dans le cas `sup`.

Bien que notre système de DTAA arrive à obtenir une performance comparable aux systèmes de RAT, il nécessite de calculer la partie décodeur pour chaque paire audio-texte, alors que les systèmes de RAT calculent généralement des représentations séparées pour chaque modalité. Seule la similarité est calculée pour chaque paire pour ces systèmes. Pour la tâche A2T, un post-traitement est nécessaire pour atteindre une performance acceptable pour notre système, et nécessite de conserver la valeur minimale et maximale de la fonction de coût pour chaque texte (ou bien une estimation de ces valeurs). Si une nouvelle description est ajoutée à la base de données, les valeurs minimales et maximales doivent également être calculées ou estimées avec plusieurs fichiers audio. Cette normalisation devrait également être nécessaire pour les expériences de classification *« zero-shot »*, qui sont proches de la tâche A2T. Les systèmes de DTAA sont donc très coûteux pour la RAT par rapport aux systèmes dédiés, ce qui limite leur application dans ce contexte.

TABLE 8.5 – Exactitude selon différentes perturbations sur CL-eval. 50% est le score d'un modèle aléatoire.

Système	Type	Perturbation	Exactitude (%, ↑)
MLP [Wu 2023a]	Substitution	BAT	49,6
MLP+ACBA [Wu 2023a]			55,4
Trans. [Wu 2023a]			50,9
Trans.+ACBA [Wu 2023a]			68,5
SR3	Substitution	BAT	76,8
		`seq2sup`	82,5
		`sup2seq`	90,3
SR3	Inverse	BAT	89,2
		`seq`	90,6
		`sup`	77,8

8.6 Bilan

Dans ce chapitre, nous proposons une méthode afin d'exploiter n'importe quel système de DTAA standard pour réaliser de la Recherche Audio Texte (RAT). Nous utilisons l'entropie croisée du modèle pour chaque paire audio-texte comme score de dissimilarité. Les résultats montrent que même s'il n'est pas spécifiquement entraîné pour la RAT, un système de DTAA peut atteindre des performances comparables ou supérieures aux systèmes dédiés pour les sous-tâches T2A et A2T. En outre, il peut même atteindre des scores de l'état de l'art par rapport aux méthodes de RAT dédiées qui n'utilisent pas de données externes.

Cependant, nous avons également observé que notre modèle surestime souvent la valeur de la fonction de coût pour un sous-ensemble de phrases, ce qui se traduit par des résultats médiocres dans la configuration initiale pour la tâche d'A2T uniquement. Pour résoudre ce problème, nous avons introduit une stratégie de post-traitement fondée sur une normalisation min-max afin d'atténuer les biais dans les scores. Cet ajustement a permis d'améliorer les résultats de manière significative, par exemple en faisant passer le R@1 de 0,038 à 0,148 pour CL et de 0,146 à 0,398 pour AC.

Enfin, nous avons évalué notre système en perturbant les descriptions d'entrée et nous avons constaté qu'il était plus performant qu'une autre méthode de RAT pour distinguer les relations entre les événements sonores. En particulier, notre modèle atteint un score de 90,6% d'exactitude pour discriminer une phrase correcte d'une phrase décrivant les événements sonores dans l'ordre inverse. Le modèle est aussi capable de distinguer les phrases décrivant une séquence de la superposition entre des événements avec un taux d'exactitude allant jusqu'à 90,3%. Tous ces résultats ont été présenté en 2023 dans une publication acceptée au *DCASE2023 Workshop* [Labbé 2023].

À l'avenir, une direction de recherche pourrait impliquer la modification de l'entraînement de la DTAA en utilisant une fonction de coût contrastive pour améliorer les performances de la RAT. Cette fonction pourrait s'appliquer conjointement avec la CE, sur les valeurs de cette dernière ou

sur les représentations intermédiaires du modèle. Une autre piste consisterait à développer de nouvelles bases de données de test pour affiner l'évaluation des systèmes de RAT, en ajoutant des informations sémantiques sur les relations des événements sonores décrits, avec des graphes de scènes utilisés dans SPICE par exemple. Récemment, une étude [Xie 2023] a également proposé d'évaluer la capacité d'un système de DTAA à prédire la séquence d'événements sonores. Cette méthode vérifie si l'un des mots *« follow, followed, then, after »* est présent dans le candidat et la référence et ramène le problème à une classification de relations dont on calcule un F-score. Cela pourrait mener à des métriques se focalisant beaucoup plus sur ces aspects, pour mieux mesurer la capacité des modèles de DTAA ou de RAT à aller plus loin que les événements sonores.

Chapitre 9

Apprentissages semi-supervisés

Les méthodes d'apprentissage utilisant des réseaux de neurones profonds nécessitent souvent une grande quantité de données annotées pour converger, ce qui peut être coûteux, fastidieux et difficile à collecter. Pour pallier ce problème, les méthodes d'apprentissage semi-supervisées (*Semi-Supervised Learning*, SSL) exploitent deux ensembles de données pendant l'entraînement : un étiqueté et un autre non étiqueté. La collecte de données non étiquetées étant beaucoup plus facile et rapide, l'ensemble non étiqueté est beaucoup plus grand. Le but des méthodes SSL est donc d'exploiter au mieux cette large quantité de données afin d'améliorer les performances tout en limitant les coûts. Dans ce chapitre, nous introduisons le fonctionnement général des méthodes SSL, puis nous détaillerons six d'entre elles. Premièrement, nous les appliquons à une tâche de classification audio mono-étiquette plus simple que la DTAA. Nous comparons les résultats de ces méthodes avec des apprentissages supervisés sur trois jeux de données différents et nous concluons sur les gains et les limites observés. Enfin, nous décrivons en fin de chapitre nos essais d'adapter l'une de ces méthodes pour réaliser de la DTAA semi-supervisée.

Sommaire

9.1 Contexte

Dans la littérature, nous pouvons classer les types d'apprentissage automatique en fonction des étiquettes. Nous parlons d'apprentissage **supervisé** lorsque le système exploite les données d'entrées X avec leurs étiquettes Y et d'apprentissage **non supervisé** lorsque seul X est utilisé. L'apprentissage **faiblement supervisé** (*weakly supervised learning*) est un sous-ensemble de l'apprentissage supervisé, et intervient lorsque que les étiquettes Y ne sont pas totalement adaptées à la tâche que le système doit accomplir. Nous nommons les apprentissages faiblement supervisés en fonction de la nature de Y :

— **apprentissage multi-instances** (*multi-instance learning*, *inexact supervision*) lorsque chaque élément de X est associé à un ensemble d'étiquettes potentiellement ambiguës. Cela arrive souvent lorsqu'Y dépend de la subjectivité des annotateurs ;

— **apprentissage robuste aux bruits** (*robust learning*, *label noise learning*, *incorrect supervision*) lorsqu'Y est bruité ou parfois incorrect ;

— **apprentissage légèrement supervisé** (*lightly supervised learning*, *partially supervised learning*) lorsque les éléments de Y ne contiennent qu'une partie des informations nécessaires à la vraie tâche que nous cherchons à accomplir. Le terme est parfois aussi utilisé lorsqu'Y est annoté par des non-experts du domaine ;

— **apprentissage semi-supervisé** (*semi-supervised learning*, *incomplete supervision*) lorsqu'Y est incomplet, c'est-à-dire que certains éléments de X n'ont pas d'étiquette dans Y. Dans ce manuscrit, nous notons ce type d'apprentissage SSL, un acronyme qui ne doit pas être confondu avec le *self-supervised learning*.

Pour les méthodes SSL, nous notons les données étiquetées X_s, leurs étiquettes Y_s et le reste des données non étiquetées X_u. Dans ce chapitre, x_s est le lot de taille B_s issu de X_s et y_s leurs vraies étiquettes. De la même manière, x_u est le lot de taille B_u issu de X_u et y_u leurs vraies étiquettes (que l'on ne connaît pas pendant l'entraînement).

Pour exploiter X_u, les méthodes SSL peuvent reposer sur plusieurs hypothèses sur les données :

— **continuité** / lissage (*continuity* / *smoothness*) : deux points proches dans l'espace représentant deux exemples distincts doivent appartenir à la même classe ;

— **amas** (*cluster*) : les données forment des amas qui ne sont associés qu'à une seule classe ;

— **variété** (*manifold*) : les données peuvent être projetées dans un espace de dimension inférieure sans perdre trop d'information.

En partant notamment de la première hypothèse, la plupart des méthodes SSL reposent sur l'auto-entraînement [Amini 2022] (*self-training* ou parfois appelé *pseudo-labeling* mais à ne pas confondre avec la méthode SSL *Pseudo-Label*). Celui-ci consiste à utiliser les sorties des modèles sur x_u pour générer indirectement des pseudo-étiquettes notées $\hat{y}_u$ qui serviront d'objectif pour le modèle. Cela nécessite que le modèle soit suffisamment bien entraîné avec X_s pour généraliser sur X_u, et donc un domaine similaire ou une distribution des classes comparables entre X_s et X_u. En apprentissage profond, l'une des méthodes SSL les plus populaires fut le *Pseudo-Label* [Lee 2013]. Il s'agit de l'une des formes les plus simples d'auto-entraînement, que nous détaillerons dans la section 9.2.1.

Cependant, la méthode *Pseudo-Label* seule ne suffit pas toujours à améliorer le modèle, et est très sensible aux biais de confirmation lorsque les pseudo-étiquettes sont erronées. Les méthodes SSL ont donc évolués en ajoutant de nombreux mécanismes qui tournent autour de plusieurs objectifs :

— **maintien de la consistance** [Bachman 2014], en forçant le modèle à produire des sorties similaires entre deux versions augmentées d'un même exemple ;

— **régularisation traditionnelle** en ajoutant des perturbations pendant l'entraînement pour éviter le surapprentissage ;

— **minimisation de l'entropie**, en encourageant le modèle à produire des sorties dont la probabilité maximale est proche de 1, notamment sur X_u.

Le premier objectif est souvent le plus représenté dans les méthodes SSL, comme dans le Π-*model* [Rasmus 2015] ou le *Temporal Ensembling* [Laine 2016], qui utilisent des augmentations spécifiques à leur domaine pour rendre le modèle plus robuste. Nous retrouvons aussi le *Virtual Adversarial Training* [Miyato 2018] qui propose de générer plutôt des exemples adversaires pour perturber les entrées x_u et tromper le modèle. Le *Deep Co-Training* [Qiao 2018] exploite également des méthodes adversaires, mais avec plusieurs modèles différents qui doivent chacun être robustes aux exemples adversaires des autres. Le *Mean Teacher* (parfois appelé *Student-Teacher*) [Tarvainen 2017] emploie également plusieurs modèles : un « étudiant » qui apprend sur les pseudo-étiquettes générées par un autre modèle nommé le « professeur ». D'autres méthodes comme *Interpolation Consistency Training* [Verma 2022] emploient des perturbations comme le Mixup (section 1.4.4.2) pour aider le modèle à généraliser entre les exemples.

En 2019, l'apparition de la méthode « holistique » *MixMatch* [Berthelot 2019] combinera plusieurs idées des méthodes précédentes (Mixup, nouvelles augmentations pour le maintien de la consistance...) avec l'ajout de mécanismes pour régulariser l'apprentissage et minimiser l'entropie du système tout en utilisant qu'un seul modèle. Les années suivantes ont vu l'apparition de beaucoup d'améliorations et de dérivées de *MixMatch*, comme *ReMixMatch* [Kurakin 2020], *FixMatch* [Sohn 2020] ou *Unsupervised Data Augmentation* [Xie 2020], que nous détaillerons dans la section 9.2. Après nos travaux, une partie de ces méthodes a également été comparée sur diverses modalités dans l'étude nommée *Unified Semi-supervised learning Benchmark* [Wang 2022a].

De très nombreuses méthodes dites « holistiques » ou « match », principalement fondées sur *FixMatch*, ont continué à être développées dans les années qui suivirent, améliorant différents aspects de la méthode. Certaines méthodes modifient le maintien de la consistance, comme *CoMatch* [Li 2020], qui introduit un terme issu des apprentissages contrastifs pour minimiser l'entropie. *AlphaMatch* [Gong 2020] remplace l'entropie croisée employée pour les pseudo-étiquettes par une fonction alpha-divergence pour mettre plus d'importance sur les pseudo-étiquettes à forte probabilité. *CRMatch* [Fan 2021] encourage les représentations intermédiaires du modèle à s'éloigner lorsque des augmentations fortes sont appliquées tout en gardant les mêmes étiquettes. *MaxMatch* [Jiang 2022] ajoute un terme pour rapprocher les prédictions des deux versions fortement augmentées les plus éloignées. *SimMatch* [Zheng 2022] rapproche les représentations intermédiaires de la version faiblement et fortement augmentée d'un même exemple. D'autres approches proposent des mécanismes visant à contrôler l'utilisation ou non des pseudo-étiquettes, notamment en modifiant le seuil utilisé par *FixMatch*. *AdaMatch* [Berthelot 2021] ajoute une interpolation entre les sorties du modèle et modifie la sélection des pseudo-étiquettes. *Dash* [Xu 2021e] sélectionne dynamiquement un seuil pour exploiter ou non les pseudo-étiquettes. *FlexMatch* [Zhang 2021]

propose d'ajouter seuils dynamiques à chaque classe pour encourager l'entraînement sur les pseudo-étiquettes dont la probabilité maximale est faible. *FreeMatch* [Wang 2022b] améliore *FlexMatch* en faisant évoluer les seuils en fonction de la confiance actuelle du modèle sur les données de x_u. *SoftMatch* [Chen 2023a] essaie de maximiser le nombre de pseudo-étiquettes utilisées tout en essayant d'éviter celles qui sont erronées à l'aide d'un seuil dynamique.

9.2 Méthodes semi-supervisées sélectionnées

Dans cette partie, nous nous sommes focalisés sur six méthodes SSL différentes : *Pseudo-Label* pour sa simplicité, *Mean Teacher* car il obtenait de bonnes performances sans nécessiter trop d'augmentations, et enfin la plupart des méthodes dites « holistiques » qui existaient au début des expériences : *MixMatch*, *ReMixMatch*, *FixMatch* et *Unsupervised Data Augmentation*.

9.2.1 *Pseudo-Label*

L'une des approches de base les plus populaires est le *Pseudo-Label* (PL) [Lee 2013]. Il consiste en trois étapes simples pendant l'entraînement : générer les pseudo-étiquettes, filtrer celles qui ont une probabilité dépassant un seuil et entraîner le modèle sur ces dernières. Les pseudo-étiquettes sont « binarisées » pour minimiser l'entropie du modèle sur les données non étiquetées. Le fonctionnement de PL est décrit dans la figure 9.1.

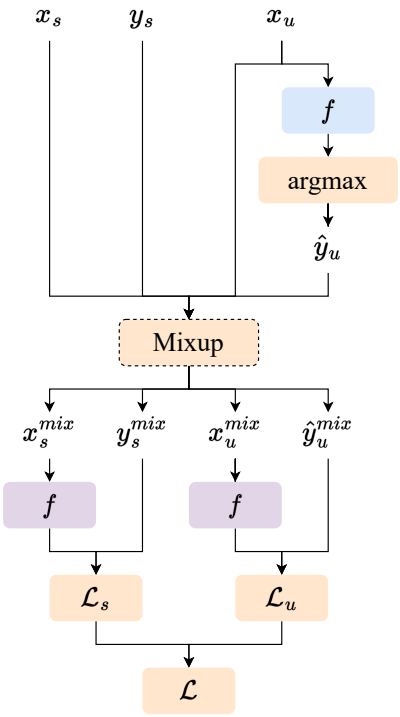


FIGURE 9.1 – Fonctionnement de notre version du *Pseudo-Label*. Les parties apprises du modèle sont en **violet**. Les pseudo-étiquettes sont générées par le modèle f gelé **(en bleu)**. x_s correspond à un exemple étiqueté par y_s, et x_u un exemple pseudo-étiqueté par $\hat{y}_u$. L'exposant $^{\text{mix}}$ correspond aux données mélangées par le Mixup. La partie Mixup n'est ajoutée que pour la version PL+Mixup.

Les deux parties $\mathcal{L}_s$ et $\mathcal{L}_u$ de la fonction de coût sont des entropies croisées traditionnelles :

$$\mathcal{L}_s = \mathrm{CE}(f(x_s), y_s) \tag{9.1a}$$
$$\mathcal{L}_u = \mathrm{CE}(f(x_u), \hat{y}_u) \tag{9.1b}$$

La fonction de coût finale $\mathcal{L}$ est la somme des deux parties, avec un hyperparamètre λ_u :

$$\mathcal{L} = \mathcal{L}_s + \lambda_\mathrm{u} \cdot \mathcal{L}_\mathrm{u} \tag{9.2}$$

9.2.2 *Mean Teacher*

Le *Mean Teacher* (MT) [Tarvainen 2017] est une méthode SSL proposant d'intégrer deux modèles : l'étudiant f qui apprend sur la partie supervisée et le professeur g qui se charge de générer les pseudo-étiquettes sur les données de X_u. Les deux modèles partagent la même architecture, mais pas les mêmes poids.

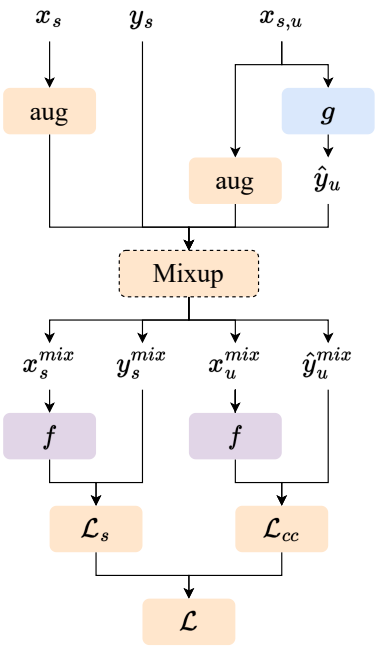


FIGURE 9.2 – Fonctionnement de notre version du *Mean Teacher*. Les pseudo-étiquettes sont générées par le professeur g gelé **(en bleu)**. Comme pour PL, la partie Mixup est utilisée seulement pour MT+Mixup.

L'étudiant apprend au travers de la descente de gradient traditionnelle et le professeur est mis à jour à l'aide d'une moyenne continue exponentielle (EMA) des étudiants précédents et paramétrée par un coefficient α_ema :

$$W_{g,t} = \alpha_\mathrm{ema} \cdot W_{g,t-1} + (1 - \alpha_\mathrm{ema}) \cdot W_{f,t} \tag{9.3}$$

Plus le coefficient α_ema est proche de 1, moins les poids du professeur à l'instant t sont impactés par les poids du dernier étudiant $W_{f,t}$. MT se différencie également de PL par le fait que le professeur génère des pseudo-étiquettes pour les deux parties étiquetées et non étiquetées. Dans le papier d'origine, les auteurs ajoutaient un bruit gaussien et des symétries horizontales sur les images en entrée des modèles, ainsi qu'un Dropout sur les représentations intermédiaires

du modèle. Dans notre version, nous appliquons des augmentations en entrée de l'étudiant seulement [Xie 2019], pour que l'enseignant puisse générer une pseudo-étiquette non perturbée. La fonction de coût pour les données non étiquetées est la MSE, qui est utilisé au lieu de l'entropie croisée, car elle est bornée et moins sensible aux pseudo-étiquettes erronées :

$$\mathcal{L}_{\text{cc}} = \text{MSE}\left(f(x_s), \perp g(x'_s)\right) + \text{MSE}\left(f(x_u), \perp g(x'_u)\right) \tag{9.4}$$

Le symbole « $\perp$ » indique la non propagation du gradient, pour ne pas mettre à jour le réseau professeur (g) durant l'apprentissage. La fonction de coût finale sera la somme des deux parties $\mathcal{L}_s$ et $\mathcal{L}_{\text{cc}}$, avec un coefficient λ_{cc} :

$$\mathcal{L} = \mathcal{L}_s + \lambda_{\text{cc}} \cdot \mathcal{L}_{\text{cc}} \tag{9.5}$$

9.2.3 *MixMatch*

MixMatch (MM) [Berthelot 2019] est une méthode SSL qui n'utilise qu'un seul modèle, mais repose sur de nombreuses augmentations de données. Elle applique également les trois principes (maintien de la consistance, minimisation de l'entropie, régularisation traditionnelle) à l'aide de plusieurs mécanismes intégrés pendant l'entraînement.

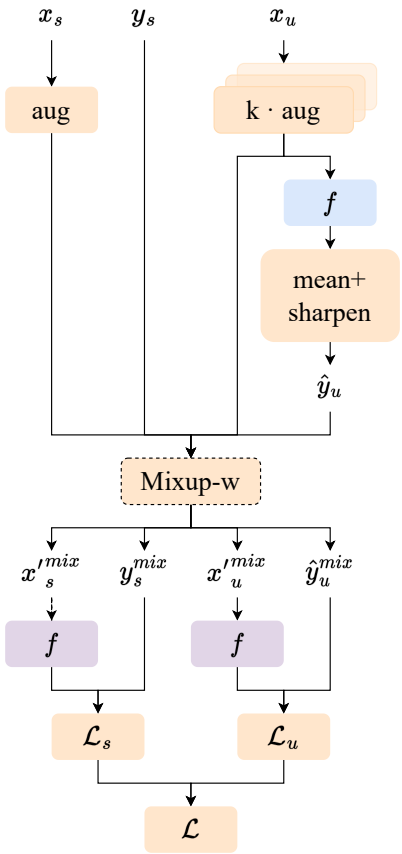


FIGURE 9.3 – Fonctionnement de notre version du *MixMatch*. Les pseudo-étiquettes sont générées par le modèle f gelé **(en bleu)**. La partie Mixup-w qui contient le Mixup est supprimée pour la version MM-Mixup.

Le premier mécanisme est l'introduction d'augmentations de données stochastiques pour rapprocher les pseudo-étiquettes d'un même fichier audio. Premièrement, la pseudo-étiquette $\hat{y}_u$ est générée à partir de k versions augmentées du même fichier x_u, nommées x'_u. Les probabilités prédites sont ensuite moyennées pour former la pseudo-étiquette $\hat{y}_u$:

$$\hat{y}_u = \frac{1}{k} \sum_{i=1}^{k} f(x'_{u,i}) \tag{9.6}$$

Le second mécanisme est l'« aiguisage » (*sharpen*), qui est une fonction qui rehausse la probabilité maximale des pseudo-labels :

$$\text{sharpen}(p, T)_i = p_i^{1/T} \Big/ \sum_{j=1}^{|p|} p_j^{1/T} \tag{9.7}$$

La température T est un hyperparamètre permettant de contrôler l'importance de la fonction et p correspond aux probabilités de la pseudo-étiquette $\hat{y}_u$. Lorsque T tend vers 0, la probabilité maximale de p tend vers 1 et les autres tendent vers 0. À l'inverse, lorsque T tend vers $+\infty$, la distribution de probabilité tend vers une loi uniforme. Le troisième mécanisme est l'utilisation d'une méthode fondée sur la version asymétrique du Mixup, que nous nommons « Mixup-w ». Cette méthode mélange les données étiquetées et non étiquetées, ainsi que leurs étiquettes et pseudo-étiquettes. Pour cela, les données x_s et x_u sont concaténées dans une liste W, puis l'ordre des exemples de W est mélangé et enfin le Mixup est appliqué entre x'_s, x'_u et W, comme montré par les équations 9.8 et 9.9.

$$x'^{\text{mix}}_s = \text{Mixup}(x'_s | W_{1...B_s}) \tag{9.8}$$

$$x'^{\text{mix}}_u = \text{Mixup}(x'_u | W_{|x_s|+1...|W|}) \tag{9.9}$$

Les données sont donc augmentées et mélangées avec d'autres exemples du lot W. À noter que nous utilisons la version asymétrique du Mixup qui permet de garantir que le lot étiqueté x'^{mix}_s soit proche de x'_s et que x'^{mix}_u soit proche de x'_u. Enfin, nous calculons les termes $\mathcal{L}_s$ et $\mathcal{L}_u$ à l'aide d'entropies croisées traditionnelles :

$$\mathcal{L}_s = \frac{1}{B_s} \sum_{(x'^{\text{mix}}_s, y^{\text{mix}}_s)} \text{CE}(f(x'^{\text{mix}}_s), y^{\text{mix}}_s) \tag{9.10}$$

$$\mathcal{L}_u = \frac{1}{N_a \cdot B_u} \sum_{(x'^{\text{mix}}_u, \hat{y}^{\text{mix}}_u)} \text{CE}(f(x'^{\text{mix}}_u), \hat{y}^{\text{mix}}_u) \tag{9.11}$$

MM utilisait à l'origine la fonction MSE pour $\mathcal{L}_u$, mais nous avons choisi de la remplacer par l'entropie croisée qui a donné de meilleurs résultats. La fonction de coût finale est :

$$\mathcal{L} = \mathcal{L}_s + \lambda_u \cdot \mathcal{L}_u \tag{9.12}$$

Pour éviter que le modèle ne s'entraîne sur les pseudo-étiquettes probablement erronées au début de l'entraînement, l'hyperparamètre λ_u vaut zéro au départ, puis il est linéairement augmenté (*rampup* ou *warmup*) pendant N_{λ_u} itérations.

9.2.4 *ReMixMatch*

ReMixMatch (RMM) [Kurakin 2020] est une évolution de MM qui rajoute et modifie plusieurs mécanismes pour améliorer les performances et la généralisation du modèle. Le fonctionnement de notre RMM est décrit dans la figure 9.4. Premièrement, MSE de $\mathcal{L}_u$ est remplacée par une CE standard. Un mécanisme d'alignement des distributions des pseudo-étiquettes est appliqué pour corriger la distribution des pseudo-étiquettes. Les augmentations sont séparées en deux groupes dits « faibles » et « forts ». Enfin, deux termes sont ajoutés à la fonction de coût pour améliorer les performances et éviter l'effondrement lors de l'apprentissage du modèle.

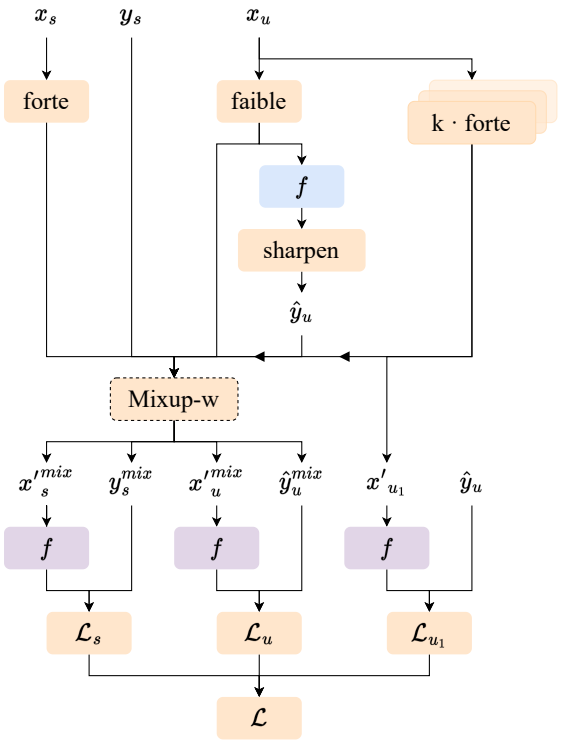


FIGURE 9.4 – Fonctionnement de notre version du *ReMixMatch*. Les pseudo-étiquettes sont générées par le modèle f gelé **(en bleu)**. La partie Mixup-w qui contient le Mixup est supprimée pour la version RMM-Mixup.

Les augmentations faibles correspondent à des perturbations qui changent peu la prédiction du modèle, tandis que les augmentations fortes modifient suffisamment l'exemple d'origine pour potentiellement tromper le modèle. Mais dans les deux cas, les augmentations ne doivent pas modifier l'exemple au point de changer sa vraie classe y. Chaque exemple non étiqueté x_u donne une version faiblement augmentée (notée faible(x_u)) et N_a versions fortement augmentées (notées forte(x_{u_i})). La version faiblement augmentée sert à générer la pseudo-étiquette $\hat{y}_u$ qui est utilisée pour toutes les versions augmentées de x_u. RMM réalise deux post-traitements sur $\hat{y}_u$: l'alignement de distribution (*Distribution Alignment*), puis le *sharpen* de MM. Le mécanisme d'alignement consiste à rapprocher les pseudo-étiquettes générées de la distribution des classes de X_s. Pour cela, deux distributions p_s et p_u sont estimées avec les N lots précédents, à l'aide des vraies étiquettes y_s ainsi que des pseudo-étiquettes $\hat{y}_u$. Une normalisation ℓ_1 est appliquée après l'alignement pour garder une somme de probabilités à 1. L'équation ci-dessous 9.13 présente le

calcul effectué par le mécanisme d'alignement.

$$\hat{y}_u = \text{Normalize}\left(f(x_u) \cdot \frac{p_s}{p_u}\right) \tag{9.13}$$

Cela permet d'éviter au modèle de s'entraîner sur des pseudo-étiquettes qui auraient une distribution de classes trop éloignée de celle des vraies classes de X_s. Après l'alignement des probabilités fait, nous appliquons la fonction *sharpen* 9.7. Comme pour MM, le Mixup asymétrique est appliqué entre les données étiquetées et non étiquetées et permet de calculer le premier et second terme de la fonction de coût. Le troisième terme (équation 9.14) consiste à entraîner le système sur le premier exemple fortement augmenté x_{u_1} avec $\hat{y}_u$ sans Mixup :

$$\mathcal{L}_{u_1} = \frac{1}{B_u} \sum_{(x'_{u_1}, \hat{y}_u)} \text{CE}\left(f(x'_{u_1}), \hat{y}_u\right) \tag{9.14}$$

Enfin, le dernier et quatrième terme est inspiré des apprentissages autosupervisés (*self-supervised*) [Gidaris 2018] et permet d'éviter l'effondrement de l'apprentissage du réseau. Le principe est d'appliquer une perturbation parmi un ensemble prédéfini de perturbations possibles et de demander au modèle de prédire laquelle a été appliquée sur l'exemple x_{u_1}. Dans le papier d'origine, x_{u_1} est une image et les perturbations possibles sont des rotations d'angle 0, 90, 180 ou 270 degrés. Pour l'audio, nous avons tenté de remplacer ce terme par des symétries horizontales et verticales du spectrogramme, sans résultat. Nous avons donc supprimé ce terme dans nos expériences, car l'apprentissage convergeait même sans ce dernier. Finalement, la fonction de coût de RMM est dans notre cas :

$$\mathcal{L} = \mathcal{L}_s + \lambda_u \cdot \mathcal{L}_u + \lambda_{u_1} \cdot \mathcal{L}_{u_1} \tag{9.15}$$

9.2.5 *FixMatch*

FixMatch (FM) [Sohn 2020] est une autre méthode SSL, présentée comme une simplification de RMM tout en étant plus performante. La figure 9.5 montre le déroulement d'une itération d'entraînement de FM. La méthode garde les deux types d'augmentations faibles et fortes, mais supprime les deux termes de la fonction de coût ajouté par RMM $\mathcal{L}_{u_1}$ et $\mathcal{L}_r$. Elle réduit le nombre d'augmentations fortes appliquées à 1, supprime l'alignement des distributions ainsi que le Mixup. Cependant, cette méthode rajoute un système de masquage afin éviter que le modèle n'apprenne sur des pseudo-étiquettes peu confiantes et donc probablement erronées.

L'augmentation faible appliquée à x_u sert à générer la pseudo-étiquette qui sera ensuite associée à la version fortement augmentée de x_u :

$$\hat{y}_u = f\left(\text{faible}(x_u)\right) \tag{9.16}$$

Le système de masquage consiste à éviter que le modèle f apprenne sur des pseudo-étiquettes peu confiantes. Pour cela, nous mettons le terme $\mathcal{L}_u$ à 0 si la probabilité maximale de $\hat{y}_u$ ne dépasse pas un certain seuil τ fixé à l'avance :

$$\begin{aligned} \text{masque} &= \mathbb{1}\left(\max\left(\hat{y}_u\right) > \tau\right) \\ \mathcal{L}_u &= \text{masque} \cdot \text{CE}\left(f\left(\text{forte}\left(x_u\right)\right), \text{argmax}\left(\hat{y}_u\right)\right) \end{aligned} \tag{9.17}$$

La fonction de coût finale sera la même que pour MM (voir équation 9.12) :

$$\mathcal{L} = \mathcal{L}_s + \lambda_u \cdot \mathcal{L}_u \tag{9.18}$$

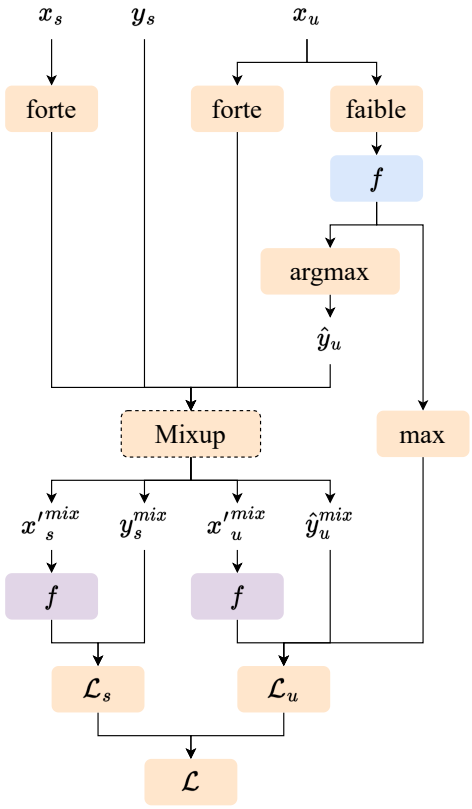


FIGURE 9.5 – Fonctionnement de notre version du *FixMatch*. Les pseudo-étiquettes sont générées par le modèle f gelé **(en bleu)**. La valeur maximale de la pseudo-étiquette sera utilisée pour masquer ou non la valeur du coût $\mathcal{L}_u$. Le Mixup est appliqué uniquement pour la version FM+Mixup.

9.2.6 *Unsupervised Data Augmentation*

Unsupervised Data Augmentation (UDA) [Xie 2020] est une autre méthode SSL très proche de FM, mais originellement testée pour la classification d'image et de texte. La figure 9.6 montre le fonctionnement de UDA.

Les pseudo-étiquettes sont calculées à partir des données x_u non augmentées après l'application de la fonction *softmax-sharpening* :

$$\hat{y}_u = \exp\left(z/T\right) \Big/ \sum_i \exp\left(z_i/T\right) \tag{9.19}$$

La valeur de z correspond aux sorties du modèle f avant la fonction d'activation *softmax* (*logits*) et T est un paramètre de température qui joue un rôle similaire à celui de *sharpen* dans l'équation 9.7. Le terme supervisé $\mathcal{L}_s$ de la fonction de coût est une CE traditionnelle entre x_s non augmenté et y_s. Le terme non supervisé $\mathcal{L}_u$ est similaire à celui de FM, si ce n'est que nous ne binarisons pas la pseudo-étiquette $\hat{y}_u$:

$$\begin{aligned} \text{masque} &= \mathbb{1}\left(\max(\hat{y}_u) > \tau\right) \\ \mathcal{L}_u &= \text{masque} \cdot \text{CE}\left(f\left(\text{forte}(x_u)\right), \hat{y}_u\right) \end{aligned} \tag{9.20}$$

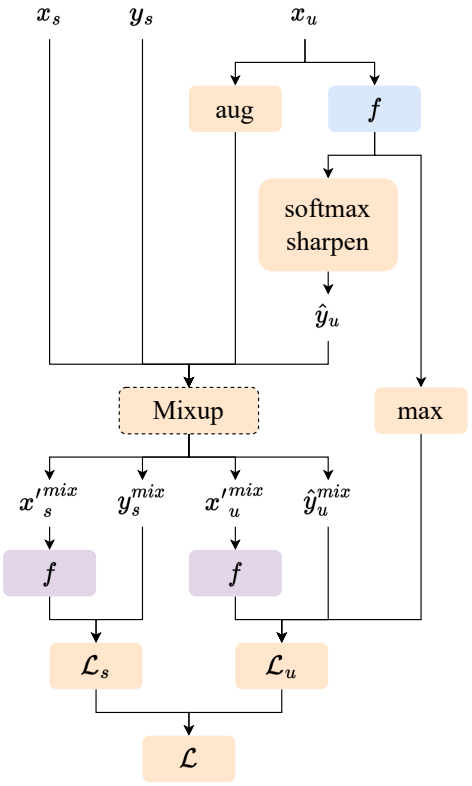


FIGURE 9.6 – Fonctionnement de notre version du UDA. Les pseudo-étiquettes sont générées par le modèle f gelé **(en bleu)**. La valeur maximale de la pseudo-étiquette sera utilisée pour masquer ou non la valeur du coût $\mathcal{L}_u$. Le Mixup est appliqué uniquement pour la version UDA+Mixup.

La fonction de coût finale est la même que pour MM ou FM (voir équation 9.12), c'est-à-dire la somme des deux termes $\mathcal{L}_s$ et $\mathcal{L}_u$ avec un coefficient λ_u.

9.3 Configuration expérimentale

9.3.1 Bases de données de classification audio

Pour évaluer nos méthodes SSL, nous avons choisi trois bases de données audio mono-étiquettes de natures différentes : ESC-10, UBS8K et GSC.

Environmental Sound Classification-10 (ESC-10) [Piczak 2015] [51] est une petite base de données de 400 fichiers audio de cinq secondes, chacun étiqueté par dix classes différentes : chien, coq, tronçonneuse, pluie, feu crépitant, hélicoptère, bébé qui pleure, tic-tac d'une horloge, éternuement et vagues de la mer. Les fichiers sont disponibles avec un taux d'échantillonnage de 44,1 kHz, mono-canal, 16 bits et sont transformés en log-Mel spectrogrammes de taille 64×431. La base de données est divisée en cinq sous-parties (*folds*) dont les classes sont équilibrées pour réaliser une validation croisée.

51. https://github.com/karolpiczak/ESC-50

Urban Sound 8k (UBS8K) [Salamon 2014] [52] contient 8742 enregistrements audio d'une à quatre secondes, divisés en dix *folds* quasi équilibrés pour un total de 18,5 heures. Le jeu de données contient dix classes différentes issues du milieu urbain : climatiseur, klaxon de voiture, enfants qui jouent, aboiement de chien, forage, bruit de moteurs, coup de feu, marteau-piqueur, sirène et musique de rue. Les fichiers audio sont échantillonnés à 22,05 kHz et sont complétés avec des zéros pour atteindre une taille fixe correspondant à quatre secondes. Les deux canaux de chaque fichier sont moyennés et les log-Mel spectrogrammes calculés sont de taille de 64×173. Les classes ne sont pas totalement équilibrées avec 429 exemples d'aboiements de chien, 929 sirènes, 374 coups de feu et 1000 exemples pour les sept autres classes.

Google Speech Commands v2 (GSC) [Warden 2018] [53] est un large corpus d'enregistrements vocaux courts contenant 100 503 exemples d'une durée allant jusqu'à une seconde, échantillonnés à 16 kHz. Chaque fichier correspond à la prononciation de l'un des 35 mots anglais possibles : *bed, bird, cat, dog, down, eight, five, follow, forward, four, go, happy, house, learn, left, marvin, nine, no, off, on, one, right, seven, sheila, six, stop, three, tree, two, up, visual, wow, yes, zero, backward*. Contrairement aux deux précédentes bases de données, GSC est divisée en trois parties : 85 511 exemples d'entraînement, 10 102 de validation et 4890 pour le test. Des fichiers supplémentaires de bruit sont également disponibles, mais ils ne seront pas utilisés dans les expériences qui suivront. Les signaux audio sont complétés avec des zéros pour atteindre une seconde, puis sont transformés en log-Mel spectrogramme de taille 64×32.

Le code utilisé pour charger les trois bases de données est fondé sur celui créé à l'origine par Léo Cancès [54], un collègue de mon équipe qui a également travaillé sur la classification de fichiers audio.

9.3.2 Modèle

Pour nos expériences, nous avons choisi le réseau de neurones WideResNet-28-2 [Zagoruyko 2017], déjà utilisé pour tester les méthodes SSL MM et FM pour l'image et relativement performant avec seulement 1,4 million de paramètres entraînables. L'architecture est la même dans toutes nos expériences, à l'exception de la dernière couche de classification qui contient 10 neurones pour ESC-10 et UBS8K et 35 pour GSC.

La structure du WideResNet-28-2 est décrite dans la figure 9.7 et se base sur un composant nommé le bloc *Basic*. Ce dernier consiste en une convolution suivie d'une couche de normalisation par lot et d'une fonction d'activation *ReLU*, décrite dans l'équation 9.21. Notons SeqBasic(N_b, N_f) la séquence de N_b blocs *Basic* contenants chacun N_f filtres.

$$\text{Basic}(N_f) = (\text{Conv3x3 @ } N_f, \text{BN}, \text{ReLU}) \tag{9.21}$$

Chaque bloc *Basic* contient une convolution 3×3 et respectivement 32, 32, 64 et 128 filtres N_f. Avant l'application de la couche de classification finale, nous obtenons un vecteur de taille 256 qui sera projeté vers le nombre de classes. L'implémentation de WideResNet-28-2 que nous utilisons est fondée sur celle de PyTorch [55].

52. https://urbansounddataset.weebly.com/urbansound8k.html
53. https://blog.research.google/2017/08/launching-speech-commands-dataset.html
54. https://github.com/lcances/semi-supervised
55. https://github.com/pytorch/vision/blob/main/torchvision/models/resnet.py

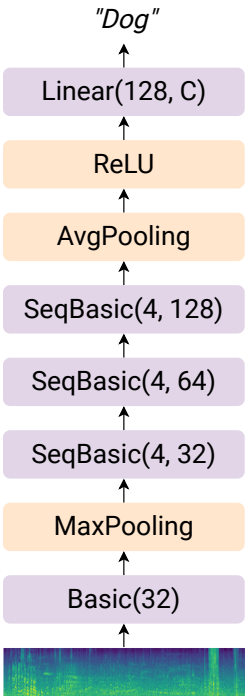


FIGURE 9.7 – Architecture du WideResNet-28-2.

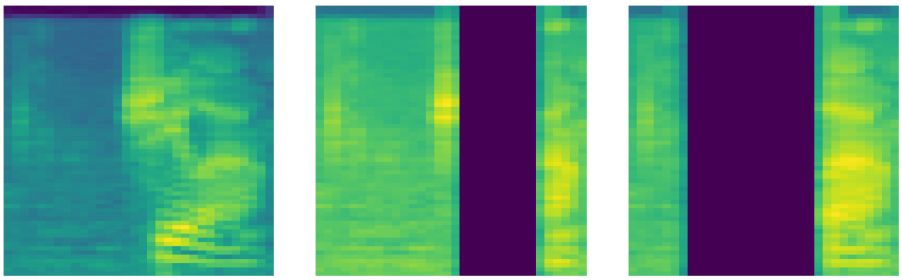
9.3.3 Augmentations

Les méthodes MM, RMM, FM et UDA dépendent énormément des augmentations de données qui sont appliquées pour stabiliser l'entraînement. Pour cela, nous avons choisi de prendre trois augmentations inspirées de celles utilisées en image.

L'**Occlusion**, parfois aussi appelé masquage temporel (*TimeMasking*), est une modification simple du signal audio qui consiste à mettre à zéro les valeurs du signal dans un certain intervalle. La position de cet intervalle est aléatoire dans le signal et sa taille est tirée aléatoirement dans un autre intervalle de ratios qui dépend de la longueur du signal d'origine. Par exemple, si l'intervalle de ratios vaut [0,1, 0,2], seulement 10 à 20% du signal pourront être masqués pendant l'entraînement. Dans notre cas, l'occlusion est appliquée sur le signal audio brut. Un exemple d'occlusion faible et forte pour un fichier audio est donné dans les figures 9.8.

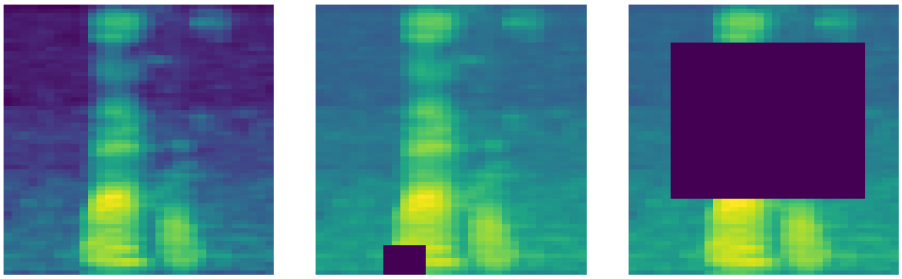
CutOut [DeVries 2017] est une augmentation inspirée du domaine de l'image qui masque une zone du spectrogramme sous forme de rectangle. Nous pouvons le voir comme une généralisation de l'occlusion qui masque une partie temporelle et fréquentielle. Comme pour l'occlusion, la position et la taille sont aléatoires et dépendent de la taille du spectrogramme. La valeur utilisée pour masquer est -80 dB, qui correspond au niveau d'énergie minimale du spectrogramme. Un exemple de CutOut faible et forte pour un fichier audio est donné dans les figures 9.9.

Enfin, la ***Speed Perturbation*** [Ko 2015] accélère ou ralentit le signal à partir d'un facteur choisi aléatoirement dans un intervalle de ratios. Comme le résultat donne un signal plus long ou plus court, il est nécessaire de tronquer ou remplir par des zéros pour garder la même longueur



(a) Non augmenté. (b) Faiblement augmenté. (c) Fortement augmenté.

FIGURE 9.8 – Spectrogrammes issus de la partie entraînement de GSC (n°57767) respectivement non augmentée, faiblement augmentée et fortement augmentée par l'*Occlusion*. Le mot prononcé est « *sheila* ».



(a) Non augmenté. (b) Faiblement augmenté. (c) Fortement augmenté.

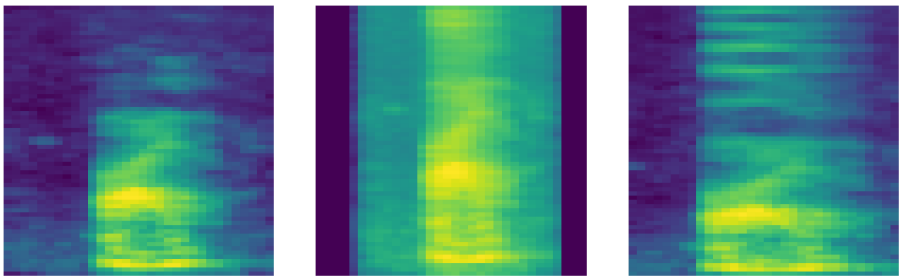
FIGURE 9.9 – Spectrogrammes issus de la partie entraînement de GSC (n°981) respectivement non augmentée, faiblement augmentée et fortement augmentée par le *CutOut*. Le mot prononcé est « *backward* ».

du signal. Un exemple de perturbation de vitesse faible et forte pour un fichier audio est donné dans les figures 9.10.

À noter que ces augmentations sont volontairement simples et ne dépendent pas de la tâche à accomplir. En effet, l'objectif de ces augmentations est d'être appliquée sur les trois bases de données de classifications, qui contiennent de très différents bruits. Nous n'avons donc pas utilisé certaines augmentations comme le décalage de hauteur (*pitch shift*) ou la réverbération qui modifierait les descriptions des fichiers audio si nous les appliquions pour la DTAA. Les paramètres de ces augmentations se trouvent dans le tableau 9.1.

TABLE 9.1 – Hyperparamètres des augmentations de données faibles et fortes.

Augmentation	**Paramètre**	**Version faible**	**Version forte**
Occlusion	taille maximale	[0,25, 0,25]	[0,75, 0,75]
CutOut	taille maximale	[0,10, 0,50]	[0,50, 1,00]
Speed perturbation	ratio	[0,50, 1,50]	[0,25, 1,75]



(a) Non augmenté. (b) Faiblement augmenté. (c) Fortement augmenté.

FIGURE 9.10 – Spectrogrammes issus de la partie entraînement de GSC (n°15315) respectivement non augmentée, faiblement augmentée et fortement augmentée par la *Speed Perturbation*. Le mot prononcé est « *five* ».

9.3.4 Hyperparamètres

Dans toutes les expériences qui suivent, nous avons choisi en premier lieu les meilleurs hyperparamètres pour un apprentissage supervisé complet sur GSC utilisant 100% des données d'entraînements. Nous gardons les valeurs des méthodes de chaque papier SSL origine dans la mesure du possible pour obtenir une comparaison équitable entre elles et garder une mise en œuvre simple. Pour la suite, nous notons les apprentissages supervisés utilisant 10% et 100% des données respectivement SUP10 et SUP100.

Paramètres en commun. Nous optimisons le réseau avec Adam et les valeurs par défaut de PyTorch : β_1 mis à 0,9, β_2 mis à 0,999 et ϵ mis à 10^{-8}. Les tailles des lots pour les méthodes supervisées sont de 30 pour ESC-10 et 128 pour UBS8K et GSC. Pour les méthodes SSL, nous utilisons 30 exemples supervisés et 30 exemples non supervisés par lot sur ESC-10. Nous récupérons 128 exemples supervisés et 128 non supervisés sur UBS8K et GSC par lot. La proportion de données annotées et non annotées dans le lot est donc égale pour faciliter l'intégration du Mixup dans les méthodes SSL. En effet, le Mixup est appliqué entre croisant les données étiquetées avec les celles non étiquetées et inversement, ce qui demande à avoir le même nombre d'éléments. La faible taille de lot de ESC-10 est due aux limites des méthodes et à la base de données. En effet, ESC-10 contient 400 exemples, dont 320 pour chaque entraînement, ce qui ne laisse que 32 exemples étiquetés pour les méthodes SSL et SUP10. Comme nous souhaitons garder les mêmes distributions de classes, nous prenons uniquement trois exemples pour chacune des 10 classes possibles. Nous obtenons donc seulement 30 exemples supervisés pour SUP10, ce qui limite la taille du lot. Pour l'apprentissage, nous utilisons un pas dynamique suivant une fonction cosinus décroissante 3.3, comme pour les systèmes de DTAA. Le paramètre α utilisé dans le Mixup est mis à 0,75 pour toutes les expériences SSL et à 0,4 pour les expériences supervisées. Nous entraînons les modèles pendant 300 époques et nous sélectionnons le modèle selon le taux d'erreur minimal sur la validation de chaque base de données. Enfin, chaque méthode SSL possède un ensemble d'hyperparamètres qui leur sont propres, que nous détaillons ci-dessous.

PL. La valeur du paramètre λ_u vaut 0 pendant les 50 premières époques puis augmente linéairement pendant 200 époques jusqu'à atteindre la valeur 1. Sa valeur n'est mise à jour qu'au début de chaque époque d'entraînement.

MT. Le coefficient de la fonction de coût λ_{cc} croît en suivant une courbe exponentielle, dépendant de l'époque courante n_e :

$$\lambda_{cc,n_e} = \lambda_{cc,max} \cdot \left(1 - \exp\left(-5 \cdot \left(1 - \min\left(\frac{n_e}{\text{wl}}, 1\right)\right)^2\right)\right) \tag{9.22}$$

La valeur finale de λ_{cc} est 1 et sera atteinte à l'époque définie par l'hyperparamètre wl (*warmup length*) qui vaut 50. Le professeur est mis à jour à chaque lot avec un coefficient de décroissance α_{ema} mis à 0,999.

MM. Nous utilisons $N_a = 2$ augmentations pour le pseudo-étiquetage, la température T vaut 0,5 et le coefficient λ_u est linéairement augmenté pendant les $N_{\lambda_u} = 16\,000$ premiers lots d'entraînement.

RMM. Les coefficients λ_u et λ_{u_1} de la fonction de coût sont constants et respectivement mis à 1,5 et 0,5. Nous utilisons $N_a = 2$ augmentations sur les données non supervisées et nous gardons les dernières étiquettes et pseudo-étiquettes des 128 lots pour aligner les distributions des pseudo-étiquettes.

FM. Le seuil du masque pour les pseudo-étiquettes τ est mis à 0,95 et le coefficient λ_u est mis à 1 comme dans le papier d'origine.

UDA. Comme pour la plupart des méthodes, le coefficient non supervisé λ_u est mis à 1, la température T du *softmax-sharpen* est mise à 0,5 et le seuil du masque est mis à 0,8.

9.4 Résultats

9.4.1 Apprentissages supervisés

Dans cette section, nous étudions d'abord les résultats des méthodes purement supervisées en utilisant plusieurs mécanismes d'augmentations (Mixup, faible, faible+Mixup, forte, forte+Mixup) et en faisant varier la proportion d'étiquettes utilisées dans chaque base de données (10%, 100%). Ces premiers résultats se trouvent dans le tableau 9.2. Nous avons également ajouté les scores de l'état de l'art de chaque base de données parmi ceux qui utilisent des modèles convolutifs. Les scores sont donnés en taux d'erreurs, ce qui correspond aux nombres de prédictions inexactes faites sur l'ensemble de données.

ESC-10. Avec 10% des données, le Mixup semble faire baisser les scores du modèle de 4% (-12,50% relatif), mais les augmentations faibles et fortes offres des gains importants, respectivement de 9,33% et 9% (29,16% et 28,13% relatif). Avec 100% des données, les augmentations faibles et fortes se rapprochent de l'état de l'art avec des gains importants de 3,33% et 3% (41,63% et 37,50% relatifs), mais le Mixup n'améliore toujours pas le score avec une perte de 0,33% (-4,13% relatif). Nous pouvons également noter que l'utilisation d'augmentations faibles et fortes semble légèrement diminuer l'écart-type des résultats sur ce jeu de données.

TABLE 9.2 – Taux d'erreurs des apprentissages supervisés (en %) sur ESC-10, UBS8K et GSC. Les références des scores de la littérature. Les valeurs en **gras** et soulignées correspondent respectivement à la première et la seconde valeur de chaque colonne. Les nombres situés après le symbole « ± » correspondent aux écart-types calculés sur les différents *folds*.

Base de données	**ESC-10**		**UBS8K**		**GSC**	
Proportion annotée	**10%**	**100%**	**10%**	**100%**	**10%**	**100%**
Littérature (CNN)	[Guzhov 2020]		[Gazneli 2022]		[Vygon 2021]	
	-	3,00	-	14,50	-	3,00
Supervisé	32,00±6,17	8,00±5,06	33,80±4,82	23,29±5,80	10,01	4,94
+Mixup	36,00±5,22	8,33±4,56	31,41±5,56	22,04±5,99	8,83	3,86
+faible	**22,67±3,46**	4,67±3,43	27,08±4,58	20,09±5,50	7,62	3,90
+faible+Mixup	24,67±4,92	**4,67±1,39**	**23,75±4,73**	**17,96±3,64**	**6,58**	3,00
+forte	23,00±5,19	5,00±2,64	25,58±4,15	20,69±4,92	7,60	3,27
+forte+Mixup	24,00±8,71	5,00±4,25	24,73±4,42	18,52±4,38	6,86	**2,98**

UBS8K. Avec 10% des données, nous pouvons voir que toutes les augmentations offrent des gains de performances. Le meilleur cas est atteint avec les augmentations faibles combinées au Mixup qui atteint 23,75% de taux d'erreur, soit 10,05% d'amélioration (29,73% relatif). Avec 100% des données, les augmentations apportent toujours des gains, mais ils sont relativement moins importants. La meilleure configuration reste la combinaison de l'augmentation faible avec le Mixup, qui apporte 5,35% de gain (22,89% relatif). Une remarque importante est que la meilleure configuration utilisant 10% des données (faible+Mixup, 23,75%) atteint presque le score utilisant 100% des données sans augmentation (23,29%), indiquant que ces dernières semblent extrêmement efficaces pour ce jeu de données.

GSC. Avec 10% des données, nous observons que les augmentations offrent toutes des gains par rapport à l'apprentissage supervisé standard comme pour UBS8K. La meilleure configuration est faible+Mixup, qui offre un gain de 3,53% (34,27% relatif). Avec 100% des données, nous constatons que les augmentations de données fournissent toujours des gains significatifs. Cette fois, le meilleur cas est obtenu avec les augmentations fortes combinées au Mixup, avec 1,96% de gain (39,68% relatif), ce qui est légèrement au-dessus de l'état de l'art des modèles CNN actuels.

Globalement, les augmentations de données appliquées offrent toujours des gains intéressants sur chaque base de données. La combinaison du Mixup avec les augmentations faibles donne les meilleurs résultats dans la plupart des cas. L'exception est l'utilisation du Mixup sur ESC-10, qui semble légèrement dégrader les performances.

9.4.2 Apprentissages semi-supervisés

Maintenant que nous avons des scores de référence, nous pouvons comparer ces résultats avec les méthodes semi-supervisées pour voir si ces dernières arrivent à exploiter leur partie non étiquetée X_u. Dans le cadre de la classification audio, le Mixup est encore plus pertinent que pour les images, car la somme de deux fichiers audio correspond bien à la somme de leurs classes correspondantes dans la majorité des cas. C'est pourquoi nous avons décidé d'intégrer le

Mixup au sein des méthodes SSL qui ne l'utilisaient pas à l'origine (PL, MT, FM et UDA) et d'observer le gain de performance. Le tableau 9.3 donne les scores obtenus par les différentes méthodes semi-supervisées (avec ou sans Mixup), comparées aux apprentissages supervisés sans augmentation ainsi qu'aux meilleurs apprentissages supervisés qui exploitent des augmentations (c.a.d. les meilleurs de chaque colonne du tableau 9.2.

TABLE 9.3 – Taux d'erreurs des apprentissages semi-supervisés (en %) sur ESC-10, UBS8K et GSC. Les valeurs en **gras** et soulignées correspondent respectivement à la première et la seconde valeur de chaque colonne. Les nombres situés après le symbole « ± » correspondent aux écart-types calculés sur les différents *folds*.

Base de données	**ESC-10**		**UBS8K**		**GSC**	
Proportion annotée	**10%**	**100%**	**10%**	**100%**	**10%**	**100%**
Supervisé	32,00±6,17	8,00±5,06	33,80±4,82	23,29±5,80	10,01	4,94
Meilleur supervisé	22,67±3,46	4,67±1,39	23,75±4,73	17,96±3,64	6,58	2,98
PL	37,83±8,99	-	29,68±5,07	-	9,98	-
PL+Mixup	36,00±3,88	-	25,61±5,74	-	5,29	-
MT	30,17±8,96	-	28,03±6,67	-	6,21	-
MT+Mixup	29,83±5,45	-	22,76±5,33	-	4,38	-
MM-Mixup	17,33±3,84	-	20,42±4,88	-	4,49	-
MM	15,33±5,58	-	**18,02±4,00**	-	**3,25**	-
RMM-Mixup	32,50±11,71	-	38,23±6,15	-	5,15	-
RMM	**12,00±5,55**	-	28,41±6,54	-	3,54	-
FM	13,33±2,89	-	21,44±4,16	-	4,44	-
FM+Mixup	14,67±7,21	-	18,27±3,80	-	3,31	-
UDA	35,67±5,99	-	26,74±7,18	-	4,45	-
UDA+Mixup	31,17±9,55	-	20,06±5,83	-	3,52	-

ESC-10. Sur cette petite base de données, le meilleur score est atteint par RMM avec seulement 12% de taux d'erreur, soit 10,67% d'amélioration au meilleur SUP10 (47,07% relatif). Ce score est suivi de près par ceux de FM, FM+Mixup, MM et MM-Mixup. Le Mixup semble jouer un rôle positif dans la majorité des méthodes (sauf FM) quand nous l'ajoutons aux méthodes qui ne l'utilisaient pas, ou un rôle négatif lorsque nous le soustrayons aux autres méthodes. Cependant, les méthodes PL, MT et UDA restent en dessous des meilleurs apprentissages supervisés qui n'utilisent que 10% des données.

UBS8K. Dans la section précédente, nous avions vu que les augmentations avaient un très fort impact sur les scores de cette base de données en particulier. Effectivement, le meilleur SUP10 rivalise avec l'apprentissage non augmenté SUP100. Les méthodes SSL qui reposent sur beaucoup d'augmentations permettent donc un gain important, avec un taux d'erreur allant jusqu'à 18,02% pour MM, soit 5,73% de gain. Ce score est très proche du meilleur apprentissage SUP100 qui est de 17,96%, indiquant que la méthode MM a permis au modèle de généraliser en exploitant dix fois moins d'étiquettes. MM est suivi de près par FM+Mixup avec 18,27% d'erreur. Chaque méthode SSL semble apporter un gain par rapport au SUP10, notamment avec l'introduction du

Mixup qui fonctionne bien sur cette base de données. RMM semble cependant être en dessous de la performance des autres méthodes SSL, malgré sa large utilisation d'augmentations pendant l'entraînement.

GSC. La meilleure performance sur cette base de données est atteinte par la méthode MM avec 3,25 de taux d'erreur, soit 3,33% de gain (50,61% relatif) par rapport aux 6,58% d'erreur du meilleur SUP10. La méthode est très proche du meilleur SUP100, avec seulement 0,27% d'erreur absolue en plus (soit 17 fichiers sur 4890). MM est également suivi de près par les méthodes FM+Mixup, UDA+Mixup et RMM. Nous remarquons que toutes les méthodes SSL obtiennent un gain assez important lorsqu'elles intègrent le Mixup dans leur fonctionnement, même pour les méthodes les moins efficaces comme PL ou MT.

Dans l'ensemble, les méthodes SSL permettent d'obtenir un gain important sur chaque base de données audio. Les méthodes MM et FM+Mixup semblent offrir les meilleures performances globales sur les trois bases de données, confirmant l'intérêt du Mixup et des méthodes SSL.

9.4.3 Temps d'entraînements

Pour sélectionner la meilleure méthode SSL, nous avons décidé de mesurer leur temps d'exécution moyen par rapport à un apprentissage supervisé. En effet, certaines méthodes ajoutent de nombreuses augmentations qui nécessitent de calculer plusieurs fois les sorties du modèle, ce qui peut augmenter considérablement le nombre d'opérations à réaliser par lot. Pour comparer les temps d'entraînements, nous calculons les durées d'exécutions moyennes D :

$$D = \frac{d}{N_f \cdot K \cdot B} \tag{9.23}$$

Sur chaque base de données, la valeur d correspond à la durée d'entraînements en secondes, N_f correspond au nombre de *folds*, K au nombre d'époques et B à la taille du lot. Nous divisons ensuite D par la durée d'un apprentissage supervisé complet (SUP100) et nous moyennons ces valeurs sur les trois bases de données pour obtenir les durées d'exécution moyennes normalisées de chaque méthode, données par l'histogramme 9.11.

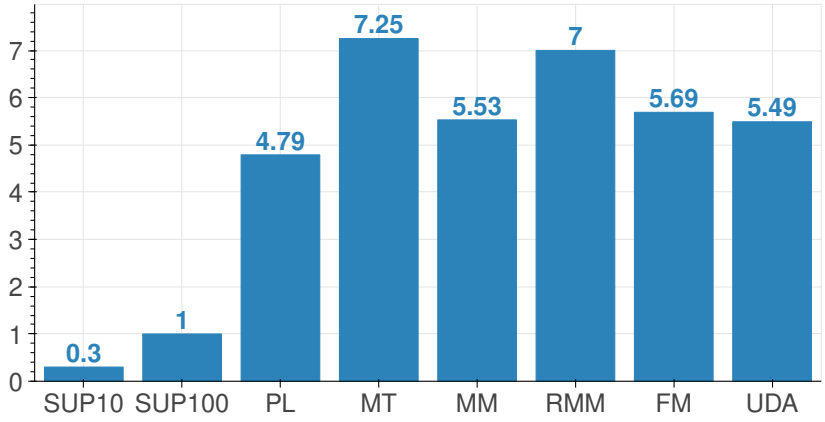


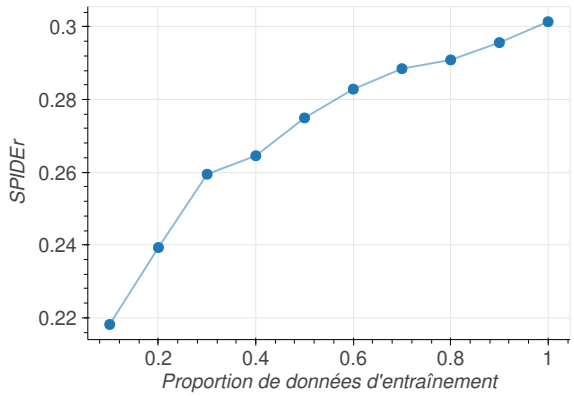
FIGURE 9.11 – Durées d'exécution moyennes normalisées pour les différentes méthodes supervisées et non supervisées. SUP10 et SUP100 font références aux apprentissages supervisés utilisant respectivement 10% et 100% des annotations.

Les méthodes MM, FM et UDA sont très similaires en temps d'exécution, notamment dû au fait qu'elles appliquent le même nombre d'augmentations et donc le même nombre d'appels au modèle. La méthode MT est la plus coûteuse, étant 7,25 fois plus lente qu'un apprentissage supervisé complet et 24,17 fois plus lente que SUP10, principalement causé par l'utilisation de deux modèles qui nécessitent tous deux une passe complète sur les données étiquetées et non étiquetées. Ces résultats plaident en faveur des méthodes MM et FM, qui semblent être à la fois les plus performantes et les plus rapides des méthodes SSL.

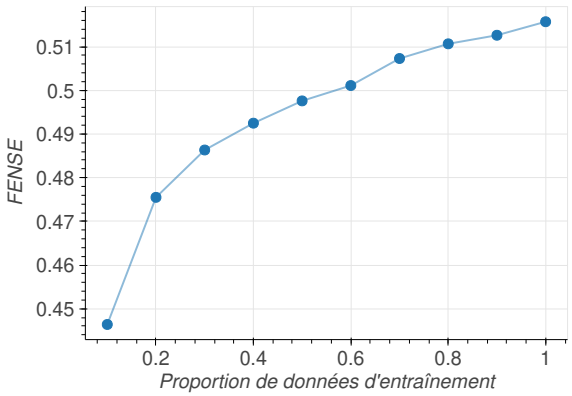
9.5 Apprentissage semi-supervisé pour la DTAA

Maintenant que nous avons trouvé des méthodes efficaces qui fonctionnent dans un contexte mono-étiquette accessible, nous pouvons tenter d'en appliquer une pour la DTAA. Malgré leur popularité, les méthodes d'apprentissage semi-supervisées sont assez peu étudiées dans d'autres contextes que la classification mono-étiquette, et nous n'avons pas trouvé d'exemple pour la tâche de DTAA. Cependant, il existe quelques stratégies dans d'autres modalités comme la DTAI ou la DTAV, mais qui restent assez différentes de celles utilisées en classification. Certaines approches, comme [Jain 2021], sont fondées sur le *Pseudo-Label*, et emploient de nombreuses augmentations sur les images et le texte pour éviter l'effondrement du système entraîné sur ses propres prédictions. D'autres approches reposent sur des méthodes d'échantillonnage [Liu 2018, Lin 2020, Jin 2023], qui demandent au modèle de générer une phrase par une méthode de décodage stochastique, puis de comparer la représentation de cette phrase avec celle de l'audio correspondant et celle d'un autre audio pris aléatoirement dans le lot. Ces représentations sont issues d'un modèle de recherche image-texte pré-entraîné, comme le modèle *Contrastive Language-Image Pretraining* [Radford 2021]. La fonction vise donc à faire générer au modèle des descriptions discriminantes sur les données non étiquetées. D'autres types de stratégies, comme [Kim 2019b], recherchent la description la plus sémantiquement similaire à partir d'une base de données de phrases pour l'utiliser en tant que pseudo-étiquette, mais cette stratégie est assez coûteuse. La dernière catégorie de méthodes [Chen 2017, Anderson 2018] repose sur la détection de concepts visuels ou de mots clés, en utilisant les mots les plus fréquents des phrases ou bien des étiquettes d'un jeu de données de classification d'images pour faire un apprentissage multitâche. Ces dernières approches sont cependant plus proches de méthodes d'apprentissage légèrement supervisé que de méthodes SSL.

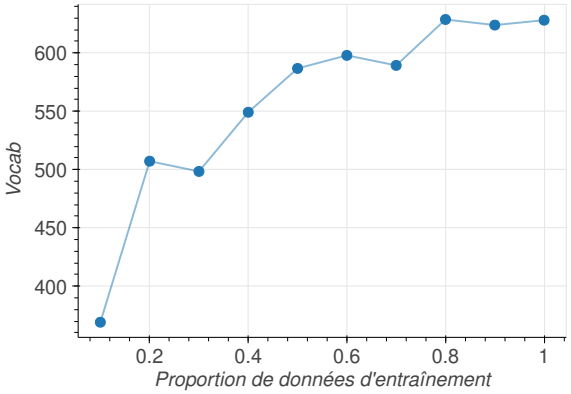
Pour nos expériences, nous réutilisons l'architecture du modèle SR3 entraîné sur CL du chapitre 5. La première étape consiste à tester des méthodes semi-supervisées dans un contexte simplifié, en limitant le nombre de données d'entraînement sur ce jeu de données pour voir si ces méthodes améliorent les performances. Pour cela, nous avons réalisé quelques entraînements sur CL en faisant varier la proportion de données annotées. La figure 9.12 montre les scores du système SR3 en fonction de cette proportion. Nous avons décidé de ne pas utiliser AC car les scores d'un apprentissage SUP10 et SUP100 sont moins éloignés que sur CL. À noter que nous utilisons toujours le même vocabulaire, c'est-à-dire celui de CL-dev dans toutes ces expériences. Comme nous nous y attendions, le système est plus performant sur les trois métriques avec l'augmentation de la quantité de données, passant de 21,8% à 30,1% pour SPIDEr, de 44,6% à 51,6% pour FENSE et de 369,0 à 628,2 mots distincts. Comme pour la tâche de classification, nous nommons respectivement SUP10 et SUP100 les apprentissages supervisés utilisant 10% et 100% des données.



(a) SPIDEr selon la proportion de données utilisée.



(b) FENSE selon la proportion de données utilisée.



(c) Vocab selon la proportion de données utilisée.

FIGURE 9.12 – Performance du modèle selon la proportion de données utilisée par tranche de 10% sur CL-eval.

En premier lieu, nous avons décidé de tester la méthode MT, qui a pour avantage d'être relativement plus simple que les autres méthodes et moins dépendante des augmentations appliquées. Pour tester la méthode, nous divisons le jeu CL-dev en 10% de données annotées et 90% de données non annotées et nous reprenons les mêmes hyperparamètres que pour la classification audio. Cependant, quelques modifications sont nécessaires pour adapter la tâche de classification. La première concerne le choix de la méthode de génération de la pseudo-étiquette, et nous avons choisi l'algorithme glouton (voir section 1.4.5.1) pour garantir un entraînement rapide. Nous avons également testé a posteriori l'algorithme de recherche en faisceau, qui n'apporte pas de meilleurs résultats. La seconde modification concerne la fonction de coût $\mathcal{L}_u$, car MSE peine à converger pour la DTAA, ce qui nous a mené à appliquer la fonction CE. Nous avons également testé deux variantes de la méthode MT. Nous avons décidé de voir si l'initialisation du modèle changeait la convergence de cette méthode, en utilisant ou non les poids du modèle SUP10. La seconde variante concerne l'augmentation de données appliquée aux entrées de l'étudiant. Dans le cas d'origine, nous n'appliquons aucune augmentation, et dans un second cas, nous appliquons un Mixup pour perturber ses entrées. Ce Mixup est appliqué entre les données étiquetées et non-étiquetées, comme pour FM. La méthode MT nous fournit à la fin de l'entraînement deux modèles de DTAA : le professeur (Prof.) et l'étudiant (Étud.), dont nous avons reporté les résultats dans le tableau 9.4.

Table 9.4 – Résultats de la méthode MT naïve sur CL. Les acronymes utilisés sont : Prof. pour le modèle Professeur et Étud pour le modèle Étudiant de l'apprentissage MT.

Système	**Initialisation**	**Mixup α**	**Modèle**	**SD**	**FNS**	**Vocab**
SUP10	Aléatoire	0,4	N/A	21,8±0,4	44,6±0,5	369,0±12,3
SUP100 (SR3)	Aléatoire	0,4	N/A	30,1±0,5	51,6±0,3	628,2±46,2
MT	Aléatoire	0	Prof.	21,3±1,6	40,0±4,8	246,0±83,8
MT	Aléatoire	0,4	Prof.	**22,1±0,8**	41,7±1,8	263,2±65,6
MT	SUP10	0	Prof.	20,6±1,1	44,4±0,3	497,6±73,3
MT	SUP10	0,4	Prof.	21,0±0,7	44,4±0,9	438,0±112,9
MT	Aléatoire	0	Étud.	19,8±0,7	44,4±0,3	484,0±152,2
MT	Aléatoire	0,4	Étud.	20,4±0,5	44,5±0,6	507,4±92,2
MT	SUP10	0	Étud.	18,6±1,2	44,3±0,4	**670,4±97,1**
MT	SUP10	0,4	Étud.	20,1±1,0	**44,9±0,3**	568,4±163,3

Contrairement à ce que nous pourrions attendre, l'introduction de données supplémentaires via la méthode MT n'améliore pas les performances dans la majorité des cas. Le meilleur système en SPIDEr obtient 22,1%, alors que l'apprentissage SUP10 obtenait déjà 21,8%. Le gain est donc très marginal dans ces cas, et la majorité des systèmes baissent légèrement en SPIDEr, FENSE ou Vocab. Commencer l'apprentissage avec les poids du SUP10 permet d'obtenir un meilleur score FENSE et un vocabulaire plus diversifié en général, mais cela n'améliore pas le SPIDEr. De même, le modèle professeur est plus performant que l'étudiant en SPIDEr, mais pas en FENSE ou vocabulaire dans la plupart des cas. Nous avons également noté que la courbe de validation de la fonction de coût est croissante lorsque nous utilisons l'initialisation du modèle SUP10, indiquant que la performance du modèle se dégrade au cours des époques d'entraînement. Nous avons cependant remarqué deux autres éléments pouvant expliquer ce manque d'amélioration de performances, détaillé dans les deux paragraphes ci-dessous.

Notre modèle emploie un encodeur pré-entraîné. Les méthodes SSL permettent d'introduire une plus grande quantité de fichiers non annotés, mais notre encodeur ConvNeXt a déjà été entraîné avec une grande quantité de données audio annotées (2 millions d'audios du jeu de données AudioSet). Cela pourrait expliquer que le fait d'ajouter quelques données non annotées n'ait qu'un impact marginal sur notre modèle, même si leurs événements sonores sont plus proches du domaine de CL.

Un système de DTAA est aussi un modèle de langage. Comme nous l'avons vu dans le chapitre 7, le modèle peut apprendre certains styles de descriptions, et celles des pseudo-étiquettes sont en général moins précises, diversifiées ou informatives que les références humaines. Le modèle n'arriverait donc pas à apprendre plus que ce qu'il sait déjà décrire. Pour pallier ce problème, nous pourrions tenter de rajouter un filtre similaire à celui FM pour masquer les pseudo-étiquettes. Cependant, nous ne pouvons pas nous reposer sur la probabilité donnée par le modèle, car comme nous l'avons vu dans le chapitre 8, ce dernier surestime considérablement la vraisemblance de certaines phrases. Une solution consisterait à créer un second modèle nommé « discriminateur », qui se serait entraîné à reconnaître les bonnes pseudo-étiquettes pour mieux les sélectionner.

Ce modèle pourrait être entraîné comme un modèle de recherche audio-texte, ou entraîné à distinguer les étiquettes et pseudo-étiquettes.

9.6 Bilan

Ce chapitre avait pour objectif d'étudier et de comparer les méthodes semi-supervisées (SSL) utilisées principalement en classification d'images dans le domaine de l'audio, ce qui reste une tâche plus simple que la DTAA. Nous avons testé six méthodes SSL : *Pseudo-Label*, *Mean Teacher*, *MixMatch*, *ReMixMatch*, *FixMatch* et *Unsupervised Data Augmentation* sur trois bases de données différentes : *Environmental Sound Classification*, *UrbanSound8k* et *Google Speech Commands*.

Les méthodes SSL se sont révélées très efficaces, allant jusqu'à s'approcher des meilleurs apprentissages supervisés en exploitant 10% des étiquettes : la différence entre MM et le meilleur supervisé utilisant 100% des données est de seulement 0,06% sur UBS8K et 0,27% sur GSC. Une partie des résultats présentés ici ont été communiqués dans un article de la revue internationale *EURASIP Journal on Audio, Speech, and Music Processing* [Cancès 2022]. De plus, le code source du chargement des données, des modèles et des méthodes SSL est publiquement accessible sous la forme d'une bibliothèque nommée `SSLH` disponible sur GitHub [56].

L'utilisation des augmentations et du Mixup semble être l'une des clés importantes pour obtenir les meilleurs gains, et encourage donc l'application de ces méthodes pour une tâche plus difficile comme la description automatique. Cependant, ces méthodes semblent fonctionner sous plusieurs conditions, comme le fait que les étiquettes soient distribuées de la même manière dans Y_s et Y_u ou que les étiquettes de Y_u existent dans Y_s. Il faut également noter que plusieurs opérations dépendent du caractère mono-étiquette de la tâche (aiguisage, alignement des distributions, somme des étiquettes...) et doivent donc être adaptées pour des phrases.

Pour la DTAA, l'utilisation de pseudo-étiquettes pour améliorer les performances du modèle ne semble pas fonctionner par l'approche MT. En effet, les performances des modèles utilisant des données non annotées supplémentaires sont comparables ou plus faibles. Nous supposons que ce problème vient de la qualité des pseudo-étiquettes générées, qui sont bien souvent moins détaillées que des références humaines. De plus, le manque de diversité des descriptions ne permet pas au modèle d'exploiter de nouveaux mots ou de reconnaître plus finement les événements sonores, surtout en sachant que l'encodeur est déjà pré-entraîné. Cependant, l'emploi d'un autre modèle pour discriminer les bonnes et mauvaises pseudo-étiquettes pourrait au moins régler ce problème, mais nécessiterait une étude plus approfondie.

56. https://github.com/Labbeti/SSLH

Chapitre 10

Description multilingue d'événements sonores

Dans la littérature, les systèmes de DTAA se sont presque toujours focalisés sur la description d'événements sonores en anglais. Pourtant, une application utilisant un système de DTAA pourrait nécessiter des descriptions dans une langue spécifique. Une approche simple consisterait à traduire les candidats anglais produits à l'aide d'un modèle de traduction automatique, mais il parait légitime de se demander si nous ne pourrions pas créer un modèle qui produirait les descriptions dans la langue cible directement.

Dans ce chapitre, nous explorons la DTAA multilingue, en exploitant des annotations manuelles ainsi que des descriptions traduites automatiquement dans d'autres langues que l'anglais. Nous étudions la DTAA en français, allemand et espagnol en créant des systèmes dédiés et entraînés sur des jeux traduits automatiquement. Nous créons également un modèle capable de générer des descriptions dans toutes les langues, et nous le comparons aux systèmes monolingues sur plusieurs jeux de données. Enfin, nous évaluons les systèmes français sur un jeu de test ré-annoté manuellement et nous les confrontons au système anglais dont les sorties ont été traduites.

Sommaire

10.1 Contexte

La DTAA vise à développer des systèmes capables de fournir des descriptions textuelles d'enregistrements audio. Alors que la plupart des jeux de données disponibles offre des descriptions en anglais (`en`), des applications pratiques peuvent demander de générer des textes en plusieurs langues. Notre étude est motivée par quelques questions fondamentales :

1. Les descriptions traduites par une machine peuvent-elles être utilisées efficacement pour construire des systèmes de DTAA dans d'autres langues que l'anglais ?
2. Si l'on cherche à traiter plusieurs langues, doit-on employer des systèmes entraînés séparément pour chacune d'entre elles ?
3. Est-il avantageux de construire de nouveaux systèmes plutôt que de se contenter de traduire les descriptions générées par un système fondé sur l'anglais ?

Pour tenter de répondre à ces questions, nous étudions l'emploi de la traduction automatique pour entraîner et évaluer des systèmes en français (`fr`), allemand (`de`) et espagnol (`es`), en se servant des deux jeux de données AC et CL. Nous développons non seulement des versions monolingues, mais aussi multilingues de nos systèmes. Enfin, nous comparons la qualité des descriptions produites par un système français avec ceux traduits à partir d'un modèle anglais, en utilisant une version française annotée manuellement d'AC-test. Cette version comprend des descriptions rédigées par un annotateur humain dans le but spécifique de notre évaluation.

10.2 Systèmes monolingues et multilingues

Systèmes monolingues. Pour chacune des quatre langues cibles, nous nous sommes appuyés sur notre système de référence SR3, décrit dans le chapitre 5. Le seul changement d'architecture nécessaire pour construire un système pour une nouvelle langue est l'adaptation de la taille des couches de représentations et de classification linéaire pour faire face aux différentes tailles de vocabulaire. Pour rappel, CNext-trans contient 28 millions de paramètres gelés (l'encodeur ConvNeXt) et 12 millions de paramètres entraînables. Cette dernière valeur varie légèrement en fonction de la langue cible. Plus précisément, la couche de classification comprend respectivement 1,2, 1,5 et 2,3 millions de paramètres, pour l'anglais, le français/espagnol et l'allemand. Nous adaptons également la liste prédéfinie de *stopwords* par langue, car elle est utilisée pour éviter de répéter certains mots durant la validation et l'inférence. Par souci d'homogénéisation des expériences, nous avons recours à un *tokenizer* différent de celui employé pour SR3, qui supporte plusieurs langues en même temps. Plus précisément, nous remplaçons le *tokenizer* `en_core_web_sm` par `xx_ent_wiki_sm`. Ceci explique notamment quelques différences entre les statistiques des données d'AC-test en anglais avec ceux donnés dans le chapitre 2, ainsi que la légère différence de performance par rapport au système SR3 (par ailleurs, cette dernière n'est pas significative).

Système multilingue. Nous avons construit le système multilingue en nous fondant sur le système monolingue, en ajoutant des couches de représentation et de classification spécifiques à chaque langue. Tous les autres éléments du modèle sont restés identiques, ce qui signifie que le modèle partage le même encodeur et une majeure partie des poids du décodeur dans toutes les langues. Seuls la couche de représentation d'entrée des mots et la couche de classification du décodeur sont spécifiques à chaque langue, comme montré dans le schéma 10.1. Comparé à un système monolingue, qui a une taille d'environ 40 millions de paramètres, le modèle multilingue contient 50,8 millions de paramètres (dont 22,8 millions de paramètres entraînables). Si nous

devions utiliser une collection de quatre systèmes monolingues, la taille totale de ces systèmes serait de 77 millions de paramètres en partageant le même encodeur (50 millions de paramètres entraînables). Le système multilingue permet donc de réduire la taille du modèle d'environ 35%. La question principale de ce travail est de comparer la qualité des descriptions générées en choisissant l'une des deux options. Pour les systèmes multilingues, une époque d'entraînement consiste à voir chaque fichier audio pour chacune des langues, pour permettre au modèle de voir les quatre langues en une seule époque.

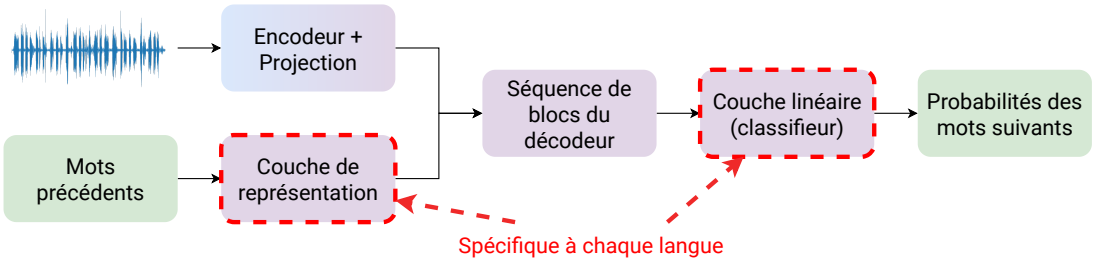


FIGURE 10.1 – Schéma du système multilingue. Les cases entourées de **rouge** désignent les parties spécifiques à chaque langue.

Pour évaluer et comparer les modèles, nous employons la métrique CIDEr-D [Vedantam 2015]. Nous indiquons également les valeurs de la métrique SB_{sim} entre les candidats et références pour chaque langue, en utilisant le modèle SBERT multilingue paraphrase-multilingual-mpnet-base-v2, car il a été entraîné avec plusieurs langues et permet donc de comparer directement les descriptions. Nous ne pouvons pas nous appuyer sur les métriques SPICE ou SPIDEr (qui, pour rappel, est la moyenne de CIDEr-D et SPICE), car SPICE emploie un analyseur syntaxique, une grammaire élaborée à la main et des règles personnalisées qui sont disponibles uniquement pour la langue anglaise à notre connaissance. L'adaptation du système de référence SR3 pour chaque langue et la version multilingue ont été conçus lors de son stage de trois mois sur ce sujet par Matéo Cousin, que nous avons supervisé Thomas Pellegrini et moi.

10.3 Données traduites

Versions traduites automatiquement. AC et CL contiennent des descriptions en anglais uniquement. Afin de traduire les descriptions des deux ensembles de données en français, en allemand et en espagnol, nous avons testé deux outils open-source récents basés sur l'apprentissage profond : Opus-MT [Tiedemann 2020] et *No Language Left Behind* (NLLB) [Costa-jussà 2022]. Une troisième option a été envisagée : le service payant proposé par *DeepL translation API*[57]. Nous avons comparé leurs traductions manuellement sur un sous-ensemble de phrases d'AC. Un certain nombre de descriptions dans AC contient des fautes d'orthographe, des onomatopées ainsi que des mots peu courants, et nous avons observé qu'Opus-MT et NLLB ne parvenaient pas à fournir des résultats satisfaisants dans ces cas-là. Deux exemples sont donnés dans le tableau 10.1, où seul DeepL fournit une traduction française réellement satisfaisante. Compte tenu de nos observations et de certaines références publiées dans la littérature [Isabelle 2018, Yulianto 2021], nous avons donc opté pour l'utilisation du traducteur DeepL.

57. https://www.deepl.com/translator

TABLE 10.1 – Deux exemples de traductions de l'anglais au français.

Modèle de traduction	**Description**
Aucun	*The sizzling of food while a dish is clanking*
Opus-MT	L'étourdissement de la nourriture pendant qu'un plat claque
NLLB	Le sifflement de la nourriture pendant qu'un plat clinche
DeepL	Le grésillement des aliments pendant qu'un plat s'entrechoque
Aucun	*Humming and vibrating of a motor*
Opus-MT	Humming et vibration d'un moteur
NLLB	Le zume et la vibration d'un moteur
DeepL	Ronflement et vibration d'un moteur

Version traduite manuellement. Nous avons créé un ensemble manuel de descriptions, en français, spécifiquement pour le sous-ensemble de test d'AC et nommé AC-test-fr-manuel. Pour ce faire, nous avons fait appel à une société spécialisée dans l'annotation de données. La conception de l'interface d'annotation a été réalisée sous Prodigy par Adrian Benard lors d'un stage de deux mois. Un annotateur s'est chargé de la rédaction de la totalité des phrases du sous-ensemble. Nous avons suivi la même méthodologie d'annotation que pour AC : l'annotateur avait accès à la piste audio pour écrire une description. Dans le cas où il n'était pas sûr des événements sonores qu'il percevait, il pouvait regarder la vidéo correspondante pour éventuellement éditer sa description. En raison de contraintes budgétaires, nous avons limité le nombre de descriptions manuelles à une par fichier audio, au lieu de cinq dans l'ensemble de données original. Pour donner un ordre de grandeur, le coût total de la traduction automatique d'AC et CL dans trois langues cibles s'est élevé à environ 300 euros, alors que l'annotation manuelle en français du sous-ensemble test d'AC seul (avec une phrase par audio) a coûté approximativement 3 500 euros.

Dans le tableau 10.2, nous indiquons la longueur moyenne des phrases et le nombre de mots uniques dans les sous-ensembles d'apprentissage et de test dans les quatre langues. Comme nous nous y attendions, les descriptions allemandes sont plus courtes (8,5 et 10,1 mots par phrase), mais contiennent beaucoup plus de mots uniques (9391 et 2792 mots). Les phrases françaises sont les plus longues (10,4 et 12,7 mots par phrase) et le vocabulaire anglais est le plus restreint (4861 mots). En ce qui concerne AC-test-fr-manuel, les descriptions sont plus courtes que les traductions : 11,3 mots en moyenne contre 12,7. Le sous-ensemble contient 927 mots uniques, ce qui semble beaucoup plus petit que le sous-ensemble traduit (1973 types de mots). Cette différence s'explique par le fait qu'il n'y ait qu'une seule description par fichier audio dans le jeu manuel au lieu de cinq dans le jeu traduit.

10.4 Résultats sur les traductions automatiques

Résultats en. Le tableau 10.3 présente les valeurs CIDEr-D et SB_{sim} sur AC-test et CL-eval, obtenues par des systèmes monolingues et multilingues en anglais uniquement. Sur AC, notre système monolingue anglais a obtenu 76,1% de CIDEr-D, ce qui est légèrement inférieur à l'état de l'art qui n'exploite pas de données d'entraînement externes Multi-TTA [Kim 2022]. Notre système multilingue a légèrement dépassé Multi-TTA avec 77,4% de CIDEr-D. En ce qui concerne

TABLE 10.2 – Taille des phrases et du vocabulaire pour les ensembles d'entraînements et de test pour AC et CL. Entr. est le diminutif pour Entraînement.

Données	Langue	Annotation	#Sent		Vocab	
			Entr.	Test	Entr.	Test
AC	en	humaine	8,7	10,3	4861	1623
	fr	automatique	10,4	12,7	5797	1973
	es	automatique	9,5	11,6	5889	1976
	de	automatique	8,5	10,1	9391	2792
	fr	humaine	N/A	11,3	N/A	927
CL	en	humaine	11,3	11,3	4366	3517
	fr	automatique	12,9	12,8	6384	3942
	es	automatique	11,6	11,6	6635	3962
	de	automatique	10,6	10,6	9537	5069

CL, nos systèmes monolingues et multilingues ont tous deux obtenu de meilleurs résultats que l'état de l'art sans données externes CNN14-trans† [Won 2021], comme pour le modèle SR3. Contrairement à AC, le modèle multilingue semble légèrement moins performant que le modèle monolingue anglais pour CL.

TABLE 10.3 – Résultats de systèmes monolingues et multilingues anglais (en) (en %), avec les métriques CD et SB_{sim} sur AC-test et CL-eval. La meilleure valeur par métrique est en **gras**.

Test	Système	Lang.	Type	Params. (M, ↓)	CD (%, ↑)	SB_{sim} (%, ↑)
AC	HTSAT-BART [Mei 2023]	en	mono	171	**78,7**	N/A
	Multi-TTA [Kim 2022]	en	mono	108	76,9	N/A
	CNext-trans (nous)	en	mono	**40**	76,1±1,9	72,2±0,2
		en	multi	51	77,4±2,2	**72,5**±0,1
CL	BEATs-BART [Wu 2023b]	en	mono	1600	**50,6**	N/A
	CNN14-trans† [Won 2021]	en	mono	88	44,1	N/A
	CNext-trans (nous)	en	mono	**40**	46,6±1,0	**58,8**±0,3
		en	multi	**40**	45,6±0,7	58,5±0,4

Résultats fr, es, de. En ce qui concerne nos résultats dans les quatre langues, les valeurs sont reportées dans le tableau 10.4. Si nous comparons les langues entre elles, les scores allemands sont plus faibles à cause de la plus grande diversité de vocabulaire, et les scores français sont plus élevé grâces aux plus longues phrases de références. Sur AC, le modèle multilingue est légèrement meilleur que le modèle monolingue, avec un gain moyen +1,0% de CIDEr-D et 0,2% de SB_{sim} sur les quatre langues. Sur CL, c'est le contraire, le système monolingue est meilleur, avec une différence de 1,4% de CIDEr-D et de 0,2% de SB_{sim}. Les scores des types de systèmes restent

TABLE 10.4 – Résultats du CNext-trans, en version monolingue et multilingue sur AC-test et CL-eval (en %). Les métriques sont CIDEr-D (CD) et SB_{sim}, avec trois différentes langues : français (`fr`), allemand (`de`) et espagnol (`es`). La meilleure valeur par métrique, langue et jeu de données est en **gras**.

Test	Langue	Type	Params. (M, ↓)	CD (%, ↑)	SB_{sim} (%, ↑)
AC	`fr`	mono	41	79,9±0,7	75,0±0,1
	`fr`	multi	51	**82,0**±1,2	**75,1**±0,1
	`es`	mono	41	**75,0**±0,8	**73,2**±0,1
	`es`	multi	51	74,1±1,5	73,1±0,1
	`de`	mono	42	68,4±1,0	73,5±0,3
	`de`	multi	51	**69,9**±0,4	**73,8**±0,1
	Moy.	mono	41	74,8±1,1	73,4±0,2
	Moy.	multi	51	**75,8**±1,3	**73,6**±0,1
CL	`fr`	mono	41	**41,8**±0,6	**63,1**±0,1
	`fr`	multi	41	41,3±1,2	63,1±0,2
	`es`	mono	41	**47,3**±0,8	**61,1**±0,2
	`es`	multi	41	44,6±1,4	60,6±0,4
	`de`	mono	43	**39,3**±0,3	64,0±0,4
	`de`	multi	43	38,0±1,0	**64,1**±0,3
	Moy.	mono	41	**43,8**±0,7	**61,8**±0,3
	Moy.	multi	41	42,4±1,1	61,6±0,3

assez proches, ce qui semble confirmer que les descriptions de DTAA de plusieurs langues sont peu diversifiées par rapport au langage naturel et peuvent donc être appris par un seul modèle. Les tableaux 10.5 et 10.6 en annexe illustrent les résultats obtenus par notre système multilingue pour deux fichiers audio d'AC-test [58] [59]. Le premier exemple montre que les descriptions peuvent parfois décrire le même événement (la pluie dans les candidats pour le premier tableau), ou au contraire plusieurs événements différents (un canard pour `en` et `es`, un chien pour `fr` et `de`). Nous donnons également les références correspondantes dans l'annexe J.

Pour déterminer le degré de similitude des descriptions produites dans les différentes langues, nous avons calculé les valeurs SB_{sim} entre elles, car le modèle de texte pré-entraîné peut directement comparer des paires de phrases de langues différentes. Avec le système monolingue, les valeurs moyennes de similarité entre les candidats du système anglais et celles des systèmes français, espagnol et allemand étaient respectivement de 77,7% et 69,4% sur AC et CL. Avec le système multilingue, nous comparons ses sorties anglaises à ses sorties françaises, espagnoles et allemandes. Nous avons obtenu des valeurs moyennes légèrement plus élevées de 78,6% et 71,3% en AC et CL, ce qui semble montrer que le système multilingue tend à produire des descriptions plus

58. https://www.youtube.com/embed/Pb6MqpdX5Jw?start=40&end=50
59. https://www.youtube.com/embed/JTHMXLC9YRs?start=30&end=40

TABLE 10.5 – Candidats de la version multilingue du CNext-trans en quatre langues différentes pour le fichier audio « `Pb6MqpdX5Jw` », issus d'AC-test.

Langue	Candidats
`en`	Rain falls and thunder roars in the distance.
`fr`	La pluie tombe et le tonnerre gronde au loin.
`es`	La lluvia cae con fuerza y los truenos retumban a lo lejos.
`de`	Regen fällt und der donner grollt.

TABLE 10.6 – Candidats de la version multilingue du CNext-trans en quatre langues différentes pour le fichier audio « `JTHMXLC9YRs` », issus d'AC-test.

Langue	Candidats
`en`	A duck quacking.
`fr`	Un chien halète et respire.
`es`	Los patos graznan y el viento sopla.
`de`	Ein hund hechelt und atmet schwer.

similaires d'une langue à l'autre que dans le cas des modèles monolingues. Cela peut s'expliquer par le partage des poids du décodeur intrinsèque au système multilingue, mais une étude plus approfondie pourrait être nécessaire pour examiner ce point.

10.5 Traduction des sorties du modèle

Maintenant que nous avons construit plusieurs modèles dans des langues différentes, nous pouvons comparer leurs performances avec un système anglais dont les sorties sont traduites automatiquement. Le tableau 10.7 présente les valeurs CIDEr-D sur AC-test-fr-manuel, en utilisant les candidats d'un système français, les candidats d'un système anglais traduits automatiquement et les candidats français d'un système multilingue. Les systèmes monolingues, et plus encore les systèmes multilingues français, obtiennent des résultats nettement supérieurs à ceux du système anglais, avec respectivement 93,2% et 95,8% comparés à 81,1% de CIDEr-D. Surprenamment, cela indique qu'il est plus intéressant d'entraîner un modèle sur une langue cible plutôt que de traduire les sorties d'un système anglais vers cette langue cible, et ce même si le modèle est entraîné sur des traductions automatiques. Pour valider cette hypothèse, il serait nécessaire de la tester avec d'autres langues, ainsi que d'autres jeux de données annotés manuellement. De plus, la qualité du traducteur automatique utilisé pourrait avoir un impact important sur ces descriptions et donc sur la performance du modèle.

10.6 Bilan

À notre connaissance, ce travail est le premier à explorer la DTAA multilingue. Nous démontrons la viabilité de la traduction automatique des descriptions anglaises en français, allemand et espagnol, sur AC et CL en créant des systèmes monolingues dédiés à chaque langue. De plus, nous avons présenté une architecture multilingue capable de générer des descriptions dans les

TABLE 10.7 – Résultats de CIDEr-D (CD) du modèle CNext-trans (en %), sur le sous-ensemble AC-test-fr-manuel.

Système	**Langue(s) d'entraînement**	**Candidats**	**CD (%, ↑)**
Monolingue	`fr`	`fr`	93,2±3,9
Multilingue	`en+fr+es+de`	`fr`	**95,8**±2,6
Monolingue	`en`	traduits vers `fr`	81,1±4,0

quatre langues, et pouvant atteindre une performance comparable aux systèmes monolingues, avec 75% de CIDEr-D sur AC et 43% sur CL. Enfin, nous avons ré-annoté le sous-ensemble de test du jeu AC manuellement en français. Les systèmes français, entraînés sur la version traduite française d'AC, ont obtenu de bien meilleurs résultats sur le sous-ensemble d'évaluation manuel, comparé au système anglais pour lequel nous avons traduit automatiquement les sorties vers le français. Surprenamment, ces résultats soutiennent l'adoption d'un système entraîné dans la langue cible plutôt que de simplement traduire les candidats d'un système anglais. Nous avons compilé une portion des résultats et des conclusions de ce chapitre dans une version prépubliée d'un article disponible sur arXiv [Cousin 2023].

Pour poursuivre la recherche autour de plusieurs langues, il est nécessaire de collecter plus de données de chacune d'entre elles, notamment pour tester et confirmer de manière plus robuste le besoin de créer des modèles dédiés à chaque langue plutôt que de traduire les sorties. Il serait également pertinent de confirmer notre approche avec d'autres systèmes de traduction automatique que DeepL, mais aussi de valider pour d'autres styles de descriptions plus riches et détaillées. De plus, nous pourrions étudier des langues plus éloignées que les 4 étudiées dans ce chapitre, comme le russe ou le chinois. À noter qu'une version russe d'AC a été récemment publiée sur HuggingFace [60] après nos travaux, mais il ne semble pas y avoir de détail sur le processus de traduction employé. Nous participerons également à l'organisation de la tâche de DTAA, dans laquelle nous proposons d'étendre la tâche avec des descriptions en français et en chinois. Pour aller encore plus loin, la capacité du modèle à générer des phrases dans plusieurs langues pourrait être combiné avec le TE de CoNeTTE, pour former un système plus généraliste et capable de fournir une grande variété de phrases par audio, par langue et style de description.

60. https://huggingface.co/datasets/d0rj/audiocaps-ru

Conclusions

Cette thèse se place dans le contexte de la Description Textuelle Automatique de l'Audio (DTAA). Elle a pour objectif de créer un système pouvant décrire n'importe quel fichier audio à l'aide d'un texte écrit en langage naturel. Ce type de système peut permettre d'aider les malentendants, de faire des recherches dans des bases de données audio, ou encore d'effectuer de la surveillance via la détection d'événements sonores. La DTAA reste une tâche encore jeune, bien qu'elle ait évoluée rapidement durant ces dernières années. Cette thèse s'est concentrée sur plusieurs aspects différents : l'entraînement, l'évaluation, ainsi que l'utilisation de différents systèmes de DTAA dans des contextes différents. Dans ce chapitre, nous présentons un résumé des travaux et des conclusions de ce manuscrit, puis nous terminons en proposant un ensemble de pistes et de perspectives pour de potentielles recherches futures.

Synthèse des travaux

La première partie de ce manuscrit a servi à présenter un état de l'art global de la DTAA. La tâche faisant intervenir des relations complexes ainsi que de grande quantité de données, la totalité des méthodes repose sur des méthodes d'apprentissage automatique utilisant des réseaux de neurones profonds.

Nous avons introduit dans le premier chapitre les types de couches neuronales utilisés pour cette tâche, et présenté les divers jeux de données publics disponibles. Une différence notable entre ces jeux réside dans le processus d'annotation, qui engendre des phrases plus ou moins diversifiées, correctes ou précises. Cela impacte fortement les descriptions produites par les systèmes automatiques, qui dépendent fortement de ces données. Ensuite, nous avons détaillé les méthodes d'apprentissages employées pour la DTAA. Ces dernières se sont d'abord inspirées de la Reconnaissance Automatique de la Parole (RAP), en utilisant des architectures de type encodeur-décodeur assez similaires. En raison du manque de données annotées, la plupart des approches de DTAA emploient de l'apprentissage par transfert ou de l'extraction de représentations. Une partie des modèles de DTAA reposent donc sur des poids issus de modèles pré-entraînés pour la classification audio (AT). Une autre stratégie, pour pallier le manque de données, est l'augmentation. Celle-ci peut s'appliquer à l'audio ou au texte, mais elle est en général difficile à appliquer dans le contexte de la DTAA. Enfin, la dernière approche de la littérature consiste à entraîner les systèmes sur plusieurs tâches, comme pour prédire des mots clés de l'audio ou contraindre une représentation intermédiaire du modèle. Cela permet de limiter le surapprentissage des modèles, et d'améliorer leur convergence durant l'apprentissage.

Ensuite, le chapitre 2 se focalise sur l'évaluation des systèmes de DTAA. En effet, une métrique simple comme le taux d'erreur mots pour comparer les phrases ne peut pas être utilisée ici, car les systèmes doivent décrire l'audio sans employer nécessairement les mêmes mots ou la même structure de phrase. Il existe de très nombreuses manières de comparer des phrases, qui prennent en compte parfois les synonymes des mots, la fréquence des n-grammes, ou encore des représentations issues de modèles de texte pré-entraînés. Certaines métriques se focalisent sur les

événements sonores décrits, tandis que d'autres cherchent à pénaliser les phrases mal construites en détectant les erreurs de fluence. La question du choix des métriques d'évaluation reste donc encore très active (quatre nouvelles métriques en 2023). Cependant, nous constatons que la DTAA ne vise pas encore d'application précise, ce qui rend le choix d'une métrique spécifique assez difficile et nébuleux (cherche-t-on réellement à repérer tous les événements sonores ? à produire une phrase correcte ou diversifiée ? à donner un maximum de détails ?).

Enfin, nous avons présenté notre propre système de DTAA, inspiré de la littérature de l'époque dans le chapitre 3. Le système emploie une architecture de type encodeur-décodeur, utilisant le modèle pré-entraîné CNN10 et un décodeur de type *Transformer*. Ce système obtient des performances acceptables et comparables à d'autres, tout en restant relativement simple à entraîner par rapport à certains autres. La création de ce système a donné lieu à deux bibliothèques publiquement accessibles : `aac-datasets` et `aac-metrics`, qui permettent de facilement manipuler les principaux jeux de données et les métriques de DTAA.

Dans la seconde et troisième partie, nous avons exploré de nombreux aspects de la DTAA, pour tenter de répondre aux quatre questions posées dans l'introduction de ce manuscrit.

La méthode d'évaluation actuelle des systèmes est-elle suffisante ? Comment la métrique principale se comporte-t-elle lorsqu'elle est soumise à plusieurs phrases similaires ?

Pour essayer de répondre à cette question, nous avons exploré dans le chapitre 4 la principale métrique nommée SPIDEr. Pour mieux comprendre comment elle se comporte, nous avons introduit SPIDEr-max, qui calcule le maximum des scores SPIDEr de plusieurs candidats par fichier audio. Dans notre cas, nous employons l'algorithme de recherche en faisceau pour générer plusieurs candidats différents. D'une manière surprenante, le score SPIDEr-max peut largement surpasser le score SPIDEr de l'état de l'art, ce qui signifie que le choix de la meilleure phrase de la recherche en faisceau peut provoquer d'énormes différences dans les scores SPIDEr. Plus précisément, le score SPIDEr-max atteint 52,8% sur AC avec seulement trois phrases, alors que l'état de l'art de l'époque atteignait 47,5%. Malgré tout, la plupart des phrases générées par fichier audio sont assez similaires, indiquant que la métrique peut facilement sous-estimer ou surestimer certaines prédictions. Nous avons également mené cette étude avec la métrique FENSE, qui semble moins sensible que SPIDEr aux mots prédits. Nous pensons donc que la métrique SPIDEr n'est pas infaillible, et ne prend pas en compte tous les aspects possibles pour l'évaluation des systèmes de DTAA. Nous avons donc choisi de conserver trois métriques dans la thèse : le SPIDEr pour comparer nos systèmes avec la littérature, le FENSE qui est moins affecté par les n-grammes communs que SPIDEr et qui intègre un détecteur d'erreur de fluence, et enfin le nombre de mots uniques pour estimer la diversité d'un système de DTAA. Certaines métriques publiées après nos travaux pourraient également être ajoutées, comme SPICE+ qui permet une meilleure interprétabilité des scores au travers des graphes de scènes ou comme SBF qui se focalise sur les événements sonores prédits par le système.

Quelles sont les meilleures architectures et méthodes d'apprentissage pour construire un modèle de DTAA ? Comment créer un système efficace en dépit des limites des jeux de données disponibles ?

Par la suite, nous avons décidé de nous focaliser sur le développement d'un système de référence plus performant, en améliorant plusieurs composants de notre système. Premièrement, nous

avons modifié certains aspects de l'algorithme de recherche en faisceau utilisé pour générer des phrases, en évitant certaines phrases vides ou répétitives. Puis, nous avons tenté une approche multitâche dans le chapitre 6 pour améliorer la capacité de notre modèle à prédire des phrases sémantiquement proches des références, sans forcer les mêmes mots. Pour cela, nous avons réutilisé un modèle pré-entraîné sur de larges corpus de textes. Nous avons obtenu un léger gain de performance, qui est causé par une réduction du surapprentissage de notre modèle. Nous avons donc décidé d'introduire une méthode de régularisation plus forte, en modifiant notamment l'hyperparamètre de pénalité des poids de l'optimiseur AdamW. Cette modification améliore grandement les performances en limitant fortement le surapprentissage. Mais une fois combinée avec la méthode multitâche, cette dernière ne permet plus d'obtenir de gain significatif, ce qui indique que notre modèle n'a pas réussi à tirer parti du modèle de texte pré-entraîné.

Dans le chapitre 5, nous avons amélioré la partie encodeur de notre modèle, en remplaçant le CNN10 par notre propre modèle nommé ConvNeXt (CNext). Ce dernier atteint 0,471 de mAP pour la classification audio sur AS, surpassant le CNN14D de PANN qui obtenait 0,425. Ce gain de performance se retrouve également en DTAA sur AC, augmentant de 1,7% le SPIDEr et le FENSE en passant du CNN14D-trans au CNext-trans. Cela confirme que l'emploi d'un encodeur audio pré-entraîné impacte fortement la performance du système de DTAA. Nous avons également recherché de meilleures augmentations, et nous avons trouvé une variante du Mixup qui permet d'améliorer les performances de 1,7% en SPIDEr, mais seulement de 0,2% de FENSE. Pour faciliter la convergence de notre système, nous avons corrigé une partie des descriptions de l'entraînement du jeu AC, qui contenaient un certain nombre d'erreurs d'annotation. Ainsi, nous avons atteint 49,5% de SPIDEr, surpassant l'état de l'art précédent de 1% avec un modèle quatre fois plus petit en nombre de paramètres (40 par rapport à 171 millions).

Pour aller encore plus loin, nous avons mené une étude dans le chapitre 9 des méthodes d'apprentissage semi-supervisées (SSL). Ces dernières permettent d'exploiter des données annotées conjointement avec des non-annotées. Nous avons tout d'abord modifié et appliqué ces méthodes à la classification mono-étiquette de l'audio sur différents jeux de données : ESC-10, UBS8K et GSC. Nous avons obtenu de très bons résultats avec seulement 10% de données annotées, s'approchant souvent des apprentissages totalement supervisés qui exploitent 100% des annotations. Par exemple, la meilleure méthode SSL sur UBS8K obtient 18,02% de taux d'erreur avec 10% de données annotées, ce qui est très proche de la meilleure méthode utilisant 100% des données annotées qui obtient 17,96%. Nous avons alors tenté d'appliquer l'une de ces méthodes pour la DTAA, mais sans succès. En effet, les annotations de DTAA sont plus complexes que de simples classes mono-étiquettes, et notre modèle ne semble pas pouvoir apprendre avec les pseudo-étiquettes peu diversifiées générées. De plus, notre modèle exploite déjà un encodeur audio pré-entraîné, ce qui signifie qu'il a peu à apprendre des nouveaux fichiers audio non annotés introduits par les méthodes SSL.

Comment pouvons-nous tirer parti de plusieurs jeux de données pour créer un système généraliste et performant ?

Pour explorer au mieux à cette question, nous avons cherché à créer un système généraliste et performant en exploitant le maximum de jeux de données de DTAA disponibles dans le chapitre 7. En premier lieu, nous avons remarqué une fuite de données entre les jeux AC et AS, qui pouvait nous mener à surestimer nos systèmes de DTAA à cause du transfert des poids pré-entraînés. Nous avons donc corrigé cette fuite en proposant un encodeur qui évite tout

recouvrement avec les données de DTAA, ce qui diminue les performances d'approximativement 3% de SPIDEr pour le modèle SR3 sur AC. Ce biais n'étant pas présent pour CL, il n'a que peu d'impact sur les performances. Puis, en étudiant les jeux de données de DTAA disponibles, nous avons constaté une différence de domaine entre eux. En effet, un modèle performant sur un ensemble d'évaluation ne l'est pas sur ceux des autres jeux. Pour résoudre ce problème, nous avons introduit une représentation de la tâche, qui indique à notre modèle la source d'où est issue la paire audio-texte. Cette tâche permet au modèle d'apprendre des styles de descriptions différents, qui correspondent à chaque source de données. Le système final est nommé CoNeTTE, et peut générer des phrases imitant les styles de descriptions des différents jeux de données. Ainsi, il obtient une performance sur plusieurs jeux de test avec un SPIDEr de 44,1% sur AC et de 30,5% sur CL. De plus, nous avons mesuré le score SPIDEr-max de ce modèle selon plusieurs tailles de faisceaux, dont nous reportons le détail dans l'annexe K. Comme pour SR1, les scores SPIDEr-max croient rapidement, et dépassent l'état de l'art avec seulement deux candidats sur AC et seulement trois sur CL.

Pouvons-nous étendre ce travail à d'autres tâches et d'autres domaines ?

Premièrement, nous nous sommes intéressés dans le chapitre 8 à la petite sœur de la DTAA, la Recherche Audio-Texte (RAT). Cette dernière vise à rechercher des éléments audio ou texte dans une base de données à partir d'une requête exprimée dans l'autre modalité. Nous avons proposé une méthode simple reposant sur l'entropie croisée de notre système de DTAA, pour évaluer n'importe quelle paire audio-texte. Cette méthode ne demande aucun entraînement supplémentaire, et permet de réaliser les deux sous-tâches de RAT : le texte-vers-audio (T2A) et l'audio-vers-texte (A2T). Étonnamment, un modèle de DTAA peut aisément atteindre des scores comparables, voire supérieurs aux modèles dédiés à la tâche de texte-vers-audio. Par exemple, notre système obtient 0,382 de R@1 sur AC alors que l'état de l'art sans données externes est à 0,368, avec un modèle quatre fois plus petit en paramètres (40 au lieu de 185 millions) pour la T2A. Néanmoins, il obtient des performances bien inférieures pour la tâche inverse (nous obtenons 0,146 par rapport à 0,431 de R@1 sur AC). Nous avons trouvé que cela était dû au modèle de langage, qui surestimait le score de certaines descriptions. Une simple stratégie de normalisation min-max a permis de corriger ce défaut, et ainsi d'atteindre des scores proches des autres systèmes dédiés à l'A2T. Enfin, nous avons analysé notre système pour savoir s'il pouvait discriminer les relations de séquence et de superposition entre les événements sonores. Comme notre système intègre un modèle de langage, il semble être beaucoup plus efficace que les systèmes de RAT pour reconnaître la phrase comprenant la bonne relation temporelle entre les événements.

Pour terminer, nous avons décidé de nous focaliser sur une nouvelle piste de recherche de la DTAA, en étudiant le cas des langues différentes de l'anglais dans le chapitre 10. Pour cela, nous avons traduit automatiquement les jeux de données disponibles dans trois langues (espagnol, français et allemand) et entraîné des modèles dessus. Nous avons également annoté manuellement un sous-ensemble de test en français pour mieux valider les systèmes de cette langue. Sur ce jeu, le modèle entraîné sur les données traduites en français est plus performant que le modèle anglais dont on traduit les sorties vers le français, avec 95,8% par rapport à 81,1% de CIDEr-D, ce qui soutient l'apprentissage de système dédié à chaque langue. Pour aller plus loin, nous avons également créé un système multilingue unique capable de générer des descriptions dans toutes les langues, et qui obtient des performances comparables aux systèmes monolingues.

Participations aux compétitions DCASE

Durant cette thèse, j'ai eu l'occasion de participer à trois reprises aux compétitions DCASE dédiées à la DTAA (tâche 6a, 2021, 2022, 2023), ainsi qu'une fois à la compétition dédiée à la RAT (tâche 6b, 2023). L'évolution des systèmes décrits dans ce manuscrit se retrouve également dans les scores des systèmes donnés dans le tableau 1, pour le jeu de données CL-test (utilisé pour classer les soumissions). Les systèmes de DTAA proposés ont été grandement améliorés tout au long de la thèse, notamment en 2023. Nous pouvons constater que les rangs que nous avons obtenus sont très différents en fonction des années. En 2021, le modèle soumis, proche de SR1, obtenait la 8^{e} place sur 13 participants, alors que le modèle proche de SR2 avait un rang moins bon en 2022, en raison de l'amélioration importante des systèmes soumis par les autres participants. En 2023 cependant, l'introduction de la pénalité des poids, de l'encodeur ConvNeXt et de la représentation de la tâche, ont permis d'obtenir un excellent classement en 3^{e} position. Notre système est bien plus léger que ceux des autres participants qui ont obtenu des scores légèrement supérieurs (1 à 2% de gain de SPIDEr pour des modèles de plusieurs milliards de paramètres). En outre, ce même système a été également utilisé pour participer à la tâche de RAT), sans modification ni entraînement supplémentaire, et a obtenu la 7^{e} place.

Table 1 – Résultats sur les sous-ensembles de test aux compétitions DCASE au fil des années.

Tâche DCASE	**Rapport technique**	**Rang**	**Score**	**Système le plus proche dans le manuscrit**
DTAA	[Labbé 2021]	8/13	22,1	SR1
	[Labbé 2022]	9/10	24,1	SR2
	[Labbé 2023b]	3/11	31,4	CoNeTTE
RAT	[Labbé 2023b]	7/10	23,4	CoNeTTE

Bilan carbone

Pour mener à bien les recherches de cette thèse, de très nombreux expériences et apprentissages ont été menés sur la DTAA. Pour compléter le bilan scientifique, nous ajoutons ici une estimation des émissions de carbone engendrée par l'entraînement des modèles tout au long de ma thèse. Pour cela, nous avons calculé la durée totale des entraînements réalisés sur les quatre machines utilisées : Osirim [61], Jean-Zay [62], Olympe [63] et ma machine locale nommée Araigne. Puis, nous avons utilisé l'outil *Machine Learning Emissions Calculator* [64] présenté dans l'étude [Lacoste 2019] pour estimer un équivalent carbone émis (avec la valeur d'efficacité par défaut de 0,432 kg/kWh). Les résultats sont reportés dans le tableau 2. Ils prennent également en compte les apprentissages qui ne se sont pas terminés à cause d'erreurs ou d'annulation. Cependant, ils n'incluent pas certains entraînements réalisés en local sur des *notebooks* ainsi que plusieurs pré/post-traitements, qui restent minoritaires par rapport aux calculs effectués sur GPU durant cette thèse.

61. https://osirim.irit.fr/
62. http://www.idris.fr/jean-zay/jean-zay-presentation.html
63. https://www.calmip.univ-toulouse.fr/
64. https://mlco2.github.io/impact#compute

TABLE 2 – Statistiques des coûts d'entraînements par machine. L'équivalent CO_2 est estimé en fonction du nombre d'heures et du type de GPU.

Nom	**GPU**	**Nb. jobs**	**Heures**	**Éq. CO_2 (kg)**
Osirim	GTX 1080 Ti	27 699	35 634,6	3848,5
Jean-Zay	V100-32G	6690	17 272,1	2238,5
Olympe	V100-32G	843	6111,8	792,1
Araigne	RTX 2080 Ti	1404	129,3	14,0
Total	N/A	36 636	59 147,9	6893,1

Avec un total de 6,9 tonnes d'équivalent CO_2, le coût des entraînements en carbone de ma thèse est donc légèrement inférieur à l'empreinte d'un français moyen par an qui atteint 8,9 tonnes d'équivalent CO_2 en 2021 [65]. De plus, j'ai remarqué que ces entraînements ont causé bien plus d'émissions que les transports nécessaires à ma venue aux différentes conférences DCASE2022, DCASE2023, EUSIPCO2023 et INTERSPEECH2023, qui s'élève à approximativement 1,5 tonnes d'équivalent CO_2 pour les trains et vols d'avion d'après un simulateur de Labos1point5 [66]. Au total, les principales émissions de carbone liées à la recherche de cette thèse représentent donc environ 8,4 tonnes d'équivalent CO_2. La quantité d'entraînements lancée aurait pu être réduite en utilisant des méthodes d'optimisation plus intelligentes que des recherches par grille dans certains cas, comme de la recherche aléatoire ou de l'optimisation bayésienne [Nogueira 2014]. Cependant, il faut noter que la plupart des grilles de recherche d'hyperparamètres de cette thèse n'ont testé que quelques combinaisons de valeurs pour un système donné (moins de 10 en général), ce qui ne garantit pas de trouver de bons résultats par d'autres méthodes en si peu d'essais. De plus, les bibliothèques implémentant de meilleures méthodes d'optimisation devraient être adaptées pour supporter des expériences sur cinq graines différentes en parallèle, tout en gardant une compatibilité avec le gestionnaire SLURM. Le gel ou non notre encodeur de DTAA a eu un impact important sur la durée de nos entraînements, de même que la nécessité de faire nos expériences sur cinq initialisations différentes.

Avec un bilan carbone aussi important, il est légitime de se questionner sur la pertinence des expériences qui ont été menées durant cette thèse. En effet, celle-ci a suivi la tendance de la littérature à créer des systèmes de plus en plus grands ou complexes pour dépasser l'état de l'art de quelques points. Nous avons cependant montré qu'il était possible de créer des systèmes relativement légers tout en gardant de bonnes performances sur les jeux de données disponibles. À l'avenir, le suivi du nombre de paramètres, de la durée d'entraînement, du nombre d'opérations et même de la consommation énergétique et des émissions engendrées les modèles peut devenir important pour déterminer leur pertinence. En effet, à l'heure actuelle, la majorité des études ne donnent que quelques informations (durée d'un entraînement, type de GPU et parfois nombre de paramètres). Pourtant, ces dernières informations sont non seulement importantes pour mieux comprendre l'impact que cette recherche a sur le changement climatique, mais pourrait également remettre en question la tâche de DTAA, qui ne semble pas encore viser d'application précise.

65. https://www.statistiques.developpement-durable.gouv.fr/lempreinte-carbone-de-la-france-de-1995-2021
66. https://apps.labos1point5.org/travels-simulator

Perspectives

Durant cette thèse, nous avons étudié de nombreux aspects liés à la DTAA, mais nous sommes loin de les avoir tous totalement explorés. Il reste de nombreuses améliorations et pistes de recherche à étudier, dont nous allons donner un aperçu dans les paragraphes suivants.

Amélioration de l'encodeur. Récemment, les méthodes autosupervisées (*self-supervised learning*) se sont beaucoup améliorées et ont commencé à fournir des représentations plus pertinentes pour l'image et l'audio. Nous pouvons penser au modèle BEATs, qui a été entraîné à décrire le signal audio à l'aide de séquences d'unités discrètes, comme s'il s'agissait de texte. Il atteint notamment l'état de l'art en 2023 pour la classification audio sur AS pour un modèle seul et a aussi été utilisé dans l'état de l'art de DTAA sur CL. Pour les modèles autosupervisés, une autre piste est ConvNeXt-V2 [Woo 2023], qui est une autre amélioration de ConvNeXt entraîné à reconstruire en sortie l'image d'entrée. Une autre idée différente consisterait à exploiter un modèle de RAT comme CLAP, ou d'entraîner notre propre encodeur sur des données audio-texte. Cela pourrait permettre à l'encodeur de capturer plus d'informations que les classes d'événements sonores avant ou pendant l'entraînement de DTAA, ce qui semble aider certains systèmes comme celui de l'étude [Xu 2022] ou le système de référence de la tâche 6a de DCASE en 2023 [67]. Toujours dans l'optique d'améliorer notre encodeur, une récente étude [Komatsu 2023] propose d'employer une méthode d'apprentissage de description de différence audio, puis d'utiliser le modèle pour la DTAA. Cela pourrait permettre à notre modèle de mieux capturer certains détails en fonction de la paire de fichiers en entrée, tout en créant un système capable de couvrir deux tâches différentes. Pour toutes ces approches, il serait également intéressant de dégeler notre encodeur durant l'apprentissage de DTAA pour permettre au modèle de repérer de nouveaux éléments décrits par les phrases et de mieux s'adapter aux fichiers audio. Nous avons tenté de dégeler notre modèle ConvNeXt, mais la performance a grandement diminué (environ 4% de baisse de SPIDEr sur AC), car le λ_{wd} global était beaucoup trop élevé pour la partie encodeur. Pour résoudre ce problème, il serait nécessaire d'utiliser deux optimiseurs avec deux pénalités de poids et peut-être deux pas d'apprentissage différents, pour éviter de trop altérer les poids de l'encodeur.

Amélioration du décodeur. Dans ce manuscrit, nous n'avons pas étudié le pré-entraînement du décodeur, alors qu'une majeure partie des systèmes de DTAA récents exploite des décodeurs tels que BART. Cela est notamment dû au fait que ces modèles sont plutôt larges (supérieurs à 100 millions de paramètres), ce qui est assez coûteux alors que notre modèle atteint déjà de très bons scores avec seulement 12 millions de paramètres pour le décodeur de SR3. Cependant, le niveau de langue des références des jeux de DTAA semble plutôt pauvre lorsque nous le comparons avec ceux des jeux de données texte. Il serait intéressant d'étudier plus en profondeur si l'utilisation de ces modèles de texte pré-entraînés apportent un gain de performance significatif. Une autre piste d'amélioration serait de moderniser l'architecture du décodeur en exploitant certaines avancées des réseaux de neurones comme l'attention multirequêtes (*Multi-Query Attention*) [Shazeer 2019] pour diminuer la complexité du modèle en factorisant certains poids. Pour le *Transformer*, il existe également l'encodage positionnel relatif [Shaw 2018], pour représenter la position d'un mot dans la phrase par rapport au mot suivant à prédire plutôt que de donner une position absolue. Certaines études récentes, comme l'*Attention Free Transformer* [Zhai 2021], le *Receptance Weighted Key Value* [Peng 2023] ou encore les *Retentive Network* [Sun 2023c],

67. https://dcase.community/challenge2023/task-automated-audio-captioning#hyper-parameters

proposent de nouvelles architectures mêlant *Transformer* et RNN, pour essayer d'obtenir un modèle à la fois performant pour chaque tâche, parallélisable durant l'apprentissage, rapide durant l'inférence ou pouvant modéliser plus efficacement les dépendances à long terme.

Introduction de nouvelles données. À court terme, l'introduction de nouvelles données pourrait parfaire un système de DTAA pour le rendre performant sur de nouveaux types d'événements sonores et de descriptions. Nous pouvons notamment citer le jeu de données AutoACD (section 1.3.2.5) qui comporte environ deux millions de fichiers audio pour l'entraînement. De plus, il contient un nouveau jeu de test qui serait pertinent de comparer à ceux de CL et d'AC. MusicCaps (section 1.3.2.6) propose moins de données que le jeu précédent, mais se focalise sur des descriptions de musique par des professionnels de ce domaine, ce qui donne des descriptions riches en détails et même composées de plusieurs phrases. *Song Describer Dataset* (section 1.3.2.7) propose également des descriptions de musique, mais pas nécessairement annotées par des spécialistes. Ces deux derniers jeux semblent contenir des phrases très différentes de celles des autres jeux de données, comme *« A strings orchestra and piano combine in this waltz to give a fantasy feeling. »*. Il faut cependant noter que certains de ces jeux ont des recouvrements avec AS, et donc possiblement avec AC, ce qui pourrait biaiser les performances des modèles.

D'autres représentations de tâches. Dans le chapitre 7, nous avons proposé une représentation de la tâche par jeu de données, permettant au modèle de générer des candidats mimiquant plusieurs styles d'annotation. Cependant, il pourrait être intéressant de pousser plus loin cette approche, en déterminant les tâches en fonction d'autres critères que le jeu de données. La tâche choisie pourrait par exemple être liée à la fréquence des mots présents dans les phrases, pour contrôler le niveau de détails à produire par le modèle. Par exemple, si nous définissions deux tâches, l'une pourrait correspondre à des descriptions simples comme *« a man speaks »* qui possèdent des mots très communs, tandis qu'une autre donnerait des descriptions plus détaillées contenant des n-grammes peu fréquents comme *« the man is speaking **through a telephone receiver** »*. Le modèle verrait donc uniquement la première TE pour les phrases simples et la seconde pour les phrases complexes. La performance d'un tel système pourrait grandement varier en fonction du jeu de données, mais pourrait permettre à un utilisateur de mieux contrôler le résultat attendu, en demandant explicitement une phrase plus précise. Cependant, un tel entraînement pourrait biaiser les prédictions du système, car certains événements peu communs ne correspondront qu'à la première ou seconde tâche. Par exemple, le modèle pourrait prédire constamment la phrase *« a man speaks »* durant l'inférence quel que soit le fichier audio en entrée si nous lui donnions la première TE, car cette phrase est très fréquente pour le sous-ensemble AC-train.

Concaténation et superposition de fichiers audio pour tester les modèles de RAT. Pour mieux évaluer et comprendre les systèmes multimodaux audio-texte, nous pourrions sommer ou concaténer deux fichiers audios pour voir si le modèle l'associe à une description qui contient les deux événements sonores. Ce type de stratégie pourrait être pertinent pour la RAT, car cela permettrait de savoir si le modèle arrive à repérer plusieurs événements sonores, mais aussi s'il arrive à les repérer dans le bon ordre. Le tableau 3 donne un ensemble de tests qui pourraient être effectués entre deux paires (A_1, T_1) et (A_2, T_2). La concaténation et la superposition de deux audios sont simples, mais pour les phrases cela nécessite l'emploi de mots clés comme *« followed by »* ou *« while »*, ou bien l'introduction d'une unité lexicale séparant les phrases. Ce type de test pourrait permettre d'évaluer plusieurs aspects en même temps. Il faudrait donner

au modèle chaque combinaison audio-texte possible, et vérifier qu'il attribue les scores les plus élevés aux exemples cohérents. Par exemple, le modèle noté f donnant un score de similarité pour une paire audio-texte devrait détecter l'ordre des éléments en suivant une inégalité comme $f(\mathrm{A}_{1,2}, \mathrm{T}_{1,2}) > f(\mathrm{A}_{1,2}, \mathrm{T}_{2,1})$, avec $\mathrm{A}_{i,j}$ désignant la concaténation de A_i et A_j (même chose pour $\mathrm{T}_{i,j}$ avec T_i et T_j). Nous pourrions également faire un test pour déterminer si le modèle distingue bien les événements superposés de ceux concaténés. La superposition des signaux audio devrait être audible en s'assurant par exemple que leur énergie quadratique moyenne relative est dans un intervalle de 5 dB. À l'inverse, si on ajoute un signal audio inaudible, l'audio le plus faible ne devrait peut-être pas être décrit. Pour la DTAA, ce type de test pourrait rester pertinent tant que la concaténation de phrases ne devienne pas trop longue, car les modèles produisent bien souvent des descriptions qui contiennent en moyenne autant de mots qu'une seule phrase. Ainsi, nous pourrions par exemple vérifier si le candidat généré par un système de DTAA à partir de concaténation de deux audios correspond à la concaténation des deux références correspondantes.

TABLE 3 – Correspondances attendues pour un modèle de RAT sur des paires audio-texte. T_i désigne une phrase correspondant à l'audio A_i. $\mathrm{A}_{i,j}$ désigne la concaténation de A_i et A_j et A_{i+j} désigne la superposition de A_i et A_j.

N°	Description	A_1	A_2	$\mathrm{A}_{1,2}$	$\mathrm{A}_{2,1}$	A_{1+2}
T_1	*A man speaks.*	✓	×	×	×	×
T_2	*Rain falls.*	×	✓	×	×	×
$\mathrm{T}_{1,2}$	*A man speaks* ***followed by*** *rain falls.*	×	×	✓	×	×
$\mathrm{T}_{2,1}$	*Rain falls* ***followed by*** *a man speaks.*	×	×	×	✓	×
T_{1+2}	*A man speaks* ***while*** *rain falls.*	×	×	×	×	✓
T_{2+1}	*Rain falls* ***while*** *a man speaks.*	×	×	×	×	✓

Des annotations plus détaillées. Tout au long de ce manuscrit, nous avons pu voir que les descriptions produites par les systèmes de DTAA étaient bien souvent assez génériques et peu diversifiées. Cela est notamment dû au manque de données précises et détaillées, qui pourraient apporter une réelle plus-value par rapport aux descriptions actuelles qui se focalisent trop sur les événements sonores pour les jeux de données les plus larges (AC et WC). En observant le domaine de l'image, nous pouvons citer le jeu de données *Visual Genome* [Krishna 2017], qui propose plus de 100 000 images annotées par plus de 42 descriptions de différentes régions de l'image, ainsi qu'une moyenne de 17 questions et réponses associées à l'image. Ce type d'annotation riche permet de lier de nombreux concepts visuels au langage, mais aussi d'obtenir des informations précises sur la localisation de certains éléments. Le jeu propose également des graphes de scènes, similaires à ceux utilisés dans la métrique SPICE, construits à partir des références humaines. Créer un tel jeu de données pour l'audio serait coûteux, mais pourrait être bénéfique à l'entraînement de systèmes plus robustes et ouvrir la voie vers une évaluation plus fine des systèmes de DTAA. L'application de certaines techniques d'apprentissage actif (*Active Learning*) pourrait limiter le coût de ce type d'annotation, en demandant aux annotateurs de décrire les fichiers audio que le modèle n'arrive pas à décrire ou pour lesquels il génère des phrases peu détaillées. Si un système de DTAA arrive à mieux décrire les audio avec beaucoup de détails, il pourrait être appliqué à d'autres tâches comme l'AT ou la RAT. Pour la recherche texte-vers-audio, nous pourrions employer des systèmes larges et plutôt lents, car les descriptions audio peuvent être calculées au préalable pour chaque fichier audio. Elles pourraient être composées de plusieurs phrases pour

s'assurer de contenir le maximum d'information.

Utilisation d'images. Dans le cas du jeu de données AS, certaines informations visuelles ont été introduites par erreurs par les annotateurs dans les descriptions audio. Pour palier à ce problème, les auteurs de [Liu 2023] ont proposé une méthode d'attention sur les représentations audio et images provenant des vidéos d'AS. Surprenamment, les systèmes de descriptions ne bénéficient pas nécessairement de la représentation de la vidéo directe, mais leur méthode d'attention multimodale permet au modèle d'utiliser ces informations visuelles uniquement lorsque les poids de l'attention de l'audio sont faibles. L'impact de ce type d'apprentissage serait cependant peu intéressant pour le jeu CL, où les annotateurs n'ont eu accès qu'à l'audio. Cependant, nous pourrions imaginer un apprentissage multitâche pour réaliser la DTAI et la DTAA par un seul système qui aurait une représentation unifiée des trois modalités. Ainsi, combiner des informations supplémentaires pourrait améliorer la généralisation des systèmes de DTAA, pour ainsi associer des concepts sonores et visuels au langage. Typiquement, cela pourrait être intéressant dans le cadre de la description automatique de vidéo contenant de l'audio.

Combinaison avec la RAP. À l'heure actuelle, les systèmes de RAP et de DTAA sont bien souvent dissociés. Pourtant, la performance d'un modèle de parole dans un environnement bruité peut dépendre de sa capacité à intégrer les différents bruits de fond de manière robuste. La plupart des systèmes de RAP sont entraînés à ignorer tous les bruits autres que la parole (à l'aide d'augmentations par exemple). Il pourrait être intéressant de créer des systèmes modélisant plutôt les événements sonores, pour être potentiellement plus efficace dans ce type de contexte. Au moment de la rédaction de ce manuscrit, seule l'étude [Narisetty 2022] semble avoir attaqué ce problème avec la DTAA, en proposant un modèle encodeur-décodeur appris à prédire la concaténation de la description des sons et des mots prononcés. Cependant, ce type d'approche est incompatible avec certaines méthodes de RAP comme la fonction de coût *Connectionnist Temporal Classification* [Graves 2006], car les descriptions de DTAA ne sont pas liées à l'audio sur l'axe temporel. Le modèle multitâche reste donc moins performant pour la RAP qu'un modèle entraîné avec cette fonction de coût. Une approche différente pourrait reposer sur une représentation de la tâche pour demander au décodeur d'effectuer la DTAA ou la RAP, mais cela nécessiterait probablement d'intégrer un second encodeur pré-entraîné conçu pour la parole, comme Wav2Vec [Baevski 2020] par exemple. De plus, une étude récente [Gong 2023] réutilisant le modèle de RAP Whisper a montré que ce dernier avait des capacités à modéliser les événements sonores pour la tâche d'AT. Cela semble justifier des recherches futures plus approfondies entre la parole et les événements sonores, qui pourraient permettre d'améliorer l'une ou l'autre tâche. La reconnaissance d'événements sonores pourrait permettre également aux modèles d'éviter d'halluciner certains mots prononcés, ou encore de corriger le contexte de certaines discussions.

Application pour les malentendants. L'une des applications mises en avant dans les descriptions de la tâche de DTAA est bien souvent l'aide aux malentendants. Mais pour arriver à ce but, il serait nécessaire de communiquer avec les personnes concernées pour connaître leurs attentes vis-à-vis d'un tel système. Cela pose de nombreuses interrogations sur plusieurs aspects des systèmes actuels. Par exemple, une description basique d'un audio comme *« Rain is falling. »* est-elle réellement adéquate pour les malentendants ? Il faudrait définir la forme que devrait prendre le texte généré (une courte phrase ou bien quelques mots-clés ? le texte doit-il être placé après une transcription de parole ?). De plus, il serait nécessaire de définir les événements sonores pertinents pour ce type de personne (le son d'une alarme par exemple). Si nous nous focalisons

sur de la description de films et de contenus audio et vidéo pour les malentendants, la DTAA devrait être utilisé lorsqu'une information sonore est non-visible. Cela peut concerner certains événements sonores comme une alarme ou bien les événements hors-champ. À l'inverse, le système ne devrait pas décrire une information si cette dernière est déjà visible, comme des discussions entre personnes ou de la pluie dans certains cas.

Annexe A

Hyperparamètres de nos systèmes

Le tableau A.1 présente les hyperparamètres et statistiques de nos différents systèmes de références conçus tout au long de cette thèse.

TABLE A.1 – Principaux hyperparamètres et statistiques de nos systèmes de référence.

Nom	**Valeur(s)**							
	SR1		**SR2**		**SR3**		**CoNeTTE**	
	AC	**CL**	**AC**	**CL**	**AC**	**CL**	**AC**	**CL**
Encodeur	CNN10		CNN14D		CNext		CNext	
Encodeur gelé ?	Non		Oui		Oui		Oui	
Décodeur	trans.		trans.		trans.		trans.	
Taille du lot (B)	8		512		512		512	
Optimiseur	Adam		AdamW		AdamW		AdamW	
Pas d'apprentissage (lr_0)	$5 \cdot 10^{-4}$		$5 \cdot 10^{-4}$		$5 \cdot 10^{-4}$		$5 \cdot 10^{-4}$	
β_1	0,9		0,9		0,9		0,9	
β_2	0,999		0,999		0,999		0,999	
ϵ	10^{-8}		10^{-8}		10^{-8}		10^{-8}	
Pénalité des poids (λ_{wd})	10^{-6}		10^{-6}		2		2	
Taille minimale des candidats (L_{min})	0		3		3		3	
Nombre de params. appris (M)	16,7		12,4		12,0		19,8	
Nombre de params. gelés (M)	0		79,7		28,2		28,2	
Nombre de params. total (M)	16,7		92,1		40,2		48,0	
MACs (G)	13,3		53,1		20,2		20,5	
Nb. d'époques (K)	50	50	400	100	400	100	400	100
Limite de la norme du gradient	10	1	10	1	10	1	10	1
Lissage des étiquettes (p_{LS})	0,1	0,2	0,1	0,2	0,1	0,2	0,1	0,2
Taille maximale des candidats (L_{max})	52	20	30	20	30	20	30	20
Taille du faisceau (N_{beam})	2	9	2	8	2	3	2	3
Décrit dans le chapitre	3		5		5		7	
Table(s) des résultats	3.3, 3.4		5.5		5.15		7.3	
Réutilisé dans le(s) chapitre(s)	4		6		7, 8, 9, 10		∅	

Annexe B

Détails des systèmes de l'état de l'art

Pour construire notre état de l'art dans la première partie de ce manuscrit, nous avons collecté 122 articles différents qui étudient la DTAA. Le tableau B.1 donne le nombre d'étude par année. On peut constater que l'intérêt pour ce sujet a réellement démarré en 2020, l'année du début de ma thèse.

Table B.1 – Nombre d'études sur la DTAA par année.

Année	2017	2019	2020	2021	2022	2023	Total
Nombre	1	5	21	31	28	36	122

Les tableaux B.2 et B.3 fournissent les informations détaillées des différents articles proposant au moins un système de DTAA. Ils donnent notamment la plupart des valeurs utilisées dans les figures 1.13 et 1.14. Pour les noms des colonnes, Params. signifie Paramètres, SD signifie SPIDEr, entr. signifie entraînement, Nb. mod. signifie nombre de modèles, RL signifie apprentissage par renforcement, Optim signifie optimiseur, lr_0 signifie pas d'apprentissage initial, B signifie taille du lot et N_{beam} est la taille du faisceau. Certaines valeurs ne sont pas précisées dans les articles correspondants, et sont donc remplacés par le symbole « ? » dans les tableaux. Un acronyme est utilisé pour représenter chaque source de données. AC, CL, MA, WC, AS sont déjà présentés dans le chapitre 1 de ce manuscrit. FSD indique des données externes issues du site Freesound, CLv concerne le sous-ensemble de validation de CL, WSJ correspond au *Wall Street Journal* corpus, SDs est l'acronyme de SoundDescs, WT est la contraction de WavText5k, et SB, SJ, ZT correspondent respectivement aux sites web SoundBible, SoundJay et Zapsplat. Les données sont également disponibles au format CSV en ligne [68].

68. https://drive.google.com/file/d/1eTg5DkVfrBOKvWOt7Fk_6KjrHxG99NSC/view?usp=sharing

TABLE B.2 – Première table des données détaillées de l'état de l'art.

N°	Article	Params. (M, ↓)	SD (↑)	Données d'entraînement	Test	Pré-entr.	Nb. mod.	RL	Année	Optim.	lr_0	B	N_{beam}
1	[Wu 2023b]	≈1600	0,326	AC+CL	CL	audio+texte	1	Non	2023	AdamW	2e-4 ; 2e-5	32	1
2	[Wu 2023b]	≈3500	0,336	AC+CL	CL	audio+texte	20	Non	2023	AdamW	2e-4 ; 2e-5	32	1
3	[Mei 2023]	220	0,255	CL	CL	audio+texte	1	Non	2023	Adam	5e-5 ; 5e-6	48	?
4	[Mei 2023]	220	0,310	CL+WC	CL	audio+texte	1	Non	2023	Adam	5e-5 ; 5e-6	48	?
5	[Mei 2023]	171	0,444	AC	AC	audio+texte	1	Non	2023	Adam	5e-5 ; 5e-6	48	?
6	[Mei 2023]	171	0,485	AC+WC	AC	audio+texte	1	Non	2023	Adam	5e-5 ; 5e-6	48	?
7	[Shin 2023]	105	0,472	AC	AC	audio	1	Non	2023	Adam	1e-5	8	3
8	[Xu 2022]	528	0,325	AC+CL+MA	CL	audio	3	Oui	2022	Adam	1e-4 ; 2e-5	128	3
9	[Won 2021]	88	0,285	CL	CL	audio	1	Non	2021	Adam	1e-4	8	3
10	[Kim 2022]	108	0,475	AC	AC	audio	1	Non	2022	AdamW	1e-4	?	3
11	[Yuan 2021]	986	0,318	AC+CL+CLv+FSD	CL	audio	12	Non	2021	?	3e-4 ; 1e-4 ; 5e-5	?	3
12	[Gontier 2021]	494	0,465	AC	AC	audio+texte	1	Non	2021	AdamW	1e-5	8	4
13	[Kim 2019a]	≈150	0,369	AC	AC	audio	1	Non	2019	Adam	?	?	?
14	[Koizumi 2020c]	82,5	0,207	CL	CL	aucun	50	Non	2020	AdamW	1e-4	48	5
15	[Koizumi 2020c]	≈1	0,196	CL	CL	aucun	1	Non	2020	AdamW	1e-4	48	5
16	[Mei 2021c]	117	0,420	AC	AC	audio	1	Non	2021	Adam	1e-4	128	5
17	[Mahfuz 2023b]	14	0,370	AC	AC	audio	1	Non	2022	?	1e-3	32	1
18	[Mahfuz 2023b]	14	0,208	CL	CL	audio	1	Non	2022	?	1e-3	32	1
19	[Liu 2023]	≈117	0,442	AC	AC	audio	1	Non	2023	Adam	1e-4	32	3
20	[Mei 2022a]	8	0,266	CL	CL	audio	1	Non	2022	?	1e-4	32	3
21	[Mei 2022a]	8	0,301	CL	CL	audio	1	Oui	2022	?	1e-4 ; 5e-5	32	3
22	[Liu 2022a]	25	0,419	AC	AC	audio+texte	1	Non	2022	Adam	5e-4	32	5
23	[Cho 2023]	≈1000	0,313	AC+CL+WC	CL	audio+texte	6	Oui	2023	?	5e-5 ; 5e-6 ; 1e-5	32	3
24	[Labbé 2023b]	98	0,320	AC+CL+MA+WC	CL	audio	5	Non	2023	AdamW	5e-4	512	3
25	[Labbé 2023b]	42	0,312	AC+CL+MA+WC	CL	audio	1	Non	2023	AdamW	5e-4	512	3
26	[Labbé 2023b]	40	0,305	CL	CL	audio	1	Non	2023	AdamW	5e-4	512	3
27	[Labbé 2023b]	87	0,269	CL	CL	audio	1	Non	2023	AdamW	5e-4	512	3
28	[Kouzelis 2022]	≈560	0,296	AC+CL+MA	CL	audio	1	Non	2022	?	1e-4 ; 1e-5 ; 1e-5	32	3
29	[Labbé 2022]	16	0,247	CL	CL	audio	1	Non	2022	Adam	5e-4	8	9
30	[Labbé 2022]	16	0,401	AC	AC	audio	1	Non	2022	Adam	5e-4	8	2
31	[Xu 2021a]	≈10	0,246	CL	CL	audio	1	Non	2021	Adam	5e-4	32	3
32	[Xu 2021a]	≈10	0,414	AC	AC	audio	1	Non	2021	Adam	5e-4	32	3
33	[Berg 2021]	4	0,318	AC	AC	aucun	1	Non	2021	Adam	?	12	2
34	[Tran 2021]	4	0,182	CL	CL	aucun	1	Non	2020	Adam	?	12	2
35	[Eren 2023]	222	0,264	CL	CL	audio+texte	1	Non	2023	AdamW	1e-5	8	?
36	[Labbé 2023a]	40	0,495	AC	AC	audio	1	Non	2023	AdamW	5e-4	512	2
37	[Labbé 2023a]	40	0,301	CL	CL	audio	1	Non	2023	AdamW	5e-4	512	3
38	[Labbé 2023a]	48	0,468	AC+CL+MA+WC	AC	audio	1	Non	2023	AdamW	5e-4	512	2
39	[Labbé 2023a]	48	0,305	AC+CL+MA+WC	CL	audio	1	Non	2023	AdamW	5e-4	512	3

TABLE B.3 – Seconde table des données détaillées de l'état de l'art.

N°	Article	Params. (M, ↓)	SD (↑)	Données d'entraînement	Test	Pré-entr.	Nb. mod.	RL	Année	Optim.	lr_0	B	N_{beam}
40	[Narisetty 2022]	≈87	0,401	AC+WSJ	AC	aucun	1	Non	2022	Adam	?	64	6
41	[Mei 2022c]	8	0,256	AC+CL+CLv	CL	audio	1	Oui	2021	?	1e-4	32	5
42	[Mei 2021b]	8	0,277	AC+CL+CLv	CL	audio	1	Oui	2021	Adam	1e-3 ; 1e-4	32	3
43	[Koh 2022]	≈10	0,246	CL	CL	audio	1	Non	2021	?	3e-4	512	4
44	[Ye 2021a]	≈80	0,269	CL	CL	audio	1	Non	2021	Adam	1e-3	32	4
45	[Ye 2021a]	≈80	0,311	CL	CL	audio	1	Oui	2021	Adam	1e-3 ; 3e-4 ; 5e-5	32	4
46	[Liu 2021]	8	0,242	AC+CL+CLv	CL	audio	1	Non	2021	Adam	1e-3	32	?
47	[Kouzelis 2023]	≈380	0,403	AC+WC	AC	audio	1	Non	2023	Adam	2e-5 ; 4e-5 ; 8e-5	64	1
48	[Kouzelis 2023]	≈380	0,247	CL+WC	CL	audio	1	Non	2023	Adam	2e-5 ; 4e-5 ; 8e-5	64	1
49	[Deshmukh 2023]	≈312	0,455	AC+WC	AC	audio+texte	1	Non	2023	Adam	1e-4	128	5
50	[Deshmukh 2023]	≈312	0,261	CL+WC	CL	audio+texte	1	Non	2023	Adam	1e-4	128	5
51	[Kim 2023b]	205	0,255	CL	CL	audio+texte	1	Non	2023	AdamW	5e-5	55	?
52	[Kim 2023b]	205	0,455	AC	AC	audio+texte	1	Non	2023	AdamW	5e-5	75	?
53	[Çakır 2020]	≈5	0,074	CL	CL	aucun	1	Non	2020	Adam	1e-4	32	1
54	[Ye 2022]	≈86	0,286	CL	CL	audio	1	Non	2022	Adam	5e-4 ; 3e-4	?	4
55	[Ye 2022]	≈344	0,323	CL	CL	audio	4	Oui	2022	Adam	5e-4 ; 3e-4 ; 5e-5	?	4
56	[Primus 2022]	130	0,264	CL	CL	audio+texte	1	Non	2022	Adam	1e-3 ; 1e-5	?	?
57	[Primus 2022]	130	0,284	AC+CL	CL	audio+texte	1	Non	2022	Adam	1e-3 ; 1e-5 ; 1e-5	?	?
58	[Primus 2022]	130	0,290	AC+CL	CL	audio+texte	1	Oui	2022	Adam	1e-3 ; 1e-5 ; 1e-5 ; 1e-5	?	?
59	[Primus 2022]	780	0,295	AC+CL	CL	audio+texte	6	Oui	2022	Adam	1e-3 ; 1e-5 ; 1e-5 ; 1e-5	?	?
60	[Chen 2022a]	>30	0,263	CL+CLv	CL	audio	1	Non	2022	Adam	1e-5 ; 1e-3	8	3
61	[Sun 2023a]	>90	0,295	AC+CL+CLv+MA+WC	CL	audio	2	Non	2023	Adam	5e-4	32	3
62	[Schaumlöffel 2023]	248	0,292	AC+CL+MA+SDs+WT	CL	audio+texte	1	Non	2023	AdamW	1e-5 ; 1e-6	64	8
63	[Wu 2020]	8	0,227	CL	CL	aucun	1	Non	2020	?	3e-4 ; 1e-4	16	3
64	[Wang 2020]	12	0,172	CL	CL	aucun	1	Non	2020	Adam	1e-3	?	5
65	[Pellegrini 2020]	2	0,124	CL	CL	aucun	1	Non	2020	Adam	5e-4	64	25
66	[Yang 2021]	3	0,166	CL	CL	aucun	1	Non	2021	Adam	?	?	2
67	[Kadlčík 2023]	39	0,224	AC+AS+CL	CL	audio+texte	1	Non	2023	?	4e-6 ; 2e-5	32	?
68	[Kadlčík 2023]	244	0,269	AC+AS+CL	CL	audio+texte	1	Non	2023	?	4e-6 ; 2e-5	32	?
69	[Kadlčík 2023]	≈1000	0,279	AC+AS+CL	CL	audio+texte	1	Non	2023	?	4e-6 ; 2e-5	32	?
70	[Huang 2022]	9	0,249	CL+CLv	CL	audio	1	Non	2022	Adam	1e-3	16	3
71	[Huang 2022]	9	0,251	CL+CLv	CL	audio	1	Non	2022	Adam	1e-3	16	3
72	[Huang 2022]	9	0,253	CL+CLv	CL	audio	1	Non	2022	AdamW	1e-3	16	3
73	[Huang 2022]	9	0,257	CL+CLv	CL	audio	1	Non	2022	AdamW	1e-3	16	3
74	[Kicinski 2022]	104	0,279	AC+CL+FSD	CL	audio	1	Non	2022	AdamW	1e-4 ; 5e-5	128	3
75	[Kicinski 2022]	207	0,255	AC+CL+FSD	CL	audio+texte	1	Non	2022	AdamW	1e-4 ; 5e-5	128	3
76	[Kicinski 2022]	104	0,260	AC+CL+FSD	CL	audio	1	Non	2022	AdamW	1e-4 ; 5e-5	128	3
77	[Kicinski 2022]	207	0,249	AC+CL+FSD	CL	audio+texte	1	Non	2022	AdamW	1e-4 ; 5e-5	128	3
78	[Han 2021]	≈2000	0,318	AC+CL+CLv+FSD+SB+SJ+ZT	CL	audio	4	Non	2022	?	3e-4 ; 1e-4 ; 5e-5	?	3

Annexe C

Résultats des *FENSE* et *S2V benchmarks*

Le *FENSE benchmark* est une méthode d'évaluation des métriques. Il repose sur un jeu de données de paires de phrases associées à un fichier audio. Il existe quatre types de paires : Humain-correct (Hc) qui correspond à une paire de références, Humain-incorrect (Hi) avec deux références, mais l'une ne correspond pas à l'audio, Humain-machine (Hm) avec une référénce et un candidat de l'audio, et enfin le cas Machine-machine (Mm) pour deux candidats de l'audio. Le tableau C.1 présente les exactitudes (*accuracies*) de nombreuses métriques sur le *FENSE benchmark* décrit dans la section 2.3.2. Ces scores ont été récupérés à partir différentes études des métriques de DTAA qui emploient le *FENSE benchmark* : [Zhou 2022, Gontier 2023, Wijngaard 2023, Mahfuz 2023a]. Globalement, FENSE reste la meilleure métrique avec les scores moyens (Total) les plus élevés, notamment grâce à son score dans le cas Mm par rapport aux autres métriques.

TABLE C.1 – Exactitudes de plusieurs métriques sur le *FENSE benchmark*.

Métrique	**AC-test**					**CL-eval**				
	Hc	**Hi**	**Hm**	**Mm**	**Total**	**Hc**	**Hi**	**Hm**	**Mm**	**Total**
BLEU1	58,6	90,3	77,4	50,3	62,4	51,0	90,6	65,5	50,3	59,0
BLEU4	54,7	85,8	78,7	50,6	61,6	52,9	88,9	65,1	53,2	60,5
METEOR	66,0	96,4	90,0	60,1	71,7	54,8	93,0	74,6	57,8	65,4
ROUGE-L	61,1	91,5	82,8	52,1	64,9	56,2	90,6	69,4	50,7	60,5
CIDEr-D	56,2	96,0	90,4	61,2	71,0	51,4	91,8	70,3	56,0	63,2
SPICE	50,2	83,8	77,8	49,1	59,7	44,3	84,4	65,5	48,9	56,3
SPIDEr	56,7	96,0	90,4	63,4	72,2	53,3	93,4	70,3	57,0	64,2
BERTScore	60,6	97,6	**92,9**	65,0	74,3	57,1	**95,5**	70,3	61,3	67,5
BLEURT	**77,3**	93,9	88,7	72,4	79,3	59,0	93,9	75,4	67,4	71,6
SB_{sim}	64,0	**99,2**	92,5	73,6	79,6	60,0	**95,5**	75,9	66,9	71,8
FENSE	64,5	98,4	91,6	**84,6**	**85,3**	60,5	94,7	80,2	**72,8**	**75,7**
SPIDEr-FL	57,1	95,5	90,0	76,3	79,1	53,8	93,4	76,7	68,9	71,9
SPICE+	63,5	95,5	91,6	83,4	84,0	**61,0**	93,9	**81,0**	70,0	74,1
$ACES_{10}$	52,2	74,5	65,7	55,7	59,9	55,7	84,0	53,4	57,0	60,5
$ACES_{13}$	55,7	77,7	69,0	53,9	60,6	56,2	83,6	58,6	58,2	62,0
SBF (TinyBERT)	40,9	93,5	92,1	66,4	N/A	52,9	89,8	63,8	57,4	N/A
SBF (MiniLM)	40,9	93,1	92,1	70,7	N/A	52,9	88,5	70,3	60,0	N/A

Le tableau C.2 présente les scores de différentes métriques sur le *S2V benchmark* décrit dans la section 2.3.8. Les scores sont issus de l'article présentant la métrique S2V [Bhosale 2023]. S2V obtient le meilleur score pour la corrélation entre les paires positives (*Correct Caption Pair*, CCP), mais est légèrement moins performante que la métrique BERTScore pour les paires négatives (*Incorrect Caption Pair*, ICP).

TABLE C.2 – Corrélations de chaque métrique pour les paires positives et négatives sur CL.

Métrique	**CCP**	**ICP**
BLEU1	62,2	86,1
BLEU2	59,1	85,6
BLEU3	54,6	81,5
BLEU4	50,3	78,2
ROUGE-L	64,8	80,2
METEOR	65,1	82,2
CIDEr-D	53,5	88,2
SPICE	43,4	79,1
BERTScore	70,7	**94,6**
S2V	**72,7**	93,6

Annexe D

Références des sorties du modèles

Le tableau D.1 contient les références associées aux quatre fichiers audio des candidats de la table 3.5 du chapitre 3.

TABLE D.1 – Références de quatre fichiers audio différents issus d'AC-test.

Audio	N°	Descriptions
6BJ455B1aAs	R1	*A rocket flies by followed by a loud explosion and fire crackling as a truck engine runs idle*
	R2	*A whooshing noise followed by an explosion*
	R3	*A missile launching followed by an explosion and metal screeching as a motor hums in the background*
	R4	*Whistling and an explosion*
	R5	*A whistling and then an explosion and crackling*
VjSEIRnLAh8	R6	*Food is frying and a woman talks*
	R7	*A woman is talking as food is frying*
	R8	*Food sizzling with some knocking and banging followed by a woman speaking*
	R9	*Frying and female speech*
	R10	*A woman speaks with food sizzling in a pan with some chopping*
_xylo5_IiaM	R11	*A woman talks and a baby whispers*
	R12	*A woman talking as a baby talks followed by plastic thumping*
	R13	*A woman and a baby are having a conversation*
	R14	*A woman speaks then a small child speaks*
	R15	*A young girl talking as a woman is talking*
PLHXGDnig4M	R16	*A person talking which later imitates a couple of meow sounds*
	R17	*A person speaks and makes meow sounds*
	R18	*A man talking then meowing and hissing*
	R19	*A man talks like a cat before growling then meowing and finally hissing*
	R20	*A man speaking then hissing*

Annexe E

Résultats des variantes du Mixup-in

Les résultats des variantes du Mixup-in décrit dans la section 5.5 sont donnés dans le tableau E.1. Surprenamment, ces variantes ne sont pas meilleures que le Mixup-in qui mélangent les deux entrées du décodeur. À noter que la valeur de FENSE est supérieure sur AC-test lorsque nous ne mélangeons que l'audio, mais le Mixup-in mélangeant les entrées x et y_{prev} est supérieur sur AC-val.

TABLE E.1 – Résultats des variantes et ablations du Mixup-in sur AC-test.

Aug.	α	**Asym.**	**p**	**SD**	**FNS**	**Vocab**
Aucun	0	Oui	0	47,1±1,5	63,8±0,4	**409,6±21,4**
x, y_{prev}	0,4	Oui	1	**48,8±0,8**	64,0±0,2	397,6±19,7
x	0,4	Oui	1	48,0±0,9	**64,1±0,5**	394,6±52,3
y_{prev}	0,4	Oui	1	47,2±1,3	63,6±0,7	394,2±42,2
x, y_{prev}, y_{next}	0,4	Oui	1	47,9±1,4	63,9±0,3	378,8±22,8

Annexe F

Contributions pour le logiciel libre

Durant cette thèse, plusieurs codes sources ont été publiés pour aider le développement des modèles de DTAA, ou pour permettre la reproductibilité de certains résultats ou modèles. Ces différentes bibliothèques peuvent être installées via l'outil Pip.

Bibliothèque `aac-datasets`

La bibliothèque `aac-datasets` propose du code permettant de faciliter le téléchargement et le chargement en mémoire des données et des métadonnées des principaux jeux de données de DTAA : AC, CL, MA et WC. Une courte documentation est également disponible en ligne [69]. [70].

```python
from aac_datasets import Clotho

dataset = Clotho(root=".", subset="eval", download=True)
item = dataset[0]
audio, captions = item["audio"], item["captions"]
# audio: Tensor of shape (n_channels, audio_max_size)
# captions: list of str
```

Bibliothèque `aac-metrics`

La bibliothèque `aac-metrics` propose du code permettant de faciliter l'installation et le calcul de nombreuses métriques de DTAA : BLEU, ROUGE-L, METEOR, CIDEr-D, SPICE, SPIDEr, SPIDEr-FL, SPIDEr-max, FENSE, SBERT-sim et FER, tout en conservant une compatibilité avec les codes d'origine de ces métriques [71] [72]. Une courte documentation est également disponible en ligne [73].

```python
from aac_metrics import evaluate

candidates: list[str] = ["a man is speaking", "rain falls"]
mult_references: list[list[str]] = [["a man speaks.", "someone speaks.", "a man is speaking while a bird is chirping in the background"], ["rain is falling hard on a surface"]]

corpus_scores, _ = evaluate(candidates, mult_references)
print(corpus_scores)
```

69. https://aac-datasets.readthedocs.io
70. https://aac-metrics.readthedocs.io
71. https://github.com/audio-captioning/caption-evaluation-tools
72. https://github.com/blmoistawinde/fense
73. https://aac-metrics.readthedocs.io

```
# dict containing the score of each metric: "bleu_1", "bleu_2", "bleu_3", "
    bleu_4", "rouge_l", "meteor", "cider_d", "spice", "spider"
# {"bleu_1": tensor(0.4278), "bleu_2": ..., ...}
```

Bibliothèque `conette`

La bibliothèque `conette` propose le code nécessaire pour calculer les sorties d'un modèle CoNeTTE sur n'importe quel fichier audio. La version actuelle ne propose que le modèle entraîné pour CL et l'interface reste assez simple à utiliser. Les poids du modèle sont disponibles sur HuggingFace [74], et une interface permet de tester le modèle and téléversant ou enregistrant depuis le micro un fichier audio sur une autre page [75].

```
from conette import CoNeTTEConfig, CoNeTTEModel

config = CoNeTTEConfig.from_pretrained("Labbeti/conette")
model = CoNeTTEModel.from_pretrained("Labbeti/conette", config=config)

path = "/your/path/to/audio.wav"
outputs = model(path)
candidate = outputs["cands"][0]
print(candidate)
```

Bibliothèque `SSLH`

La bibliothèque `SSLH` propose tout le code d'entraînement et de test des méthodes semi-supervisées appliquées à la classification audio décrits dans le chapitre 9. On y retrouve notamment les méthodes MM, RMM, FM, UDA ainsi que certaines de leurs variantes, sur les jeux de données CIFAR-10, ESC-10, UBS8K et GSC.

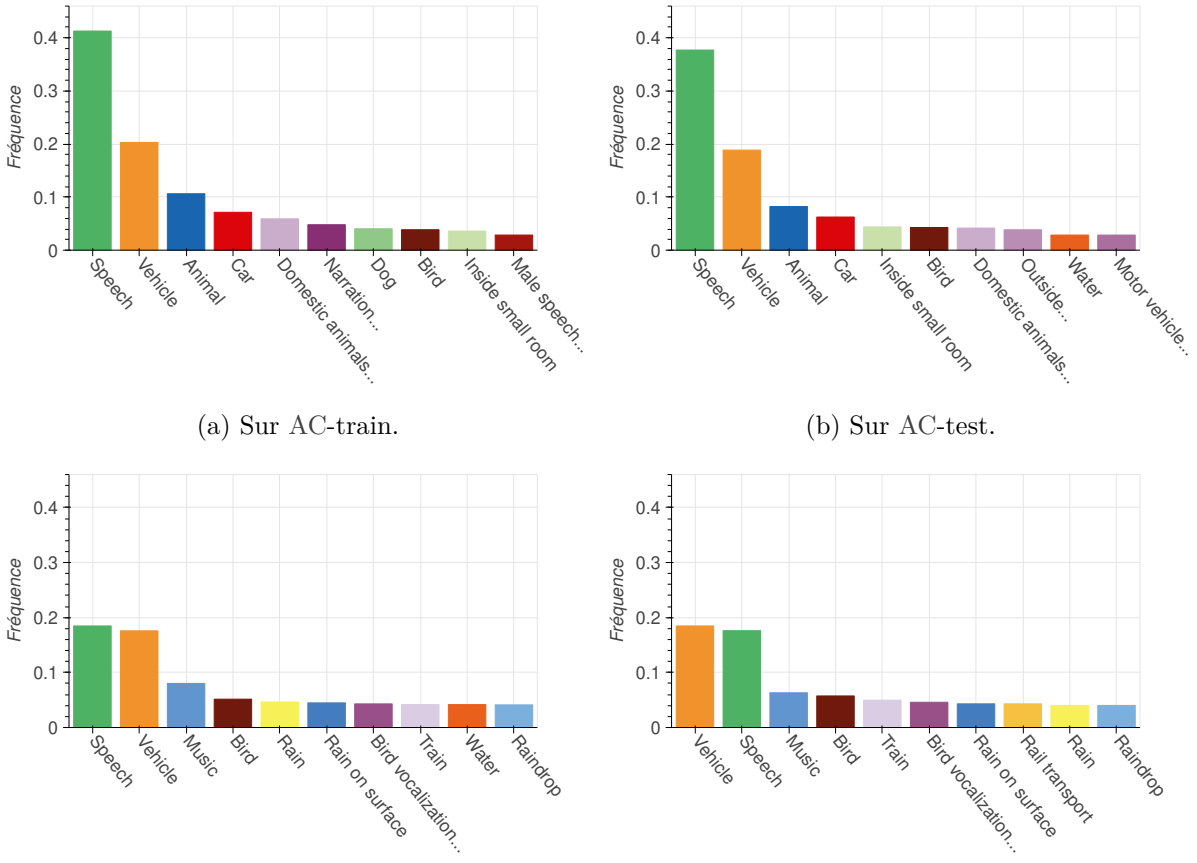
```
# Exemple d'apprentissage de MixMatch sur Google Speech Commands
python -m sslh.mixmatch data=gsc data.download=True
```

74. https://huggingface.co/Labbeti/conette
75. https://huggingface.co/spaces/Labbeti/conette

Annexe G

Distributions des événements sonores

Les figures G.1 représentent les distributions des événements sonores sur les sous-ensembles d'entraînement et de test d'AC et de CL. Les distributions des événements de CL sont estimées par notre modèle ConvNeXt, entraîné sur AS$^{\text{-AC}}$. Curieusement, la classe « *Animal* » n'est pas prédite par le modèle sur CL, alors que la classe « *Bird* » l'est. Il pourrait s'agir d'une erreur du modèle CNext, qui pourrait être réglée en exploitant des seuils différents pour chaque classe d'événement sonore plutôt qu'un seuil global fixé à 0,3.



(a) Sur AC-train.

(b) Sur AC-test.

(c) Sur CL-dev.

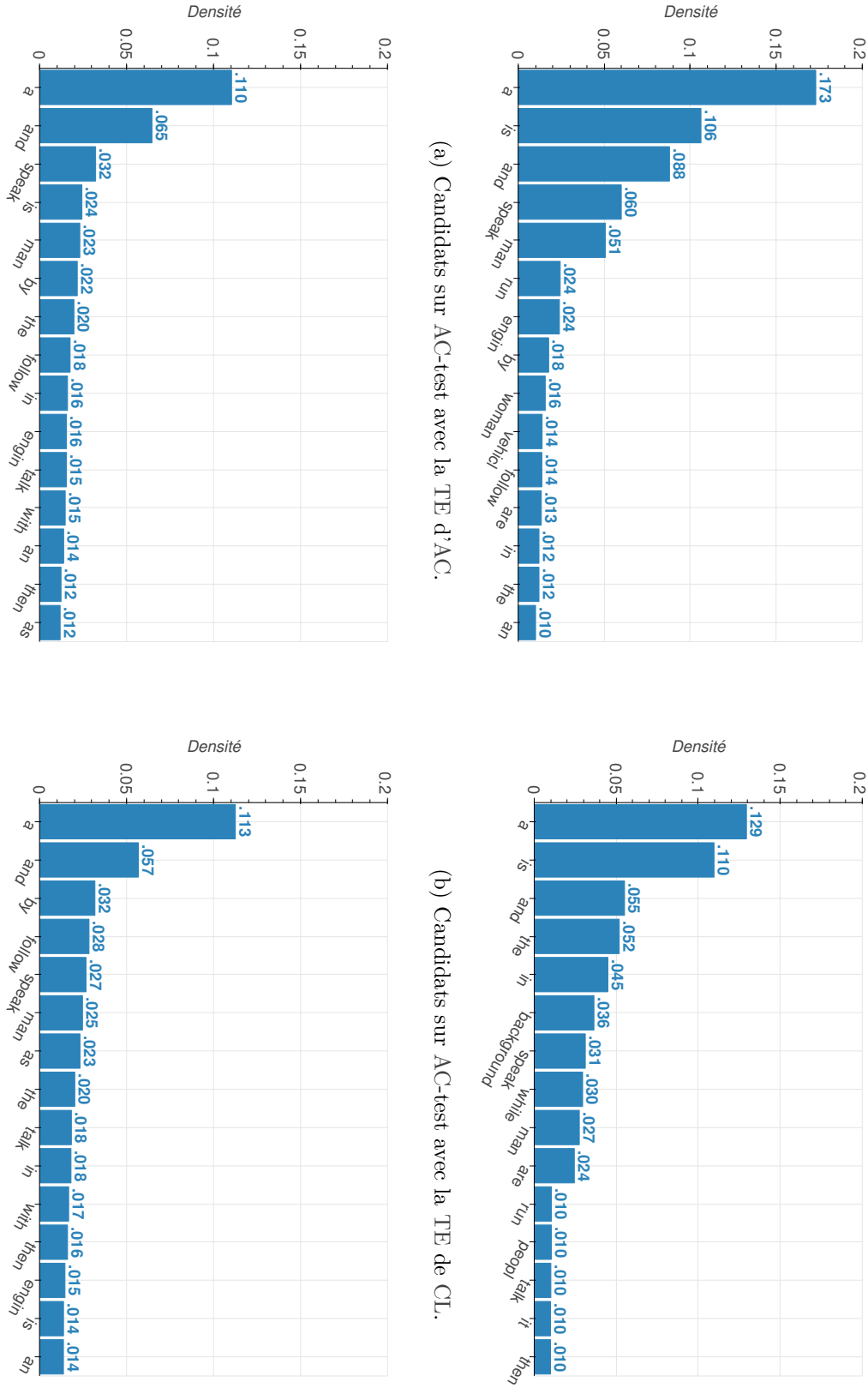
(d) Sur CL-eval.

FIGURE G.1 – Distributions des dix événements sonores les plus fréquents sur quatre sous-ensembles.

Annexe H

Impact de la représentation de tâche sur les descriptions

Les figures des pages suivantes représentent les distributions des unigrammes, trigrammes et des catégories grammaticales des mots issus des candidats générés par différents TE pour le modèle CoNeTTE spécialisé sur CL, ainsi que des références sur les sous-ensembles de tests AC-test et CL-eval. Les remarques sur ces figures sont données dans la section 7.3.2.



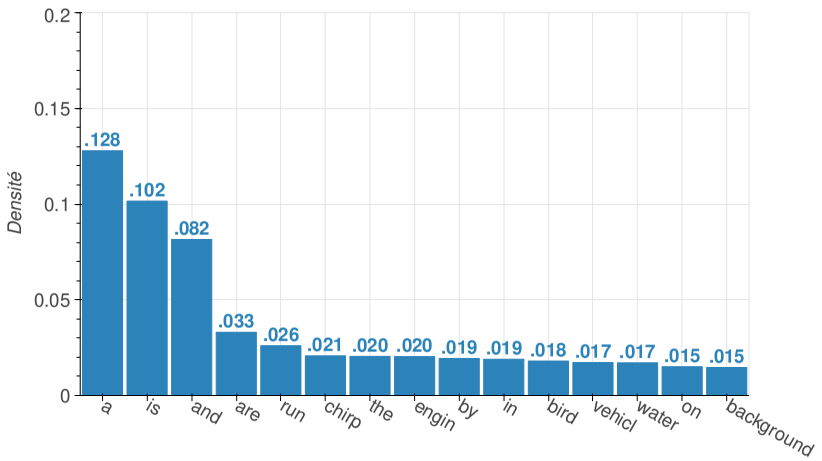
(a) Candidats sur AC-test avec la TE d'AC.

(b) Candidats sur AC-test avec la TE de CL.

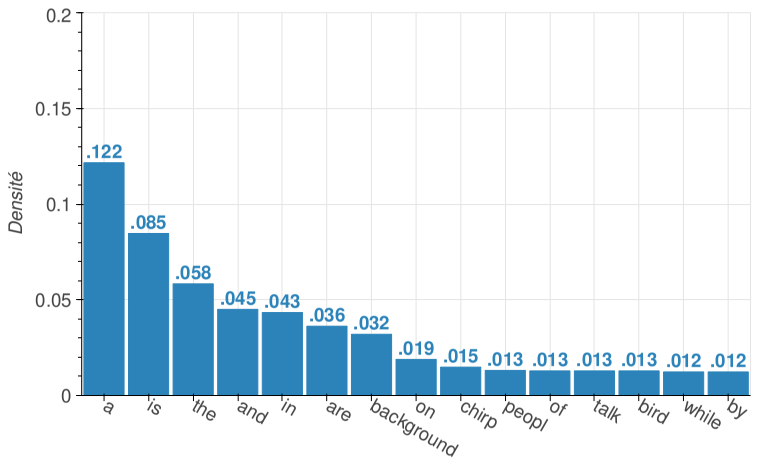
(c) Références sur AC-train.

(d) Références sur AC-test.

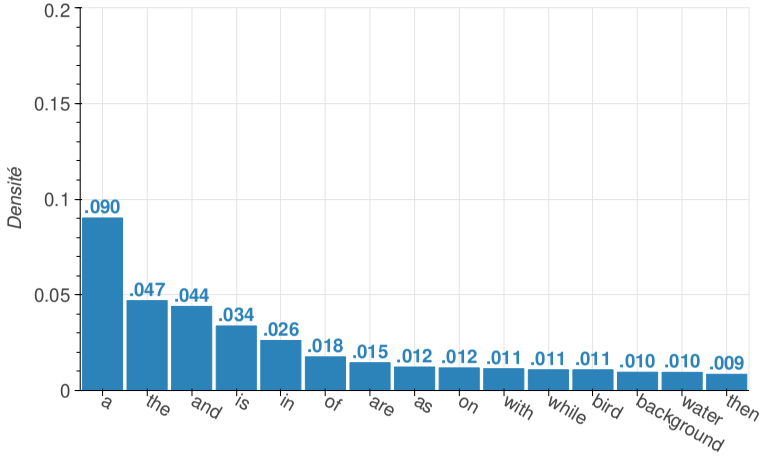
FIGURE H.1 – Distributions des **unigrammes** sur AC-test et AC-train.



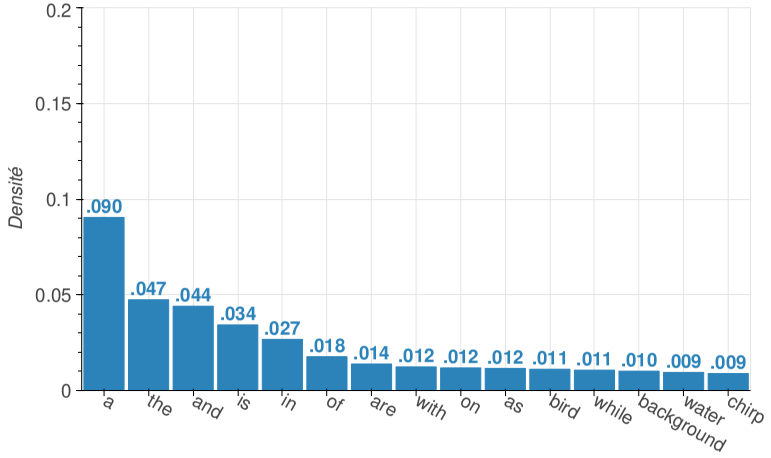
(a) Candidats sur CL-eval avec la TE d'AC.



(b) Candidats sur CL-eval avec la TE de CL.

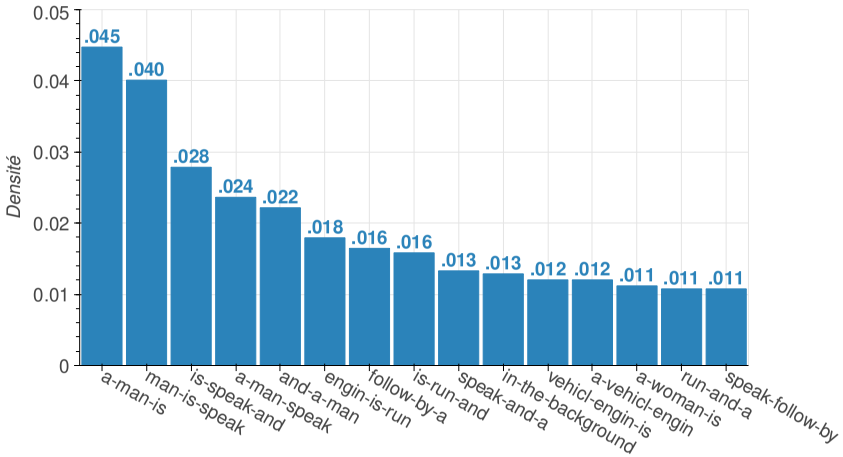


(c) Références sur CL-dev.

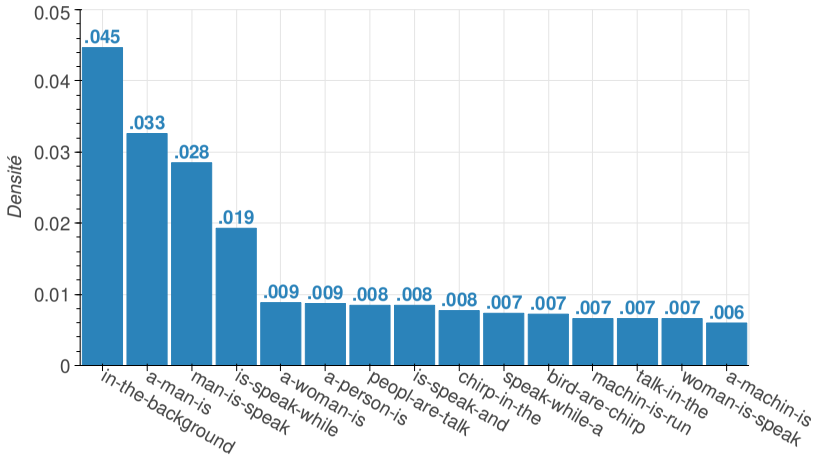


(d) Références sur CL-eval.

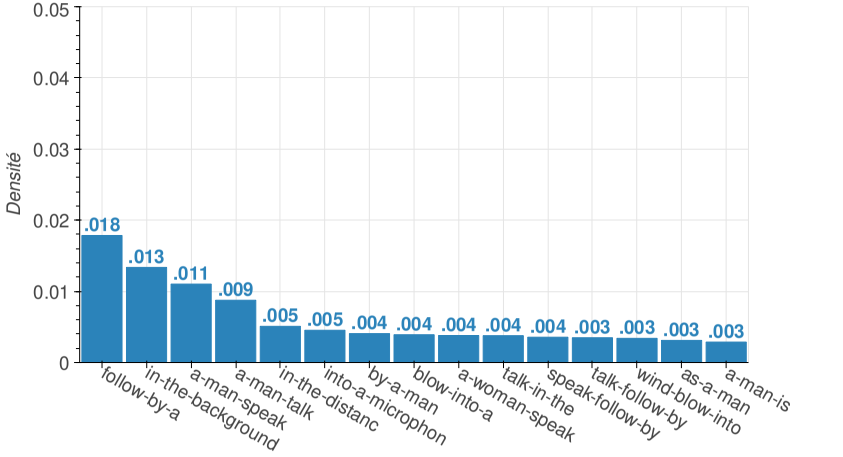
FIGURE H.2 – Distributions des **unigrammes** sur CL-eval et CL-dev.



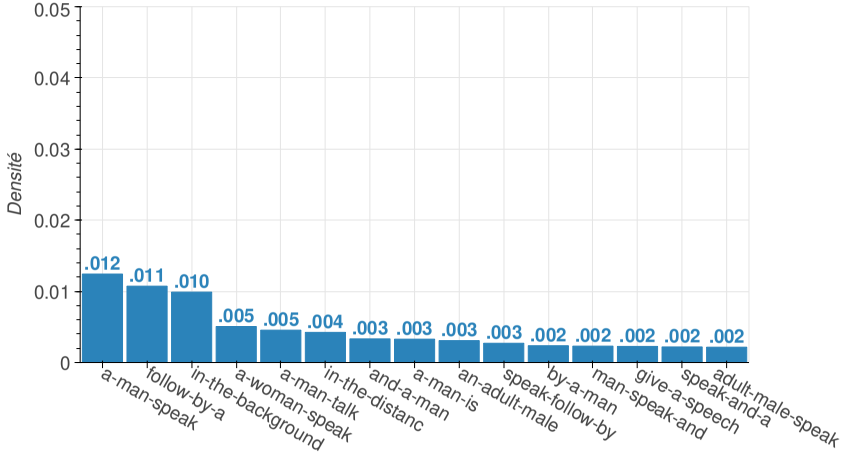
(a) Candidats sur AC-test avec la TE d'AC.



(b) Candidats sur AC-test avec la TE de CL.

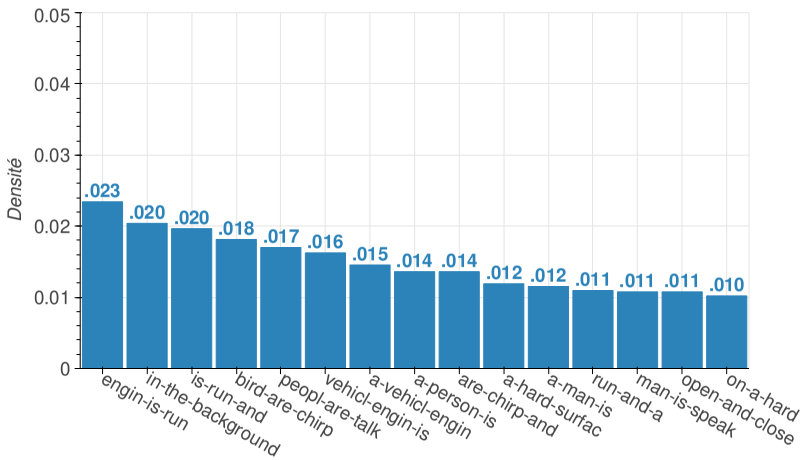


(c) Références sur AC-train.

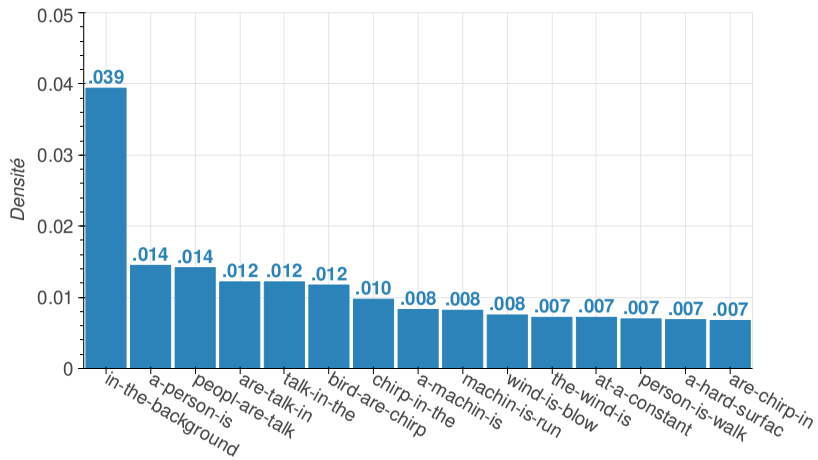


(d) Références sur AC-test.

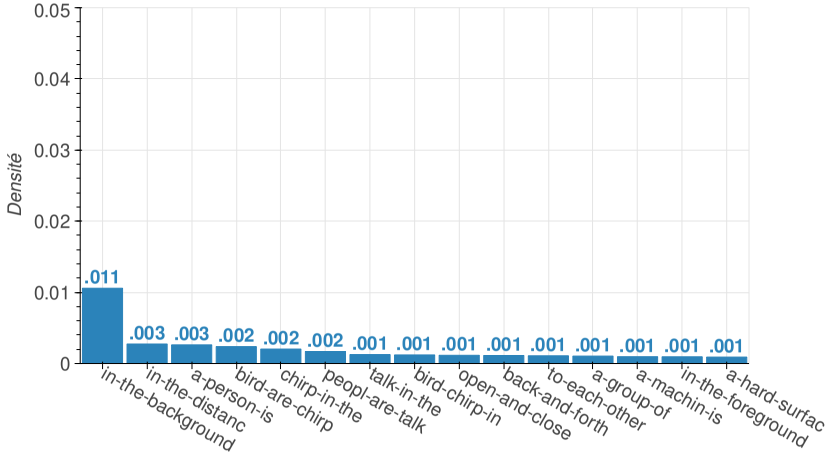
FIGURE H.3 – Distributions des **trigrammes** sur AC-test and AC-train.



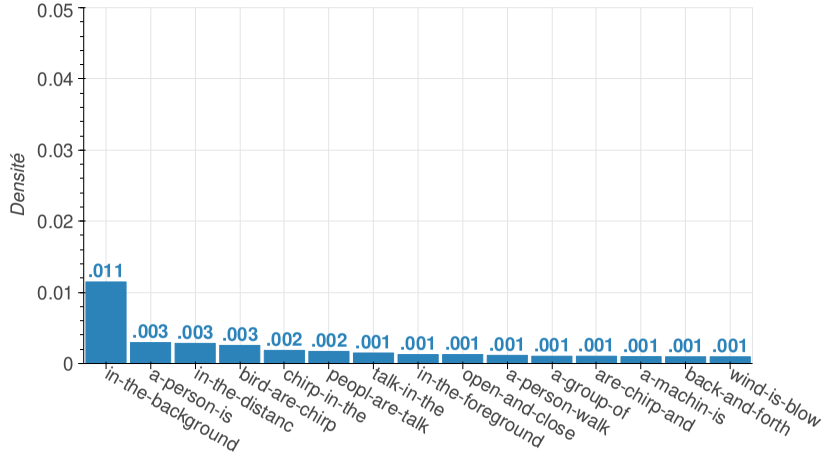
(a) Candidats sur CL-eval avec la TE d'AC.



(b) Candidats sur CL-eval avec la TE de CL.

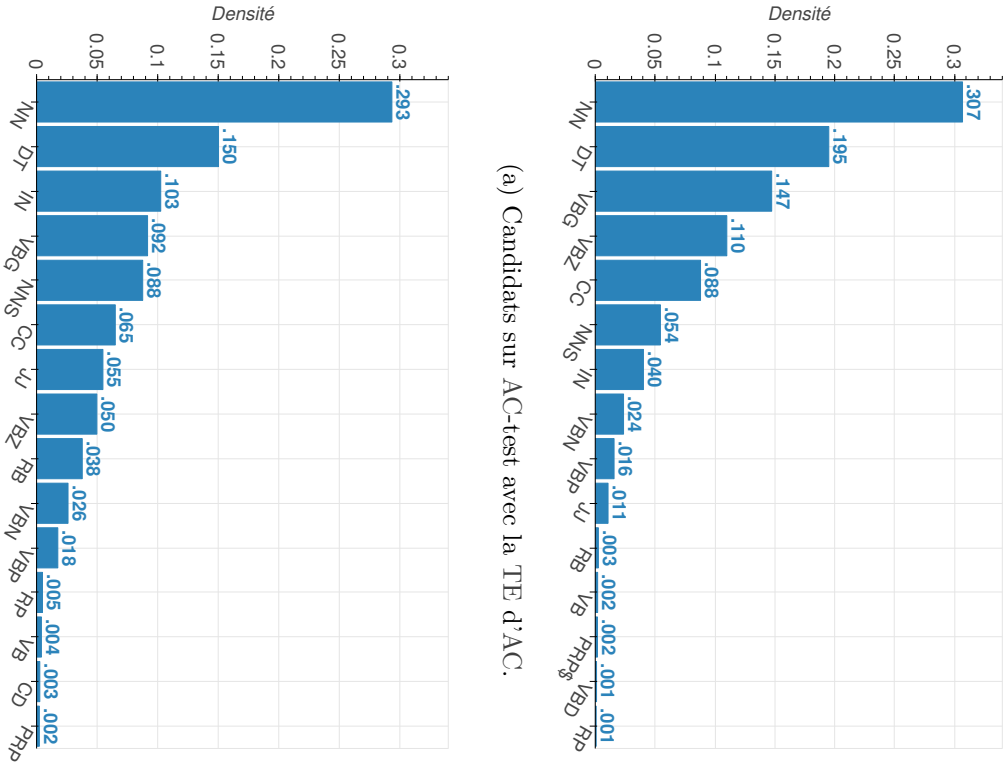


(c) Références sur CL-dev.

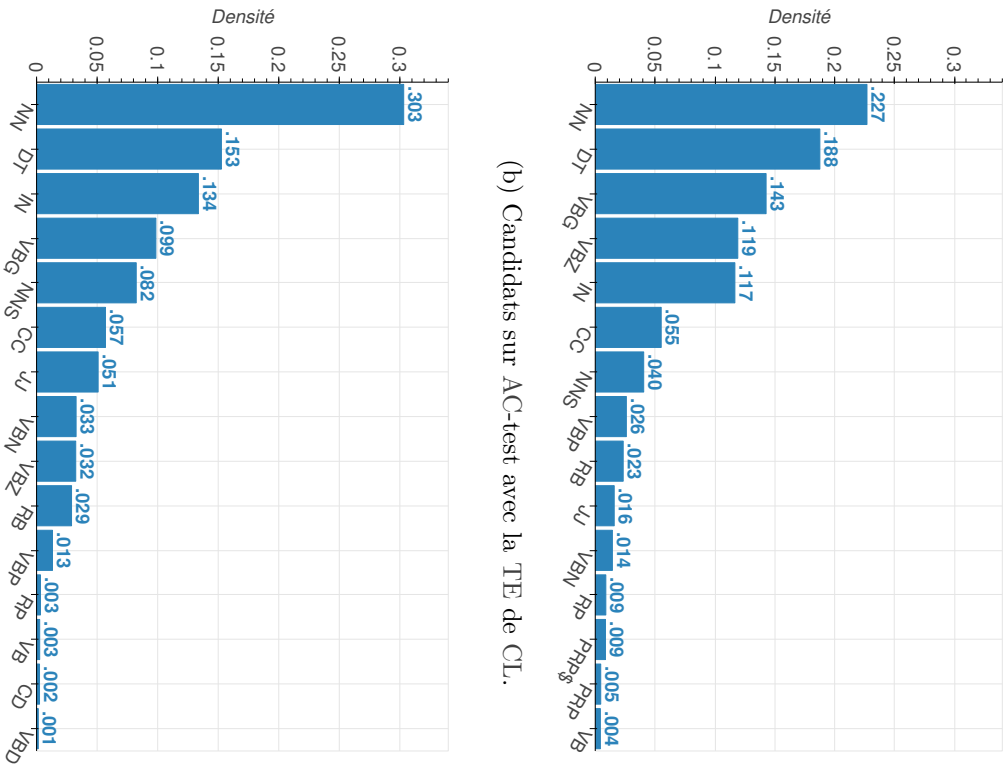


(d) Références sur CL-eval.

FIGURE H.4 – Distributions des **trigrammes** sur CL-eval and CL-dev.



(a) Candidats sur AC-test avec la TE d'AC.

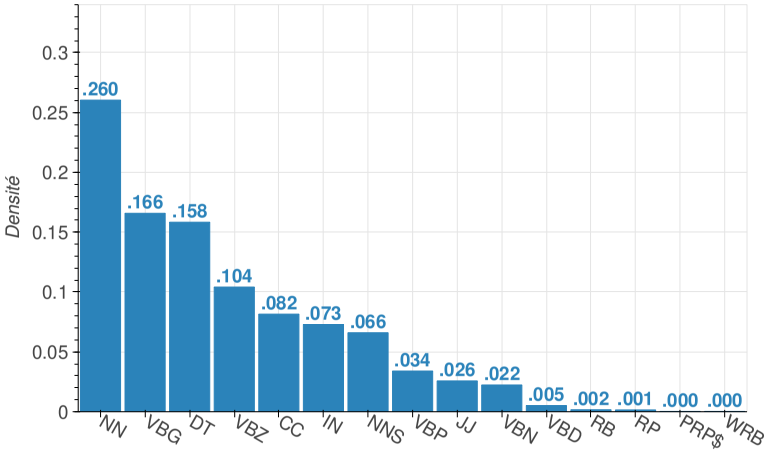


(b) Candidats sur AC-test avec la TE de CL.

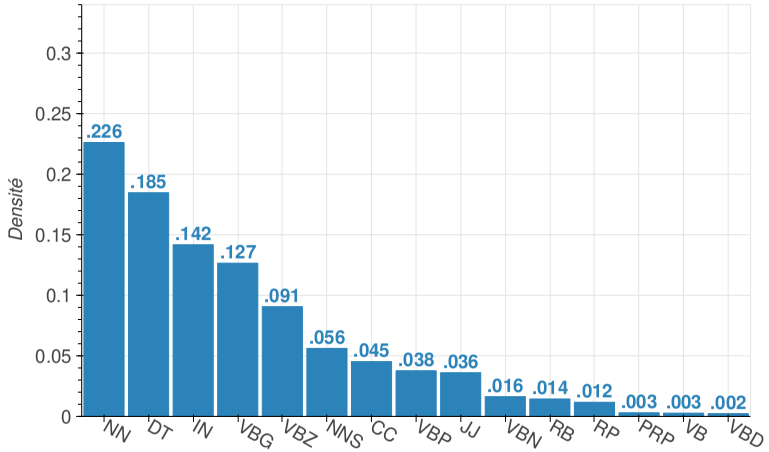
(c) Références sur AC-train.

(d) Références sur AC-test.

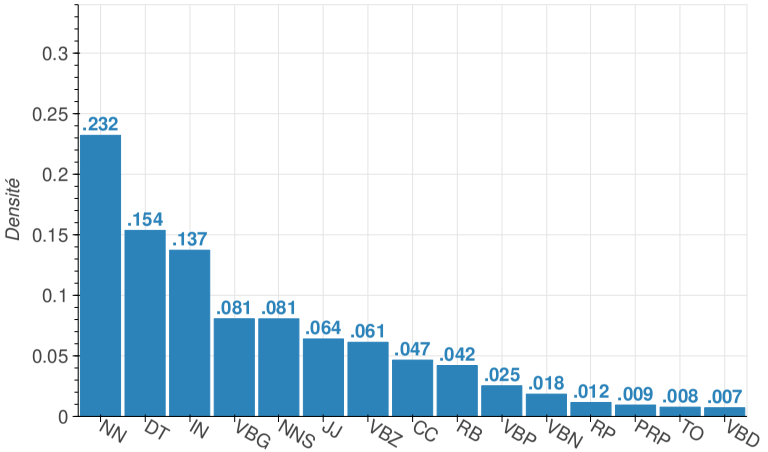
FIGURE H.5 – Distributions des **catégories grammaticales des mots** sur AC-test et AC-train.



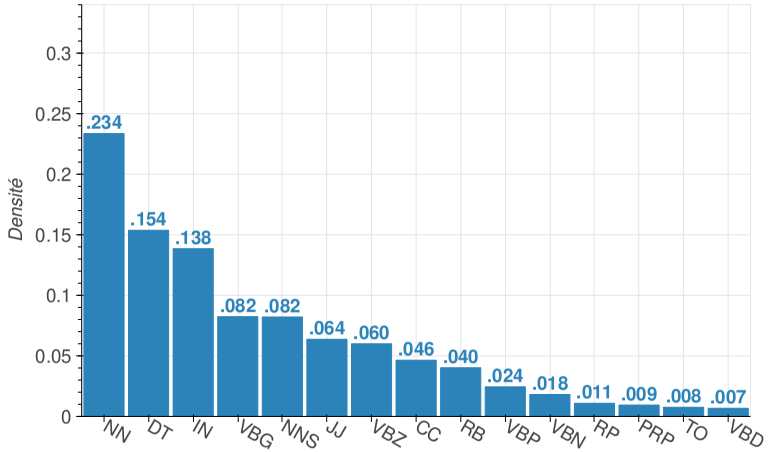
(a) Candidats sur CL-eval avec la TE d'AC.



(b) Candidats sur CL-eval avec la TE de CL.



(c) Références sur CL-dev.



(d) Références sur CL-eval.

FIGURE H.6 – Distributions des **catégories grammaticales des mots** sur CL-eval et CL-dev.

Annexe I

Recouvrements entre les différents jeux de données de DTAA

Cette section contient les détails des recouvrements entre les jeux de données CL avec FSD50K dans le tableau I.1, CL avec WC dans le tableau I.2 et AC avec WC dans le tableau I.3. Le sous-ensemble *« full »* désigne la concaténation de tous les sous-ensembles d'un jeu de données. FSD et AS-SL correspondent aux sous-ensembles de WavCaps issus respectivement de Freesound et de la partie fortement annotée d'AudioSet. Le sous-ensemble *« train2 »* d'AC désigne l'ensemble des données que nous avons corrigées (section 5.6). Chaque valeur correspond à la proportion de sources du jeu de données de la ligne qui sont incluses dans le jeu de données de la colonne.

On retrouve environ 5% des sources de CL dans l'entraînement de FSD50K et 88% des sources de CL sont contenues dans WC. Pour AC, 17% des fichiers audio sont en commun avec WC. Il faut cependant noter qu'il manque les métadonnées des liens sources pour certains fichiers de CL. De plus, on peut également remarquer la présence d'un léger recouvrement entre les sources de l'entraînement, la validation et le test de CL. Cela n'est pas mentionné dans l'étude d'origine, et pourrait donc être une erreur des auteurs sur quelques fichiers.

Table I.1 – Recouvrement entre CL et FSD50K par sous-ensemble en %.

		CL					**FSD50K**			
		dev	**val**	**eval**	**t2a**	**full**	**train**	**val**	**eval**	**full**
CL	**dev**	100	0,05	0,03	0	100	5,51	0,99	3,58	10,08
	val	0,19	100	0,19	0	100	5,10	0,96	2,98	9,05
	eval	0,10	0,19	100	0	100	5,19	1,44	3,17	9,81
	t2a	0	0	0	100	100	7,30	1,30	5,10	13,70
	full	55,48	15,09	15,09	14,47	100	5,66	1,10	3,65	10,41
FSD50K	**train**	0,57	0,14	0,15	0,20	1,06	100	0	0	100
	val	0,91	0,24	0,36	0,31	1,82	0	100	0	100
	eval	1,34	0,30	0,32	0,50	2,46	0	0	100	100
	full	0,75	0,18	0,20	0,27	1,40	71,87	8,15	19,98	100

TABLE I.2 – Recouvrement entre CL et WC par sous-ensemble en %.

		CL					WC	
		dev	**val**	**eval**	**t2a**	**full**	**FSD**	**FSD^{-CL}**
CL	**dev**	100	0,05	0,03	0	100	89,51	0
	val	0,19	100	0,19	0	100	88,26	0
	eval	0,10	0,19	100	0	100	88,17	0
	t2a	0	0	0	100	100	88,60	0
	full	55,48	15,09	15,09	14,47	100	88,99	0
WC	**FSD**	1,31	0,35	0,35	0,34	2,34	100	97,66
	FSD^{-CL}	0	0	0	0	0	100	100

TABLE I.3 – Recouvrement entre AC et WC par sous-ensemble en %.

		AC					WC	
		train	**val**	**test**	**train2**	**full**	**AS-SL**	**AS-SL^{-AC}**
AC	**train**	100	0	0	99,94	100	17,61	0
	val	0	100	0	0	100	17,78	0
	test	0	0	100	0	100	17,74	0
	train2	100	0	0	100	100	17,61	0
	full	98,55	0,49	0,96	98,52	100	17,62	0
WC	**AS-SL**	8,10	0,08	0,16	8,10	8,34	100	91,66
	AS-SL^{-AC}	0	0	0	0	0	100	100

Annexe J

Références des sorties du modèle multilingue

Les tableaux ci-dessous J.1 et J.2 donne les cinq références anglaises pour deux fichiers audio associés aux candidats des tableaux 10.5 et 10.6 du chapitre 10.

Table J.1 – Références anglaises du fichier « `Pb6MqpdX5Jw` », issus d'AC-test.

Références (en)
Clip-clops gallop as the wind blows and thunder cracks.
A horse clip-clops in a windy rain as thunder cracks in the distance.
A horse gallops then trot on grass as gusts of wind blow and thunderclaps in the distance.
Rain falls and distant thunder roars with nearby faint clip-clops of a horse.
A light rain with gentle thunder.

Table J.2 – Références anglaises du fichier « `JTHMXLC9YRs` », issus d'AC-test.

Références (en)
Ducks quack with distant passing traffic.
Wind blowing hard followed by breathing and faint quacks.
Breathing and ducks quaking.
Faint quacking of a duck with some light clicks and rustling.
A person heavily breathing as wood creaks and ducks quack.

Annexe K

SPIDEr-max avec CoNeTTE

Les figures K.1 et K.2 représentent les scores SPIDEr et SPIDEr-max du modèle CoNeTTE (spécialisé pour CL), pour des tailles de faisceau de 1 à 10 sur les deux ensembles de test. Comme nous l'avions pu le constater dans le chapitre 4, les scores de SPIDEr-max surpassent très rapidement les scores SPIDEr de l'état de l'art, avec seulement une taille de faisceau de 2. Cela implique que le score SPIDEr pourrait être radicalement amélioré en mieux sélectionnant le meilleur candidat de la recherche en faisceau, même avec seulement deux phrases différentes. Cependant, cela reste un problème difficile à résoudre automatiquement.

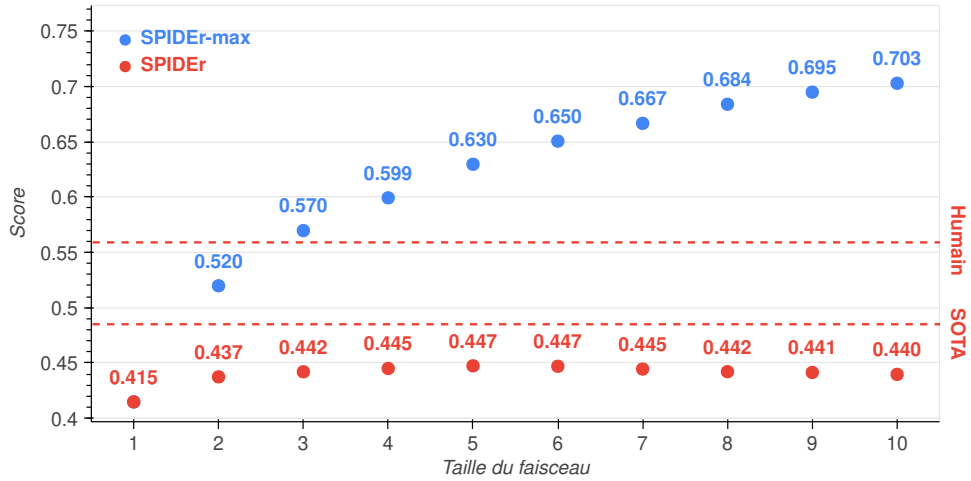


FIGURE K.1 – SPIDEr and SPIDEr-max scores en fonction de différentes tailles de faisceaux sur AC-test, pour le modèle CoNeTTE.

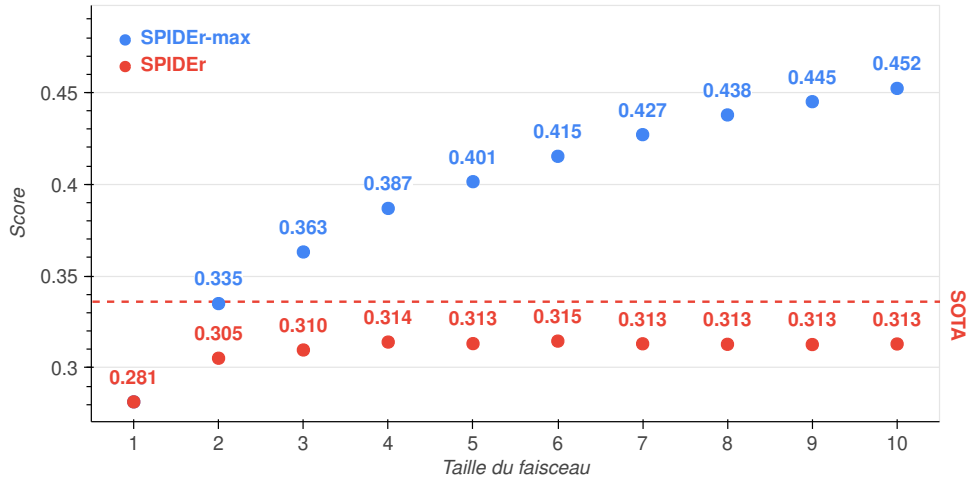


FIGURE K.2 – SPIDEr and SPIDEr-max scores en fonction de différentes tailles de faisceaux sur CL-eval, pour le modèle CoNeTTE.

Ma bibliographie

Journal

[Cancès 2022] Léo Cances, Etienne Labbé et Thomas Pellegrini. Comparison of semi-supervised deep learning algorithms for audio classification. EURASIP Journal on Audio, Speech, and Music Processing, vol. 2022, no. 1, page 23, Sep 2022. URL : https://doi.org/10.1186/s13636-022-00255-6.

Conférences

[Labbé 2023c] Etienne Labbé, Julien Pinquier et Thomas Pellegrini. Multitask Learning in Audio Captioning : A Sentence Embedding Regression Loss Acts as a Regularizer. In 2023 31st European Signal Processing Conference (EUSIPCO), pages 760–764, 2023. URL : https://ieeexplore.ieee.org/document/10290108.

[Pellegrini 2023] Thomas Pellegrini, Ismail Khalfaoui-Hassani, Etienne Labbé et Timothée Masquelier. Adapting a ConvNeXt Model to Audio Classification on AudioSet. In Proc. INTERSPEECH 2023, pages 4169–4173, 2023. URL : https://www.isca-speech.org/archive/pdfs/interspeech_2023/pellegrini23_interspeech.pdf.

Workshops

[Labbé 2022] Etienne Labbé, Thomas Pellegrini et Julien Pinquier. Is my Automatic Audio Captioning System so Bad ? SPIDEr-max : A Metric to Consider Several Caption Candidates. In Proceedings of the 7th Detection and Classification of Acoustic Scenes and Events 2022 Workshop (DCASE2022), Nancy, France, November 2022. URL : https://dcase.community/documents/workshop2022/proceedings/DCASE2022Workshop_Labbe_46.pdf.

[Labbé 2023] Étienne Labbé, Thomas Pellegrini et Julien Pinquier. Killing Two Birds with One Stone : Can an Audio Captioning System Also Be Used for Audio-Text Retrieval ? In Proceedings of the 8th Detection and Classification of Acoustic Scenes and Events 2023 Workshop (DCASE2023), pages 96–100, Tampere, Finland, September 2023. URL : https://dcase.community/documents/workshop2023/proceedings/DCASE2023Workshop_Labb%C3%A9_43.pdf.

Rapports techniques

[Labbé 2021] Etienne Labbé et Thomas Pellegrini. IRIT-UPS DCASE 2021 Audio Captioning System. Rapport technique, DCASE2021 Challenge, July 2021. URL : https://dcase.community/documents/challenge2021/technical_reports/DCASE2021_Labbe_102_t6.pdf.

[Labbé 2022] Etienne Labbé, Thomas Pellegrini et Julien Pinquier. IRIT-UPS DCASE 2022 task6a system : stochastic decoding methods for audio captioning. Rapport technique, DCASE2022 Challenge, July 2022. URL : https://dcase.community/documents/challenge2022/technical_reports/DCASE2022_Labbe_87_t6a.pdf.

[Labbé 2023b] Etienne Labbé, Thomas Pellegrini et Julien Pinquier. IRIT-UPS DCASE 2023 audio captioning and retrieval system. Rapport technique, DCASE2023 Challenge, May 2023. URL : https://dcase.community/documents/challenge2023/technical_reports/DCASE2023_Labbe_59_t6a.pdf.

Prépublications

[Cousin 2023] Matéo Cousin, Étienne Labbé et Thomas Pellegrini. Multilingual Audio Captioning using machine translated data. arXiv preprint arXiv :2309.07615, 2023. URL : https://arxiv.org/pdf/2309.07615.pdf.

[Labbé 2023a] Etienne Labbé, Thomas Pellegrini et Julien Pinquier. CoNeTTE : An efficient Audio Captioning system leveraging multiple datasets with Task Embedding, 2023. URL : https://arxiv.org/pdf/2309.00454.pdf.

Bibliographie

[Abdelnour 2018] Jerome Abdelnour, Giampiero Salvi et Jean Rouat. *CLEAR : A dataset for compositional language and elementary acoustic reasoning*. arXiv preprint arXiv :1811.10561, 2018. URL : https://arxiv.org/pdf/1811.10561.pdf.

[Abu-El-Haija 2016] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Apostol (Paul) Natsev, George Toderici, Balakrishnan Varadarajan et Sudheendra Vijayanarasimhan. *YouTube-8M : A Large-Scale Video Classification Benchmark*. In arXiv :1609.08675, 2016. URL : https://arxiv.org/pdf/1609.08675v1.pdf.

[Agostinelli 2023] Andrea Agostinelli, Timo I. Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, Matt Sharifi, Neil Zeghidour et Christian Frank. *MusicLM : Generating Music From Text*, 2023. URL : https://arxiv.org/pdf/2301.11325.pdf.

[Amini 2022] Massih-Reza Amini, Vasilii Feofanov, Loic Pauletto, Emilie Devijver et Yury Maximov. *Self-Training : A Survey*, 2022. URL : https://arxiv.org/pdf/2202.12040.pdf.

[Anderson 2016] Peter Anderson, Basura Fernando, Mark Johnson et Stephen Gould. *SPICE : Semantic Propositional Image Caption Evaluation*. In Bastian Leibe, Jiri Matas, Nicu Sebe et Max Welling, editeurs, Computer Vision – ECCV 2016, pages 382–398, Cham, 2016. Springer International Publishing. URL : https://panderson.me/images/SPICE.pdf.

[Anderson 2018] Peter Anderson, Stephen Gould et Mark Johnson. *Partially-Supervised Image Captioning*. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi et R. Garnett, editeurs, Advances in Neural Information Processing Systems, volume 31. Curran Associates, Inc., 2018. URL : https://proceedings.neurips.cc/paper_files/paper/2018/file/d2ed45a52bc0edfa11c2064e9edee8bf-Paper.pdf.

[Bachman 2014] Philip Bachman, Ouais Alsharif et Doina Precup. *Learning with Pseudo-Ensembles*, 2014. URL : https://arxiv.org/pdf/1412.4864.pdf.

[Baevski 2020] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed et Michael Auli. *wav2vec 2.0 : A framework for self-supervised learning of speech representations*. Advances in neural information processing systems, vol. 33, pages 12449–12460, 2020. URL : https://proceedings.neurips.cc/paper/2020/file/92d1e1eb1cd6f9fba3227870bb6d7f07-Paper.pdf.

[Bahdanau 2016] Dzmitry Bahdanau, Kyunghyun Cho et Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*, 2016. URL : https://arxiv.org/pdf/1409.0473.pdf.

[Bannard 2005] Colin Bannard et Chris Callison-Burch. *Paraphrasing with Bilingual Parallel Corpora*. In Kevin Knight, Hwee Tou Ng et Kemal Oflazer, editeurs, Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05), pages 597–604, Ann Arbor, Michigan, Juin 2005. Association for Computational Linguistics. URL : https://aclanthology.org/P05-1074.

[Bengio 2013] Yoshua Bengio, Nicholas Léonard et Aaron Courville. *Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation*, 2013. URL : https://arxiv.org/pdf/1308.3432.pdf.

[Bengio 2015] Samy Bengio, Oriol Vinyals, Navdeep Jaitly et Noam Shazeer. *Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks*. In Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, NIPS'15, page 1171–1179, Cambridge, MA, USA, 2015. MIT Press. URL : https://proceedings.neurips.cc/paper/2015/file/e995f98d56967d946471af29d7bf99f1-Paper.pdf.

[Berg 2021] Jan Berg et Konstantinos Drossos. *Continual Learning for Automated Audio Captioning Using the Learning without Forgetting Approach*. In Proceedings of the 6th Detection and Classification of Acoustic Scenes and Events 2021 Workshop (DCASE2021), pages 140–144, Barcelona, Spain, November 2021. URL : https://dcase.community/documents/workshop2021/proceedings/DCASE2021Workshop_Berg_51.pdf.

[Berthelot 2019] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver et Colin Raffel. *MixMatch : A Holistic Approach to Semi-Supervised Learning*. In proc. NeurIPS, pages 5049–5059, Vancouver, 2019. URL : https://proceedings.neurips.cc/paper_files/paper/2019/file/1cd138d0499a68f4bb72bee04bbec2d7-Paper.pdf.

[Berthelot 2021] David Berthelot, Rebecca Roelofs, Kihyuk Sohn, Nicholas Carlini et Alex Kurakin. *AdaMatch : A Unified Approach to Semi-Supervised Learning and Domain Adaptation*. CoRR, vol. abs/2106.04732, 2021. URL : https://arxiv.org/pdf/2106.04732.pdf.

[Bhosale 2023] Swapnil Bhosale, Rupayan Chakraborty et Sunil Kumar Kopparapu. *A Novel Metric For Evaluating Audio Caption Similarity*. In ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 1–5, 2023. URL : https://ieeexplore.ieee.org/abstract/document/10096526.

[Bogdanov 2019] Dmitry Bogdanov, Minz Won, Philip Tovstogan, Alastair Porter et Xavier Serra. *The MTG-Jamendo Dataset for Automatic Music Tagging*. In Machine Learning for Music Discovery Workshop, International Conference on Machine Learning (ICML 2019), Long Beach, CA, United States, 2019. URL : http://hdl.handle.net/10230/42015.

[Bowman 2015] Samuel R. Bowman, Gabor Angeli, Christopher Potts et Christopher D. Manning. *A large annotated corpus for learning natural language inference*. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 632–642, Lisbon, Portugal, Septembre 2015. Association for Computational Linguistics. URL : https://aclanthology.org/D15-1075.

[Chaganty 2018] Arun Chaganty, Stephen Mussmann et Percy Liang. *The price of debiasing automatic metrics in natural language evalaution*. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers), pages 643–653, Melbourne, Australia, Juillet 2018. Association for Computational Linguistics. URL : https://aclanthology.org/P18-1060.

[Chan 2016] William Chan, Navdeep Jaitly, Quoc Le et Oriol Vinyals. *Listen, attend and spell : A neural network for large vocabulary conversational speech recognition*. In 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 4960–4964, 2016. URL : https://ieeexplore.ieee.org/document/7472621.

[Chen 2017] Wenhu Chen, Aurelien Lucchi et Thomas Hofmann. *A Semi-supervised Framework for Image Captioning*, 2017. URL : https://arxiv.org/pdf/1611.05321.pdf.

[Chen 2020] Honglie Chen, Weidi Xie, Andrea Vedaldi et Andrew Zisserman. *Vggsound : A Large-Scale Audio-Visual Dataset*. In ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 721–725, 2020. URL : https://ieeexplore.ieee.org/document/9053174.

[Chen 2022a] Chen Chen, Nana Hou, Yuchen Hu, Heqing Zou, Xiaofeng Qi et Chng Eng Siong. *Interactive Audio-text Representation for Automated Audio Captioning with Contrastive Learning*. In Interspeech, 2022. URL : https://api.semanticscholar.org/CorpusID:247779277.

[Chen 2022b] Ke Chen, Xingjian Du, Bilei Zhu, Zejun Ma, Taylor Berg-Kirkpatrick et Shlomo Dubnov. *HTS-AT : A Hierarchical Token-Semantic Audio Transformer for Sound Classification and Detection*. In ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 646–650, 2022. URL : https://ieeexplore.ieee.org/document/9746312.

[Chen 2022c] Sanyuan Chen, Yu Wu, Chengyi Wang, Shujie Liu, Daniel Tompkins, Zhuo Chen et Furu Wei. *BEATs : Audio Pre-Training with Acoustic Tokenizers*, 2022. URL : https://arxiv.org/pdf/2212.09058.pdf.

[Chen 2023a] Hao Chen, Ran Tao, Yue Fan, Yidong Wang, Jindong Wang, Bernt Schiele, Xing Xie, Bhiksha Raj et Marios Savvides. *SoftMatch : Addressing the Quantity-Quality Trade-off in Semi-supervised Learning*, 2023. URL : https://arxiv.org/pdf/2301.10921.pdf.

[Chen 2023b] Sihan Chen, Xingjian He, Longteng Guo, Xinxin Zhu, Weining Wang, Jinhui Tang et Jing Liu. *VALOR : Vision-Audio-Language Omni-Perception Pretraining Model and Dataset*, 2023. URL : https://arxiv.org/pdf/2304.08345.pdf.

[Chen 2023c] Sihan Chen, Handong Li, Qunbo Wang, Zijia Zhao, Mingzhen Sun, Xinxin Zhu et Jing Liu. *VAST : A Vision-Audio-Subtitle-Text Omni-Modality Foundation Model and Dataset*, 2023. URL : https://arxiv.org/pdf/2305.18500.pdf.

[Chiang 2023] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica et Eric P. Xing. *Vicuna : An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality*, March 2023. URL : https://lmsys.org/blog/2023-03-30-vicuna/.

[Cho 2014] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk et Yoshua Bengio. *Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation*. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1724–1734, Doha, Qatar, Octobre 2014. Association for Computational Linguistics. URL : https://aclanthology.org/D14-1179.

[Cho 2023] Jae-Heung Cho, Yoon-Ah Park, Jaewon Kim et Joon-Hyuk Chang. *HYU submission for the DCASE 2023 task 6a : automated audio captioning model using AL-MixGen and synonyms substitution*. Rapport technique, DCASE2023 Challenge, May 2023. URL : https://dcase.community/documents/challenge2023/technical_reports/DCASE2023_Chang_113_t6a.pdf.

[Costa-jussà 2022] Marta R Costa-jussà, James Cross*et al.* *No language left behind : Scaling human-centered machine translation*. arXiv preprint arXiv :2207.04672, 2022. URL : https://arxiv.org/ftp/arxiv/papers/2207/2207.04672.pdf.

[Denkowski 2014] Michael Denkowski et Alon Lavie. *Meteor Universal : Language Specific Translation Evaluation for Any Target Language*. In Proceedings of the Ninth Workshop

on Statistical Machine Translation, pages 376–380, Baltimore, Maryland, USA, 2014. Association for Computational Linguistics. URL : http://aclweb.org/anthology/W14-3348.

[Deshmukh 2022] Soham Deshmukh, Benjamin Elizalde et Huaming Wang. *Audio Retrieval with WavText5K and CLAP Training*. arXiv preprint arXiv :2209.14275, 2022. URL : https://arxiv.org/pdf/2209.14275.pdf.

[Deshmukh 2023] Soham Deshmukh, Benjamin Elizalde, Dimitra Emmanouilidou, Bhiksha Raj, Rita Singh et Huaming Wang. *Training Audio Captioning Models without Audio*, 2023. URL : https://browse.arxiv.org/pdf/2309.07372v1.pdf.

[Devlin 2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee et Kristina Toutanova. *BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding*. CoRR, vol. abs/1810.04805, 2018. URL : http://arxiv.org/pdf/1810.04805.pdf.

[DeVries 2017] Terrance DeVries et Graham W. Taylor. *Improved Regularization of Convolutional Neural Networks with Cutout*, 2017. URL : https://arxiv.org/pdf/1708.04552.pdf.

[Dosovitskiy 2021] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit et Neil Houlsby. *An Image is Worth 16x16 Words : Transformers for Image Recognition at Scale*. In 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net, 2021. URL : https://openreview.net/forum?id=YicbFdNTTy.

[Drossos 2017] K. Drossos, S. Adavanne et T. Virtanen. *Automated audio captioning with recurrent neural networks*. In 2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), pages 374–378, 2017. URL : https://ieeexplore.ieee.org/abstract/document/8170058.

[Drossos 2020] Konstantinos Drossos, Samuel Lipping et Tuomas Virtanen. *Clotho : an Audio Captioning Dataset*. In Proc. IEEE Int. Conf. Acoustic., Speech and Signal Process. (ICASSP), pages 736–740, 2020. URL : https://ieeexplore.ieee.org/document/9052990.

[Elliott 2014] Desmond Elliott et Frank Keller. *Comparing Automatic Evaluation Measures for Image Description*. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2 : Short Papers), pages 452–457, Baltimore, Maryland, Juin 2014. Association for Computational Linguistics. URL : https://aclanthology.org/P14-2074.

[Emmanouilidou 2023] Dimitra Emmanouilidou. *Investigations in Audio Captioning : Addressing Vocabulary Imbalance and Evaluating Suitability of Language-Centric Performance Metrics*. In European Signal Processing Conference (EUSIPCO). European Association for Signal Processing (EURASIP), May 2023. URL : https://www.microsoft.com/en-us/research/uploads/prod/2023/05/Kothinti_2022_INVESTIGATIONS-IN-AUDIO-CAPTIONING-ADDRESSING-VOCABULARY-IMBALANCE.pdf.

[Eren 2020a] Aysegül Özkaya Eren et Mustafa Sert. *Audio Captioning Based on Combined Audio and Semantic Embeddings*. In 2020 IEEE International Symposium on Multimedia (ISM), pages 41–48, 2020. URL : https://ieeexplore.ieee.org/abstract/document/9327916.

[Eren 2020b] Aysegül Özkaya Eren et Mustafa Sert. *Audio Captioning using Gated Recurrent Units*. ArXiv, vol. abs/2006.03391, 2020. URL : https://api.semanticscholar.org/CorpusID:219401790.

[Eren 2023] Ayşegül Özkaya Eren et Mustafa Sert. *Automated Audio Captioning With Topic Modeling*. IEEE Access, vol. 11, pages 4983–4991, 2023. URL : https://ieeexplore.ieee.org/document/10015020.

[Fan 2018] Angela Fan, Mike Lewis et Yann Dauphin. *Hierarchical Neural Story Generation*. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers), pages 889–898, Melbourne, Australia, Juillet 2018. Association for Computational Linguistics. URL : https://aclanthology.org/P18-1082.

[Fan 2021] Yue Fan, Anna Kukleva et Bernt Schiele. *Revisiting Consistency Regularization for Semi-Supervised Learning*, 2021. URL : https://arxiv.org/pdf/2112.05825.pdf.

[Fonseca 2021] Eduardo Fonseca, Xavier Favory, Jordi Pons, Frederic Font et Xavier Serra. *FSD50K : An Open Dataset of Human-Labeled Sound Events*. IEEE/ACM Trans. Audio, Speech and Lang. Proc., vol. 30, page 829–852, dec 2021. URL : https://doi.org/10.1109/TASLP.2021.3133208.

[Gage 1994] Philip Gage. *A New Algorithm for Data Compression*. C Users J., vol. 12, no. 2, page 23–38, feb 1994. URL : https://unideveloperco.com/wp-content/uploads/2021/12/BPE_Gage_UDC.pdf.

[Gazneli 2022] Avi Gazneli, Gadi Zimerman, Tal Ridnik, Gilad Sharir et Asaf Noy. *End-to-End Audio Strikes Back : Boosting Augmentations Towards An Efficient Audio Classification Network*, 2022. URL : https://arxiv.org/pdf/2204.11479.pdf.

[Geman 1992] Stuart Geman, Elie Bienenstock et René Doursat. *Neural Networks and the Bias/Variance Dilemma*. Neural Computation, vol. 4, pages 1–58, 01 1992. URL : https://www.dam.brown.edu/people/documents/bias-variance.pdf.

[Gemmeke 2017] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal et Marvin Ritter. *Audio Set : An ontology and human-labeled dataset for audio events*. In 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 776–780, New Orleans, LA, Mars 2017. IEEE. URL : http://ieeexplore.ieee.org/document/7952261/.

[Gidaris 2018] Spyros Gidaris, Praveer Singh et Nikos Komodakis. *Unsupervised Representation Learning by Predicting Image Rotations*, 2018. URL : https://arxiv.org/pdf/1803.07728.pdf.

[Gong 2020] Chengyue Gong, Dilin Wang et Qiang Liu. *AlphaMatch : Improving Consistency for Semi-supervised Learning with Alpha-divergence*, 2020. URL : https://arxiv.org/pdf/2011.11779.pdf.

[Gong 2021] Yuan Gong, Yu-An Chung et James Glass. *AST : Audio Spectrogram Transformer*. In Proc. Interspeech 2021, pages 571–575, 2021. URL : https://www.isca-speech.org/archive/pdfs/interspeech_2021/gong21b_interspeech.pdf.

[Gong 2023] Yuan Gong, Sameer Khurana, Leonid Karlinsky et James Glass. *Whisper-AT : Noise-Robust Automatic Speech Recognizers are Also Strong General Audio Event Taggers*. In Proc. INTERSPEECH 2023, pages 2798–2802, 2023. URL : https://www.isca-speech.org/archive/pdfs/interspeech_2023/gong23d_interspeech.pdf.

[Gontier 2021] Felix Gontier, Romain Serizel et Christophe Cerisara. *Automated Audio Captioning by Fine-Tuning BART with AudioSet Tags*. In Proceedings of the 6th Detection and Classification of Acoustic Scenes and Events 2021 Workshop (DCASE2021), pages 170–174, Barcelona, Spain, November 2021. URL : https://dcase.community/documents/workshop2021/proceedings/DCASE2021Workshop_Gontier_57.pdf.

[Gontier 2023] Félix Gontier, Romain Serizel et Christophe Cerisara. *SPICE+ : Evaluation of Automatic Audio Captioning Systems with Pre-Trained Language Models*. In ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 1–5, 2023. URL : https://ieeexplore.ieee.org/document/10097021.

[Graves 2006] Alex Graves, Santiago Fernández, Faustino Gomez et Jürgen Schmidhuber. *Connectionist temporal classification : labelling unsegmented sequence data with recurrent neural networks*. In Proceedings of the 23rd international conference on Machine learning, pages 369–376, 2006. URL : https://www.cs.toronto.edu/~graves/icml_2006.pdf.

[Grootendorst 2022] Maarten Grootendorst. *BERTopic : Neural topic modeling with a class-based TF-IDF procedure*, 2022. URL : https://arxiv.org/pdf/2203.05794.pdf.

[Guzhov 2020] Andrey Guzhov, Federico Raue, Jörn Hees et Andreas Dengel. *ESResNet : Environmental Sound Classification Based on Visual Domain Models*, 2020. URL : https://arxiv.org/pdf/2004.07301.pdf.

[Guzhov 2021] Andrey Guzhov, Federico Raue, Jörn Hees et Andreas R. Dengel. *ESResNe(X)t-fbsp : Learning Robust Time-Frequency Transformation of Audio*. 2021 International Joint Conference on Neural Networks (IJCNN), pages 1–8, 2021. URL : https://api.semanticscholar.org/CorpusID:233388010.

[Hamming 1962] Richard Hamming. Numerical methods for scientists and engineers. New York, 1962. URL : http://tvhdh.vnio.org.vn:8080/xmlui/handle/123456789/11184.

[Han 2021] Qichen Han, Weiqiang Yuan, Dong Liu, Xiang Li et Zhen Yang. *Automated Audio Captioning with Weakly Supervised Pre-Training and Word Selection Methods*. In Proceedings of the 6th Detection and Classification of Acoustic Scenes and Events 2021 Workshop (DCASE2021), pages 6–10, Barcelona, Spain, November 2021. URL : https://dcase.community/documents/workshop2021/proceedings/DCASE2021Workshop_Han_9.pdf.

[Hao 2023] Xiaoshuai Hao, Yi Zhu, Srikar Appalaraju, Aston Zhang, Wanqian Zhang, Bo Li et Mu Li. *MixGen : A New Multi-Modal Data Augmentation*. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) Workshops, pages 379–389, January 2023. URL : https://openaccess.thecvf.com/content/WACV2023W/Pretrain/papers/Hao_MixGen_A_New_Multi-Modal_Data_Augmentation_WACVW_2023_paper.pdf.

[Hendrycks 2016] Dan Hendrycks et Kevin Gimpel. *Gaussian Error Linear Units (GELUs)*, 2016. URL : https://arxiv.org/pdf/1606.08415.pdf.

[Hershey 2017] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, R. Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron J. Weiss et Kevin Wilson. *CNN architectures for large-scale audio classification*. In 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 131–135, 2017. URL : https://ieeexplore.ieee.org/document/7952132.

[Hinton 2012a] Geoffrey Hinton. *Lecture notes in Neural Networks for Machine Learning*, 2012. URL : https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

[Hinton 2012b] Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath et Brian Kingsbury. *Deep Neural Networks for Acoustic Modeling in Speech Recognition : The Shared Views of Four Research Groups*. IEEE Signal Processing Magazine, vol. 29,

no. 6, pages 82–97, 2012. URL : https://ieeexplore.ieee.org/abstract/document/6296526.

[Hochreiter 1997] Sepp Hochreiter et Jürgen Schmidhuber. *Long Short-Term Memory*. Neural Computation, vol. 9, no. 8, pages 1735–1780, 1997. URL : https://ieeexplore.ieee.org/abstract/document/6795963.

[Hodosh 2013] Micah Hodosh, Peter Young et Julia Hockenmaier. *Framing Image Description as a Ranking Task : Data, Models and Evaluation Metrics*. J. Artif. Int. Res., vol. 47, no. 1, page 853–899, may 2013. URL : https://www.jair.org/index.php/jair/article/view/10833/25854.

[Holtzman 2020a] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes et Yejin Choi. *The Curious Case of Neural Text Degeneration*. In International Conference on Learning Representations, 2020. URL : https://openreview.net/forum?id=rygGQyrFvH.

[Holtzman 2020b] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes et Yejin Choi. *The Curious Case of Neural Text Degeneration.* In ICLR. OpenReview.net, 2020. URL : http://dblp.uni-trier.de/db/conf/iclr/iclr2020.html#HoltzmanBDFC20.

[Hsu 2021] Wei-Ning Hsu, David Harwath, Tyler Miller, Christopher Song et James Glass. *Text-Free Image-to-Speech Synthesis Using Learned Segmental Units*. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1 : Long Papers), pages 5284–5300, Online, Août 2021. Association for Computational Linguistics. URL : https://aclanthology.org/2021.acl-long.411.

[Huang 2016] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra et Kilian Q Weinberger. *Deep networks with stochastic depth*. In Proc. ECCV, pages 646–661, Amsterdam, 2016. Springer. URL : https://link.springer.com/content/pdf/10.1007/978-3-319-46493-0_39.pdf.

[Huang 2022] Tianyang Huang, Chaofan Pan, Wenyao Chen, Chenyang Zhu, Shengchen Li et Xi Shao. *Audio captioning using pre-trained model and data augmentation*. Rapport technique, DCASE2022 Challenge, July 2022. URL : https://dcase.community/documents/challenge2022/technical_reports/DCASE2022_Pan_62_t6a.pdf.

[Ikawa 2019] Shota Ikawa et Kunio 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), pages 99–103, New York University, NY, USA, October 2019. URL : https://dcase.community/documents/workshop2019/proceedings/DCASE2019Workshop_Ikawa_82.pdf.

[Isabelle 2018] Pierre Isabelle et Roland Kuhn. *A Challenge Set for French –> English Machine Translation*. arXiv preprint arXiv :1806.02725, 2018. URL : https://arxiv.org/pdf/1806.02725.pdf.

[Jain 2021] Arjit Jain, Pranay Reddy Samala, Preethi Jyothi, Deepak Mittal et Maneesh Kumar Singh. *Perturb, Predict & Paraphrase : Semi-Supervised Learning using Noisy Student for Image Captioning.* In IJCAI, pages 758–764, 2021. URL : https://www.ijcai.org/proceedings/2021/0105.pdf.

[Jang 2017] Eric Jang, Shixiang Gu et Ben Poole. *Categorical Reparameterization with Gumbel-Softmax*. In International Conference on Learning Representations, 2017. URL : https://openreview.net/forum?id=rkE3y85ee.

[Jiang 2022] Yangbangyan Jiang, Xiaodan Li, Yuefeng Chen, Yuan He, Qianqian Xu, Zhiyong Yang, Xiaochun Cao et Qingming Huang. *MaxMatch : Semi-Supervised Learning With Worst-Case Consistency*. IEEE Transactions on Pattern Analysis and Machine Intelligence, pages 1–27, 2022. URL : https://doi.org/10.1109%2Ftpami.2022.3208419.

[Jin 2023] Chuanyang Jin. *Semi-Supervised Image Captioning with CLIP*, 2023. URL : https://arxiv.org/pdf/2306.15111.pdf.

[Kadlčík 2023] Marek Kadlčík, Adam Hájek, Jürgen Kieslich et Radosław Winiecki. *A Whisper transformer for audio captioning trained with synthetic captions and transfer learning*. Rapport technique, DCASE2023 Challenge, May 2023. URL : https://dcase.community/documents/challenge2023/technical_reports/DCASE2023_Kadlcik_68_t6a.pdf.

[Kicinski 2022] Dawid Kicinski, Teodor Lamort de Gail et Pawel Bujnowski. *Exploring audio captioning with keyword-guided text generation*. Rapport technique, DCASE2022 Challenge, July 2022. URL : https://dcase.community/documents/challenge2022/technical_reports/DCASE2022_Kicinski_115_t6a.pdf.

[Kim 2019a] Chris Dongjoo Kim, Byeongchang Kim, Hyunmin Lee et Gunhee 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, Volume 1 (Long and Short Papers), pages 119–132, Minneapolis, Minnesota, Juin 2019. Association for Computational Linguistics. URL : https://aclanthology.org/N19-1011.

[Kim 2019b] Dong-Jin Kim, Jinsoo Choi, Tae-Hyun Oh et In So Kweon. *Image Captioning with Very Scarce Supervised Data : Adversarial Semi-Supervised Learning Approach*. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Association for Computational Linguistics, 2019. URL : https://doi.org/10.18653%2Fv1%2Fd19-1208.

[Kim 2021] Baekseung Kim, Hyejin Won, Il-Youp Kwak et Changwon 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), pages 221–225, Barcelona, Spain, November 2021. URL : https://dcase.community/documents/workshop2021/proceedings/DCASE2021Workshop_Kim_70.pdf.

[Kim 2022] Eungbeom Kim, Jinhee Kim, Yoori Oh, Kyungsu Kim, Minju Park, Jaeheon Sim, Jinwoo Lee et Kyogu Lee. *Improving Audio-Language Learning with MixGen and Multi-Level Test-Time Augmentation*, 2022. URL : https://arxiv.org/pdf/2210.17143.pdf.

[Kim 2023a] Jaewon Kim, Yoon-Ah Park, Jae-Heung Cho et Joon-Hyuk Chang. *Improving Automated Audio Captioning Fluency Through Data Augmentation and Ensemble Selection*. In Proceedings of the 8th Detection and Classification of Acoustic Scenes and Events 2023 Workshop (DCASE2023), pages 86–90, Tampere, Finland, September 2023. URL : https://dcase.community/documents/workshop2023/proceedings/DCASE2023Workshop_Kim_61.pdf.

[Kim 2023b] Minkyu Kim, Kim Sung-Bin et Tae-Hyun Oh. *Prefix Tuning for Automated Audio Captioning*. In ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 1–5, 2023. URL : https://ieeexplore.ieee.org/document/10096877.

[Kingma 2015] Diederik P. Kingma et Jimmy Ba. *Adam : A Method for Stochastic Optimization*. In Yoshua Bengio et Yann LeCun, editeurs, 3rd International Conference on Learning

Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015. URL : http://arxiv.org/pdf/1412.6980.pdf.

[Klein 2003] Dan Klein et Christopher D. Manning. *Accurate unlexicalized parsing*. In Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - ACL '03, volume 1, pages 423–430, Sapporo, Japan, 2003. Association for Computational Linguistics. URL : http://portal.acm.org/citation.cfm?doid=1075096.1075150.

[Ko 2015] Tom Ko, Vijayaditya Peddinti, Daniel Povey et Sanjeev Khudanpur. *Audio augmentation for speech recognition*. In Proc. Interspeech, pages 3586–3589, Dresden, 2015. URL : https://www.isca-speech.org/archive/pdfs/interspeech_2015/ko15_interspeech.pdf.

[Koepke 2022] A. Sophia Koepke, Andreea-Maria Oncescu, Joao Henriques, Zeynep Akata et Samuel Albanie. *Audio Retrieval with Natural Language Queries : A Benchmark Study*. IEEE Transactions on Multimedia, pages 1–1, 2022. URL : https://ieeexplore.ieee.org/abstract/document/9707629.

[Koh 2022] Andrew Koh, Xue Fuzhao et Chng Eng Siong. *Automated Audio Captioning Using Transfer Learning and Reconstruction Latent Space Similarity Regularization*. In ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 7722–7726, 2022. URL : https://ieeexplore.ieee.org/document/9747676.

[Koizumi 2020a] Y. Koizumi, R. Masumura, K. Nishida, M. Yasuda et S. Saito. *A Transformer-based Audio Captioning Model with Keyword Estimation*. In INTERSPEECH 2020, Oct. 2020. URL : http://www.interspeech2020.org/uploadfile/pdf/Wed-1-2-8.pdf.

[Koizumi 2020b] Yuma Koizumi, Yasunori Ohishi, Daisuke Niizumi, Daiki Takeuchi et Masahiro Yasuda. *Audio Captioning using Pre-Trained Large-Scale Language Model Guided by Audio-based Similar Caption Retrieval*. ArXiv, vol. abs/2012.07331, 2020. URL : https://api.semanticscholar.org/CorpusID:229157309.

[Koizumi 2020c] Yuma Koizumi, Daiki Takeuchi, Yasunori Ohishi, Noboru Harada et Kunio Kashino. *The NTT DCASE2020 Challenge Task 6 System : Automated Audio Captioning With Keywords and Sentence Length Estimation*. Rapport technique, DCASE2020 Challenge, June 2020. URL : https://dcase.community/documents/challenge2020/technical_reports/DCASE2020_Koizumi_63_t6.pdf.

[Komatsu 2023] Tatsuya Komatsu, Yusuke Fujita, Kazuya Takeda et Tomoki Toda. *Audio Difference Learning for Audio Captioning*, 2023. URL : https://arxiv.org/pdf/2309.08141.pdf.

[Kong 2020] Qiuqiang Kong, Yin Cao, Turab Iqbal, Yuxuan Wang et Mark Plumbley. *PANNs : Large-Scale Pretrained Audio Neural Networks for Audio Pattern Recognition*. IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 28, pages 2880–2894, 01 2020. URL : https://www.researchgate.net/publication/347217099_PANNs_Large-Scale_Pretrained_Audio_Neural_Networks_for_Audio_Pattern_Recognition.

[Koutini 2022] Khaled Koutini, Jan Schlüter, Hamid Eghbal-zadeh et Gerhard Widmer. *Efficient Training of Audio Transformers with Patchout*. In Interspeech 2022, 23rd Annual Conference of the International Speech Communication Association, Incheon, Korea, 18-22 September 2022, pages 2753–2757. ISCA, 2022. URL : https://doi.org/10.21437/Interspeech.2022-227.

[Kouzelis 2022] Thodoris Kouzelis, Grigoris Bastas, Athanasios Katsamanis et Alexandros Potamianos. *Efficient audio captioning transformer with patchout and text guidance*. Rapport technique, DCASE2022 Challenge, July 2022. URL : https://dcase.community/documents/challenge2022/technical_reports/DCASE2022_Kouzelis_60_t6a.pdf.

[Kouzelis 2023] Theodoros Kouzelis et Vassilis Katsouros. *Weakly-Supervised Automated Audio Captioning via Text Only Training*. In Proceedings of the 8th Detection and Classification of Acoustic Scenes and Events 2023 Workshop (DCASE2023), pages 91–95, Tampere, Finland, September 2023. URL : https://dcase.community/documents/workshop2023/proceedings/DCASE2023Workshop_Kouzelis_16.pdf.

[Krishna 2017] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma*et al.* *Visual genome : Connecting language and vision using crowdsourced dense image annotations*. International journal of computer vision, vol. 123, pages 32–73, 2017. URL : https://link.springer.com/article/10.1007/s11263-016-0981-7.

[Krishnan 2021] Ashwath Krishnan, Sudhanva Rajesh et Shylaja SS. *Text-based Image Retrieval Using Captioning*. In 2021 Fourth International Conference on Electrical, Computer and Communication Technologies (ICECCT), pages 1–5, 2021. URL : https://ieeexplore.ieee.org/document/9616897.

[Krizhevsky 2012] Alex Krizhevsky, Ilya Sutskever et Geoffrey E Hinton. *ImageNet Classification with Deep Convolutional Neural Networks*. In F. Pereira, C.J. Burges, L. Bottou et K.Q. Weinberger, editeurs, Advances in Neural Information Processing Systems, volume 25. Curran Associates, Inc., 2012. URL : https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[Krogh 1991] Anders Krogh et John A. Hertz. *A Simple Weight Decay Can Improve Generalization*. In Proceedings of the 4th International Conference on Neural Information Processing Systems, NIPS'91, page 950–957, San Francisco, CA, USA, 1991. Morgan Kaufmann Publishers Inc. URL : https://papers.nips.cc/paper_files/paper/1991/file/8eefcfdf5990e441f0fb6f3fad709e21-Paper.pdf.

[Kurakin 2020] Alex Kurakin, Colin Raffel, David Berthelot, Ekin Dogus Cubuk, Han Zhang, Kihyuk Sohn et Nicholas Carlini. *ReMixMatch : Semi-Supervised Learning with Distribution Matching and Augmentation Anchoring*. In ICLR, 2020. URL : https://openreview.net/pdf?id=HklkeR4KPB.

[Lacoste 2019] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt et Thomas Dandres. *Quantifying the Carbon Emissions of Machine Learning*. arXiv preprint arXiv :1910.09700, 2019. URL : https://arxiv.org/pdf/1910.09700.pdf.

[Laine 2016] Samuli Laine et Timo Aila. *Temporal Ensembling for Semi-Supervised Learning*. CoRR, vol. abs/1610.02242, 2016. URL : http://arxiv.org/pdf/1610.02242.pdf.

[LeCun 1995] Yann LeCun, Yoshua Bengio*et al.* *Convolutional networks for images, speech, and time series*. The handbook of brain theory and neural networks, vol. 3361, no. 10, page 1995, 1995. URL : https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=e26cc4a1c717653f323715d751c8dea7461aa105.

[Lee 2013] Dong-Hyun Lee*et al.* *Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks*. In Workshop on challenges in representation learning, ICML, volume 3, page 896, 2013. URL : https://www.kaggle.com/blobs/download/forum-message-attachment-files/746/pseudo_label_final.pdf.

[Lewis 2020] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov et Luke 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, pages 7871–7880, Online, Juillet 2020. Association for Computational Linguistics. URL : https://aclanthology.org/2020.acl-main.703.

[Li 2016] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao et Bill Dolan. *A Diversity-Promoting Objective Function for Neural Conversation Models*. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, pages 110–119, San Diego, California, Juin 2016. Association for Computational Linguistics. URL : https://aclanthology.org/N16-1014.

[Li 2020] Junnan Li, Caiming Xiong et Steven C. H. Hoi. *CoMatch : Semi-supervised Learning with Contrastive Graph Regularization*. CoRR, vol. abs/2011.11183, 2020. URL : https://arxiv.org/pdf/2011.11183.pdf.

[Lin 2004] Chin-Yew Lin. *ROUGE : A Package for Automatic Evaluation of Summaries*. In Text Summarization Branches Out, pages 74–81, Barcelona, Spain, Juillet 2004. Association for Computational Linguistics. URL : https://aclanthology.org/W04-1013.

[Lin 2020] Ke Lin, Zhuoxin Gan et Liwei Wang. *Semi-Supervised Learning for Video Captioning*. In Trevor Cohn, Yulan He et Yang Liu, editeurs, Findings of the Association for Computational Linguistics : EMNLP 2020, pages 1096–1106, Online, Novembre 2020. Association for Computational Linguistics. URL : https://aclanthology.org/2020.findings-emnlp.98.

[Liu 2017] Siqi Liu, Zhenhai Zhu, Ning Ye, Sergio Guadarrama et Kevin Murphy. *Improved Image Captioning via Policy Gradient optimization of SPIDEr*. In IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017, pages 873–881. IEEE Computer Society, 2017. URL : https://doi.org/10.1109/ICCV.2017.100.

[Liu 2018] Xihui Liu, Hongsheng Li, Jing Shao, Dapeng Chen et Xiaogang Wang. *Show, Tell and Discriminate : Image Captioning by Self-retrieval with Partially Labeled Data*. In Proceedings of the European Conference on Computer Vision (ECCV), September 2018. URL : https://openaccess.thecvf.com/content_ECCV_2018/papers/Xihui_Liu_Show_Tell_and_ECCV_2018_paper.pdf.

[Liu 2021] Xubo Liu, Qiushi Huang, Xinhao Mei, Tom Ko, H. Tang, Mark D. Plumbley et Wenwu Wang. *CL4AC : A Contrastive Loss for Audio Captioning*. In Proceedings of the 6th Detection and Classification of Acoustic Scenes and Events 2021 Workshop (DCASE2021), pages 196–200, Barcelona, Spain, November 2021. URL : https://dcase.community/documents/workshop2021/proceedings/DCASE2021Workshop_Liu_65.pdf.

[Liu 2022a] Xubo Liu, Xinhao Mei, Qiushi Huang, Jianyuan Sun, Jinzheng Zhao, Haohe Liu, Mark D Plumbley, Volkan Kilic et Wenwu Wang. *Leveraging pre-trained BERT for audio captioning*. In 2022 30th European Signal Processing Conference (EUSIPCO), pages 1145–1149. IEEE, 2022. URL : https://ieeexplore.ieee.org/abstract/document/9909761.

[Liu 2022b] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell et Saining Xie. *A ConvNet for the 2020s*. In 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 11966–11976,

2022. URL : https://openaccess.thecvf.com/content/CVPR2022/papers/Liu_A_ConvNet_for_the_2020s_CVPR_2022_paper.pdf.

[Liu 2023] Xubo Liu, Qiushi Huang, Xinhao Mei, Haohe Liu, Qiuqiang Kong, Jianyuan Sun, Shengchen Li, Tom Ko, Yu Zhang, Lilian H. Tang, Mark D. Plumbley, Volkan Kılıç et Wenwu Wang. *Visually-Aware Audio Captioning With Adaptive Audio-Visual Attention*. In Proc. INTERSPEECH 2023, pages 2838–2842, 2023. URL : https://www.isca-speech.org/archive/pdfs/interspeech_2023/liu23l_interspeech.pdf.

[Loshchilov 2017] Ilya Loshchilov et Frank Hutter. *Decoupled Weight Decay Regularization*. In International Conference on Learning Representations, 2017. URL : https://openreview.net/pdf?id=Bkg6RiCqY7.

[Mahfuz 2023a] Rehana Mahfuz, Yinyi Guo, Arvind Krishna Sridhar et Erik Visser. *Detecting False Alarms and Misses in Audio Captions*, 2023. URL : https://arxiv.org/pdf/2309.03326.pdf.

[Mahfuz 2023b] Rehana Mahfuz, Yinyi Guo et Erik Visser. *Improving Audio Captioning Using Semantic Similarity Metrics*. In ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 1–5, 2023. URL : https://ieeexplore.ieee.org/document/10096522.

[Manco 2023] Ilaria Manco, Benno Weck, SeungHeon Doh, Minz Won, Yixiao Zhang, Dmitry Bodganov, Yusong Wu, Ke Chen, Philip Tovstogan, Emmanouil Benetos, Elio Quinton, György Fazekas et Juhan Nam. *The Song Describer Dataset : a Corpus of Audio Captions for Music-and-Language Evaluation*, 2023. URL : https://arxiv.org/pdf/2311.10057.pdf.

[Martín-Morató 2022] Irene Martín-Morató, Manu Harju et Annamaria Mesaros. *A Summarization Approach to Evaluating Audio Captioning*. In Proceedings of the 7th Detection and Classification of Acoustic Scenes and Events 2022 Workshop (DCASE2022), Nancy, France, November 2022. URL : https://dcase.community/documents/workshop2022/proceedings/DCASE2022Workshop_Martin-Morato_35.pdf.

[Martin 2021] Irene Martin et Annamaria Mesaros. *Diversity and Bias in Audio Captioning Datasets*. In Proceedings of the 6th Detection and Classification of Acoustic Scenes and Events 2021 Workshop (DCASE2021), pages 90–94, Barcelona, Spain, November 2021. URL : https://dcase.community/documents/workshop2021/proceedings/DCASE2021Workshop_Martin_34.pdf.

[McKee 2023] Daniel McKee, Justin Salamon, Josef Sivic et Bryan Russell. *Language-Guided Music Recommendation for Video via Prompt Analogies*. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 14784–14793, 2023. URL : https://openaccess.thecvf.com/content/CVPR2023/papers/McKee_Language-Guided_Music_Recommendation_for_Video_via_Prompt_Analogies_CVPR_2023_paper.pdf.

[Mei 2021a] Xinhao Mei, Qiushi Huang, Xubo Liu, Gengyun Chen, Jingqian Wu, Yusong Wu, Jinzheng ZHAO, Shengchen Li, Tom Ko, H. Tang, Xi Shao, Mark D. Plumbley et Wenwu Wang. *An Encoder-Decoder Based Audio Captioning System with Transfer and Reinforcement Learning*. In Proceedings of the 6th Detection and Classification of Acoustic Scenes and Events 2021 Workshop (DCASE2021), pages 206–210, Barcelona, Spain, November 2021. URL : https://dcase.community/documents/workshop2021/proceedings/DCASE2021Workshop_Mei_67.pdf.

[Mei 2021b] Xinhao Mei, Qiushi Huang, Xubo Liu, Gengyun Chen, Jingqian Wu, Yusong Wu, Jinzheng Zhao, Shengchen Li, Tom Ko, H. Lilian Tang, Xi Shao, Mark D. Plumbley et Wenwu Wang. *An Encoder-Decoder Based Audio Captioning System With Transfer and Reinforcement Learning for DCASE Challenge 2021 Task 6*. Rapport technique, DCASE2021 Challenge, July 2021. URL : https://dcase.community/documents/challenge2021/technical_reports/DCASE2021_Mei_88_t6.pdf.

[Mei 2021c] Xinhao Mei, Xubo Liu, Qiushi Huang, Mark D. Plumbley et Wenwu Wang. *Audio Captioning Transformer*. In Proceedings of the 6th Detection and Classification of Acoustic Scenes and Events 2021 Workshop (DCASE2021), pages 211–215, Barcelona, Spain, November 2021. URL : https://dcase.community/documents/workshop2021/proceedings/DCASE2021Workshop_Mei_68.pdf.

[Mei 2022a] Xinhao Mei, Xubo Liu, Haohe Liu, Jianyuan Sun, Mark D. Plumbley et Wenwu Wang. *Automated audio captioning with keywords guidance*. Rapport technique, DCASE2022 Challenge, July 2022. URL : https://dcase.community/documents/challenge2022/technical_reports/DCASE2022_Mei_117_t6a.pdf.

[Mei 2022b] Xinhao Mei, Xubo Liu, Jianyuan Sun, Mark Plumbley et Wenwu Wang. *On Metric Learning for Audio-Text Cross-Modal Retrieval*. In Proc. Interspeech 2022, pages 4142–4146, 2022. URL : https://www.isca-speech.org/archive/pdfs/interspeech_2022/mei22_interspeech.pdf.

[Mei 2022c] Xinhao Mei, Xubo Liu, Jianyuan Sun, Mark D. Plumbley et Wenwu Wang. *Diverse Audio Captioning Via Adversarial Training*. In ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 8882–8886, 2022. URL : https://ieeexplore.ieee.org/document/9746894.

[Mei 2023] Xinhao Mei, Chutong Meng, Haohe Liu, Qiuqiang Kong, Tom Ko, Chengqi Zhao, Mark D. Plumbley, Yuexian Zou et Wenwu Wang. *WavCaps : A ChatGPT-Assisted Weakly-Labelled Audio Captioning Dataset for Audio-Language Multimodal Research*. arXiv preprint arXiv :2303.17395v1, 2023. URL : https://arxiv.org/abs/2303.17395v1.

[Meister 2022] Clara Meister, Tiago Pimentel, Gian Wiher et Ryan Cotterell. *Typical Decoding for Natural Language Generation*. arXiv :2202.00666 [cs], Mars 2022. arXiv : 2202.00666. URL : http://arxiv.org/pdf/2202.00666.pdf.

[Mesaros 2018] Annamaria Mesaros, Toni Heittola et Tuomas Virtanen. *A multi-device dataset for urban acoustic scene classification*. In Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018), pages 9–13, November 2018. URL : https://arxiv.org/pdf/1807.09840.pdf.

[Mikolov 2013a] Tomas Mikolov, Kai Chen, Greg Corrado et Jeffrey Dean. *Efficient estimation of word representations in vector space*. arXiv preprint arXiv :1301.3781, 2013. URL : https://arxiv.org/pdf/1301.3781.pdf.

[Mikolov 2013b] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado et Jeff Dean. *Distributed Representations of Words and Phrases and their Compositionality*. In C. J. Burges, L. Bottou, M. Welling, Z. Ghahramani et K. Q. Weinberger, editeurs, Advances in Neural Information Processing Systems, volume 26. Curran Associates, Inc., 2013. URL : https://proceedings.neurips.cc/paper_files/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.

[Miller 1994] George A. Miller. *WordNet : A Lexical Database for English*. In Human Language Technology : Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994, 1994. URL : https://aclanthology.org/H94-1111.

[Miyato 2018] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama et Shin Ishii. *Virtual adversarial training : a regularization method for supervised and semi-supervised learning*. IEEE transactions on pattern analysis and machine intelligence, vol. 41, no. 8, pages 1979–1993, 2018. URL : https://ieeexplore.ieee.org/abstract/document/8417973.

[Montani 2021] Ines Montani, Matthew Honnibal, Matthew Honnibal, Sofie Van Landeghem, Adriane Boyd, Henning Peters, Paul O'Leary McCann, Maxim Samsonov, Jim Geovedi, Jim O'Regan, György Orosz, Duygu Altinok, Søren Lind Kristiansen, Roman, Explosion Bot, Leander Fiedler, Grégory Howard, Wannaphong Phatthiyaphaibun, Yohei Tamura, Sam Bozek, murat, Mark Amery, Björn Böing, Pradeep Kumar Tippa, Leif Uwe Vogelsang, Bram Vanroy, Ramanan Balakrishnan, Vadim Mazaev et GregDubbin. *explosion/spaCy : v3.2.1 : doc_cleaner component, new Matcher attributes, bug fixes and more*, Décembre 2021. URL : https://doi.org/10.5281/zenodo.5764736.

[Nair 2010] Vinod Nair et Geoffrey E. Hinton. *Rectified Linear Units Improve Restricted Boltzmann Machines*. In Proceedings of the 27th International Conference on International Conference on Machine Learning, ICML'10, page 807–814, Madison, WI, USA, 2010. Omnipress. URL : https://www.cs.toronto.edu/~fritz/absps/reluICML.pdf.

[Narisetty 2022] Chaitanya Narisetty, Emiru Tsunoo, Xuankai Chang, Yosuke Kashiwagi, Michael Hentschel et Shinji Watanabe. *Joint Speech Recognition and Audio Captioning*. In ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 7892–7896, 2022. URL : https://ieeexplore.ieee.org/document/9746601.

[Nogueira 2014] Fernando Nogueira. *Bayesian Optimization : Open source constrained global optimization tool for Python*, 2014. URL : https://github.com/fmfn/BayesianOptimization.

[Novikova 2017] Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry et Verena Rieser. *Why We Need New Evaluation Metrics for NLG*. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2241–2252, Copenhagen, Denmark, Septembre 2017. Association for Computational Linguistics. URL : https://aclanthology.org/D17-1238.

[Oncescu 2021] Andreea-Maria Oncescu, A. Sophia Koepke, João F. Henriques, Zeynep Akata et Samuel Albanie. *Audio Retrieval with Natural Language Queries*. In Proc. Interspeech 2021, pages 2411–2415, 2021. URL : https://www.isca-speech.org/archive/pdfs/interspeech_2021/oncescu21_interspeech.pdf.

[Papineni 2001] Kishore Papineni, Salim Roukos, Todd Ward et Wei-Jing Zhu. *BLEU : a method for automatic evaluation of machine translation*. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02, page 311, Philadelphia, Pennsylvania, 2001. Association for Computational Linguistics. URL : http://portal.acm.org/citation.cfm?doid=1073083.1073135.

[Park 2019] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk et Quoc V. Le. *SpecAugment : A Simple Data Augmentation Method for Automatic Speech Recognition*. In Interspeech 2019, pages 2613–2617. ISCA, sep 2019. URL : https://doi.org/10.21437%2Finterspeech.2019-2680.

[Pascanu 2013] Razvan Pascanu, Tomas Mikolov et Yoshua Bengio. *On the Difficulty of Training Recurrent Neural Networks*. In Proc. of the 30th International Conf. on International Conf. on Machine Learning - Volume 28, ICML'13, page III–1310–III–1318. JMLR.org, 2013. URL : https://proceedings.mlr.press/v28/pascanu13.pdf.

[Pellegrini 2020] Thomas Pellegrini. *IRIT-UPS DCASE 2020 audio captioning system.* Rapport technique, DCASE2020 Challenge, June 2020. URL : https://dcase.community/documents/challenge2020/technical_reports/DCASE2020_Pellegrini_131_t6.pdf.

[Pellegrini 2022] Thomas Pellegrini. *IRIT-UPS DCASE 2022 Language-Based Audio Retrieval System.* Rapport technique, DCASE2022 Challenge, July 2022. URL : https://dcase.community/documents/challenge2022/technical_reports/DCASE2022_Pellegrini_3_t6b.pdf.

[Peng 2023] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, Kranthi Kiran GV, Xuzheng He, Haowen Hou, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartlomiej Koptyra, Hayden Lau, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Xiangru Tang, Bolun Wang, Johan S. Wind, Stansilaw Wozniak, Ruichong Zhang, Zhenyuan Zhang, Qihang Zhao, Peng Zhou, Jian Zhu et Rui-Jie Zhu. *RWKV : Reinventing RNNs for the Transformer Era*, 2023. URL : https://arxiv.org/pdf/2305.13048.pdf.

[Pennington 2014] Jeffrey Pennington, Richard Socher et Christopher Manning. *GloVe : Global Vectors for Word Representation.* In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar, Octobre 2014. Association for Computational Linguistics. URL : https://aclanthology.org/D14-1162.

[Perez-Castanos 2020] Sergi Perez-Castanos, Javier Naranjo-Alcazar, Pedro Zuccarello et Maximo Cobos. *Listen Carefully and Tell : An Audio Captioning System Based on Residual Learning and Gammatone Audio Representation.* In Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020), pages 150–154, Tokyo, Japan, November 2020. URL : https://dcase.community/documents/workshop2020/proceedings/DCASE2020Workshop_Perez-Castanos_19.pdf.

[Piczak 2015] Karol J. Piczak. *ESC : Dataset for Environmental Sound Classification.* In proc. ACM Multimedia, page 1015–1018, Brisbane, 2015. URL : https://doi.org/10.1145/2733373.2806390.

[Plakal 2020] M. Plakal et D. Ellis. *YAMNet*, Janvier 2020. URL : https://github.com/tensorflow/models/tree/master/research/audioset/yamnet.

[Porter 2001] Martin F. Porter. *Snowball : A language for stemming algorithms.* Published online, October 2001. Accessed 11.03.2008, 15.00h. URL : http://snowball.tartarus.org/texts/introduction.html.

[Primus 2022] Paul Primus et Gerhard Widmer. *CP-JKU's submission to task 6a of the DCASE2022 challenge : a BART encoder-decoder for automatic audio captioning trained via the reinforce algorithm and transfer learning.* Rapport technique, DCASE2022 Challenge, July 2022. URL : https://dcase.community/documents/challenge2022/technical_reports/DCASE2022_Primus_97_t6a.pdf.

[Primus 2023] Paul Primus, Khaled Koutini et Gerhard Widmer. *CP-JKU'S SUBMISSION TO TASK 6b OF THE DCASE2023 CHALLENGE : AUDIO RETRIEVAL WITH PaSST AND GPT-AUGMENTED CAPTIONS.* Rapport technique, DCASE2023 Challenge, June 2023. URL : https://dcase.community/documents/challenge2023/technical_reports/DCASE2023_Primus_72_t6b.pdf.

[Qian 1999] Ning Qian. *On the momentum term in gradient descent learning algorithms*. Neural Networks, vol. 12, no. 1, pages 145–151, 1999. URL : https://www.sciencedirect.com/science/article/pii/S0893608098001166.

[Qiao 2018] Siyuan Qiao, Wei Shen, Zhishuai Zhang, Bo Wang et Alan L. Yuille. *Deep Co-Training for Semi-Supervised Image Recognition*. CoRR, vol. abs/1803.05984, 2018. URL : http://arxiv.org/pdf/1803.05984.pdf.

[Radford 2019] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei et Ilya Sutskever. *Language Models are Unsupervised Multitask Learners*, 2019. URL : https://api.semanticscholar.org/CorpusID:160025533.

[Radford 2021] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark*et al.* *Learning transferable visual models from natural language supervision*. In International conference on machine learning, pages 8748–8763. PMLR, 2021. URL : http://proceedings.mlr.press/v139/radford21a/radford21a.pdf.

[Radford 2023] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine Mcleavey et Ilya Sutskever. *Robust Speech Recognition via Large-Scale Weak Supervision*. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato et Jonathan Scarlett, editeurs, Proceedings of the 40th International Conference on Machine Learning, volume 202 of *Proceedings of Machine Learning Research*, pages 28492–28518. PMLR, 23–29 Jul 2023. URL : https://proceedings.mlr.press/v202/radford23a.html.

[Rasmus 2015] Antti Rasmus, Harri Valpola, Mikko Honkala, Mathias Berglund et Tapani Raiko. *Semi-Supervised Learning with Ladder Networks*, 2015. URL : https://arxiv.org/pdf/1507.02672.pdf.

[Reimers 2019] Nils Reimers et Iryna Gurevych. *Sentence-BERT : Sentence Embeddings using Siamese BERT-Networks*, 2019. URL : https://arxiv.org/pdf/1908.10084.pdf.

[Rennie 2017] Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross et Vaibhava Goel. *Self-Critical Sequence Training for Image Captioning*. In 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, pages 1179–1195. IEEE Computer Society, 2017. URL : https://doi.org/10.1109/CVPR.2017.131.

[Salamon 2014] Justin Salamon, Christopher Jacoby et Juan Pablo Bello. *A Dataset and Taxonomy for Urban Sound Research*. In proc. ACM Multimedia, page 1041–1044, 2014. URL : https://doi.org/10.1145/2647868.2655045.

[Salewski 2023] Leonard Salewski, Stefan Fauth, A. Sophia Koepke et Zeynep Akata. *Zero-shot audio captioning with audio-language model guidance and audio context keywords*, 2023. URL : https://arxiv.org/pdf/2311.08396.pdf.

[Schaumlöffel 2023] Timothy Schaumlöffel, Martina G. Vilas et Gemma Roig. *PEACS : Prefix encoding for auditory caption synthesis*. Rapport technique, DCASE2023 Challenge, May 2023. URL : https://dcase.community/documents/challenge2023/technical_reports/DCASE2023_Schaumloeffel_107_t6a.pdf.

[Sellam 2020] Thibault Sellam, Dipanjan Das et Ankur Parikh. *BLEURT : Learning Robust Metrics for Text Generation*. In Dan Jurafsky, Joyce Chai, Natalie Schluter et Joel Tetreault, editeurs, Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 7881–7892, Online, Juillet 2020. Association for Computational Linguistics. URL : https://aclanthology.org/2020.acl-main.704.

[Sennrich 2015] Rico Sennrich, Barry Haddow et Alexandra Birch. *Neural Machine Translation of Rare Words with Subword Units*. CoRR, vol. abs/1508.07909, 2015. URL : http://arxiv.org/pdf/1508.07909.pdf.

[Shaw 2018] Peter Shaw, Jakob Uszkoreit et Ashish Vaswani. *Self-Attention with Relative Position Representations*. In Marilyn Walker, Heng Ji et Amanda Stent, editeurs, Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 2 (Short Papers), pages 464–468, New Orleans, Louisiana, Juin 2018. Association for Computational Linguistics. URL : https://aclanthology.org/N18-2074.

[Shazeer 2019] Noam Shazeer. *Fast Transformer Decoding : One Write-Head is All You Need*, 2019. URL : https://arxiv.org/pdf/1911.02150.pdf.

[Shin 2022] Inkyu Shin, Yi-Hsuan Tsai, Bingbing Zhuang, Samuel Schulter, Buyu Liu, Sparsh Garg, In So Kweon et Kuk-Jin Yoon. *MM-TTA : Multi-Modal Test-Time Adaptation for 3D Semantic Segmentation*. In CVPR, 2022. URL : https://openaccess.thecvf.com/content/CVPR2022/papers/Shin_MM-TTA_Multi-Modal_Test-Time_Adaptation_for_3D_Semantic_Segmentation_CVPR_2022_paper.pdf.

[Shin 2023] Wooseok Shin, Hyun Joon Park, Jin Sob Kim, Dongwon Kim, Seungjin Lee et Sung Won Han. *Rethinking Transfer and Auxiliary Learning for Improving Audio Captioning Transformer*. In Proc. INTERSPEECH 2023, pages 2128–2132, 2023. URL : https://www.isca-speech.org/archive/pdfs/interspeech_2023/shin23_interspeech.pdf.

[Simonyan 2014] Karen Simonyan et Andrew Zisserman. *Very deep convolutional networks for large-scale image recognition*. arXiv preprint arXiv :1409.1556, 2014. URL : https://arxiv.org/pdf/1409.1556.pdf.

[Sohn 2020] Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin et Chun-Liang Li. *Fixmatch : Simplifying semi-supervised learning with consistency and confidence*. Advances in neural information processing systems, vol. 33, pages 596–608, 2020. URL : https://proceedings.neurips.cc/paper_files/paper/2020/file/06964dce9addb1c5cb5d6e3d9838f733-Paper.pdf.

[Song 2020] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu et Tie-Yan Liu. *MPNet : Masked and Permuted Pre-Training for Language Understanding*. In Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. URL : https://proceedings.neurips.cc/paper/2020/file/c3a690be93aa602ee2dc0ccab5b7b67e-Paper.pdf.

[Srivastava 2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever et Ruslan Salakhutdinov. *Dropout : A Simple Way to Prevent Neural Networks from Overfitting*. Journal of Machine Learning Research, vol. 15, no. 56, pages 1929–1958, 2014. URL : http://jmlr.org/papers/v15/srivastava14a.html.

[Su 2023] Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen tau Yih, Noah A. Smith, Luke Zettlemoyer et Tao Yu. *One Embedder, Any Task : Instruction-Finetuned Text Embeddings*, 2023. URL : https://arxiv.org/pdf/2212.09741.pdf.

[Sun 2023a] Haoran Sun, Zhiyong Yan, Yongqing Wang, Heinrich Dinkel, Junbo Zhang et Yujun Wang. *Leveraging multi-task training and image retrieval with CLAP for audio captioning*. Rapport technique, DCASE2023 Challenge, May 2023. URL : https://dcase.community/documents/challenge2023/technical_reports/DCASE2023_Yan_127_t6a.pdf.

[Sun 2023b] Luoyi Sun, Xuenan Xu, Mengyue Wu et Weidi Xie. *A Large-scale Dataset for Audio-Language Representation Learning*, 2023. URL : https://arxiv.org/pdf/2309.11500.pdf.

[Sun 2023c] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang et Furu Wei. *Retentive Network : A Successor to Transformer for Large Language Models*, 2023. URL : https://arxiv.org/pdf/2307.08621.pdf.

[Szegedy 2016] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens et Zbigniew Wojna. *Rethinking the Inception Architecture for Computer Vision*. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2818–2826, 2016. URL : https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Szegedy_Rethinking_the_Inception_CVPR_2016_paper.pdf.

[Takeuchi 2020] Daiki Takeuchi, Yuma Koizumi, Yasunori Ohishi, Noboru Harada et Kunio Kashino. *Effects of Word-Frequency Based Pre- and Post- Processings for Audio Captioning*. In Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020), pages 190–194, Tokyo, Japan, November 2020. URL : https://dcase.community/documents/workshop2020/proceedings/DCASE2020Workshop_Takeuchi_79.pdf.

[Takeuchi 2023] Daiki Takeuchi, Yasunori Ohishi, Daisuke Niizumi, Noboru Harada et Kunio Kashino. *Audio Difference Captioning Utilizing Similarity-Discrepancy Disentanglement*. In Proceedings of the 8th Detection and Classification of Acoustic Scenes and Events 2023 Workshop (DCASE2023), pages 191–195, Tampere, Finland, September 2023. URL : https://dcase.community/documents/workshop2023/proceedings/DCASE2023Workshop_Takeuchi_30.pdf.

[Tarvainen 2017] Antti Tarvainen et Harri Valpola. *Mean teachers are better role models : Weight-averaged consistency targets improve semi-supervised deep learning results*. In proc. NeurIPS, pages 1195–1204, Long Beach, 2017. URL : https://proceedings.neurips.cc/paper_files/paper/2017/file/68053af2923e00204c3ca7c6a3150cf7-Paper.pdf.

[Tiedemann 2020] Jörg Tiedemann et Santhosh Thottingal. *OPUS-MT – Building open translation services for the World*. In Proceedings of the 22nd Annual Conference of the European Association for Machine Translation, pages 479–480, Lisboa, Portugal, Novembre 2020. URL : https://aclanthology.org/2020.eamt-1.61.

[Tran 2021] An Tran, Konstantinos Drossos et Tuomas Virtanen. *WaveTransformer : An Architecture for Audio Captioning Based on Learning Temporal and Time-Frequency Information*. In 2021 29th European Signal Processing Conference (EUSIPCO), European Signal Processing Conference, pages 576–580. IEEE, 2021. jufoid=55867 ; European Signal Processing Conference, EUSIPCO 2021 ; Conference date : 23-08-2021 Through 27-08-2021. URL : https://eurasip.org/Proceedings/Eusipco/Eusipco2021/pdfs/0000576.pdf.

[Tsubaki 2023] Shunsuke Tsubaki, Yohei Kawaguchi, Tomoya Nishida, Keisuke Imoto, Yuki Okamoto, Kota Dohi et Takashi Endo. *Audio-Change Captioning to Explain Machine-Sound Anomalies*. In Proceedings of the 8th Detection and Classification of Acoustic Scenes and Events 2023 Workshop (DCASE2023), pages 201–205, Tampere, Finland, September 2023. URL : https://dcase.community/documents/workshop2023/proceedings/DCASE2023Workshop_Tsubaki_8.pdf.

[van den Oord 2018] Aaron van den Oord, Yazhe Li et Oriol Vinyals. *Representation Learning with Contrastive Predictive Coding*, 2018. URL : https://arxiv.org/abs/1807.03748.

[Vaswani 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser et Illia Polosukhin. *Attention is All you Need*. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan et R. Garnett, editeurs, Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017. URL : https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[Vedantam 2015] Ramakrishna Vedantam, C. Lawrence Zitnick et Devi Parikh. *CIDEr : Consensus-based image description evaluation*. In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 4566–4575, 2015. URL : https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Vedantam_CIDEr_Consensus-Based_Image_2015_CVPR_paper.pdf.

[Verma 2022] Vikas Verma, Kenji Kawaguchi, Alex Lamb, Juho Kannala, Arno Solin, Yoshua Bengio et David Lopez-Paz. *Interpolation consistency training for semi-supervised learning*. Neural Networks, vol. 145, pages 90–106, jan 2022. URL : https://doi.org/10.1016%2Fj.neunet.2021.10.008.

[Vinyals 2015] O. Vinyals, A. Toshev, S. Bengio et D. Erhan. *Show and tell : A neural image caption generator*. In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3156–3164, Los Alamitos, CA, USA, jun 2015. IEEE Computer Society. URL : https://doi.ieeecomputersociety.org/10.1109/CVPR.2015.7298935.

[Vygon 2021] Roman Vygon et Nikolay Mikhaylovskiy. *Learning Efficient Representations for Keyword Spotting with Triplet Loss*. In Speech and Computer, pages 773–785. Springer International Publishing, 2021. URL : https://doi.org/10.1007/978-3-030-87802-3_69.

[Wang 2020] Helin Wang, Bang Yang, Yuexian Zou et Dading Chong. *Automated Audio Captioning With Temporal Attention*. Rapport technique, DCASE2020 Challenge, June 2020. URL : https://dcase.community/documents/challenge2020/technical_reports/DCASE2020_Wang_5_t6.pdf.

[Wang 2021] Helin Wang, Yuexian Zou et Wenwu Wang. *SpecAugment++ : A Hidden Space Data Augmentation Method for Acoustic Scene Classification*. In Interspeech, 2021. URL : https://api.semanticscholar.org/CorpusID:232428343.

[Wang 2022a] Yidong Wang, Hao Chen, Yue Fan, Wang Sun, Ran Tao, Wenxin Hou, Renjie Wang, Linyi Yang, Zhi Zhou, Lan-Zhe Guo, Heli Qi, Zhen Wu, Yu-Feng Li, Satoshi Nakamura, Wei Ye, Marios Savvides, Bhiksha Raj, Takahiro Shinozaki, Bernt Schiele, Jindong Wang, Xing Xie et Yue Zhang. *USB : A Unified Semi-supervised Learning Benchmark for Classification*, 2022. URL : https://arxiv.org/pdf/2208.07204.pdf.

[Wang 2022b] Yidong Wang, Hao Chen, Qiang Heng, Wenxin Hou, Yue Fan, Zhen Wu, Jindong Wang, Marios Savvides, Takahiro Shinozaki, Bhiksha Raj, Bernt Schiele et Xing Xie. *FreeMatch : Self-adaptive Thresholding for Semi-supervised Learning*, 2022. URL : https://arxiv.org/pdf/2205.07246.pdf.

[Wang 2023a] Chung-Che Wang, Jiawei Du et Jyh-Shing Roger Jang. *DCASE 2023 TASK 6B : TEXT-TO-AUDIO RETRIEVAL USING PRETRAINED MODELS*. Rapport technique, DCASE2023 Challenge, June 2023. URL : https://dcase.community/documents/challenge2023/technical_reports/DCASE2023_Wang_40_t6b.pdf.

[Wang 2023b] Peng Wang, Shijie Wang, Junyang Lin, Shuai Bai, Xiaohuan Zhou, Jingren Zhou, Xinggang Wang et Chang Zhou. *ONE-PEACE : Exploring One General Representation*

Model Toward Unlimited Modalities. arXiv preprint arXiv :2305.11172, 2023. URL : https://arxiv.org/pdf/2305.11172.pdf.

[Warden 2018] Pete Warden. *Speech Commands : A Dataset for Limited-Vocabulary Speech Recognition*. CoRR, vol. abs/1804.03209, 2018. URL : http://arxiv.org/pdf/1804.03209.pdf.

[Weck 2021] Benno Weck, Xavier Favory, Konstantinos Drossos et Xavier Serra. *Evaluating Off-the-Shelf Machine Listening and Natural Language Models for Automated Audio Captioning*. In Proceedings of the 6th Detection and Classification of Acoustic Scenes and Events 2021 Workshop (DCASE2021), pages 60–64, Barcelona, Spain, November 2021. URL : https://dcase.community/documents/workshop2021/proceedings/DCASE2021Workshop_Weck_25.pdf.

[Wijngaard 2023] Gijs Wijngaard, Elia Formisano, Bruno L Giordano et Michel Dumontier. *ACES : Evaluating Automated Audio Captioning Models on the Semantics of Sounds*. In 2023 31th European Signal Processing Conference (EUSIPCO). EUSIPCO, 2023. URL : https://eurasip.org/Proceedings/Eusipco/Eusipco2023/pdfs/0000770.pdf.

[Williams 2018] Adina Williams, Nikita Nangia et Samuel Bowman. *A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference*. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long Papers), pages 1112–1122, New Orleans, Louisiana, Juin 2018. Association for Computational Linguistics. URL : https://aclanthology.org/N18-1101.

[Won 2021] Hyejin Won, Baekseung Kim, Il-Youp Kwak et Changwon Lim. *CAU Submission to DCASE 2021 Task6 : Transformer Followed by Transfer Learning for Audio Captioning*. Rapport technique, DCASE2021 Challenge, July 2021. URL : https://dcase.community/documents/challenge2021/technical_reports/DCASE2021_Won_103_t6.pdf.

[Woo 2023] Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon et Saining Xie. *ConvNeXt V2 : Co-designing and Scaling ConvNets with Masked Autoencoders*. In 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 16133–16142, 2023. URL : https://openaccess.thecvf.com/content/CVPR2023/papers/Woo_ConvNeXt_V2_Co-Designing_and_Scaling_ConvNets_With_Masked_Autoencoders_CVPR_2023_paper.pdf.

[Wu 2016] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes et Jeffrey Dean. *Google's Neural Machine Translation System : Bridging the Gap between Human and Machine Translation*, 2016. URL : https://arxiv.org/pdf/1609.08144.pdf.

[Wu 2019] Mengyue Wu, Heinrich Dinkel et Kai Yu. *Audio Caption : Listen and Tell*. ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 830–834, 2019. URL : https://ieeexplore.ieee.org/document/8682377.

[Wu 2020] Yusong Wu, Kun Chen, Ziyue Wang, Xuan Zhang, Fudong Nian, Shengchen Li et Xi Shao. *Audio Captioning Based on Transformer and Pre-Training for 2020 DCASE Audio Captioning Challenge*. Rapport technique, DCASE2020 Challenge, June 2020. URL : https://dcase.community/documents/challenge2020/technical_reports/DCASE2020_Wu_136_t6.pdf.

[Wu 2023a] Ho-Hsiang Wu, Oriol Nieto, Juan Pablo Bello et Justin Salamon. *Audio-Text Models Do Not Yet Leverage Natural Language*, 2023. URL : https://arxiv.org/pdf/2303.10667.pdf.

[Wu 2023b] Shih-Lun Wu, Xuankai Chang, Gordon Wichern, Jee-weon Jung, François Germain, Jonathan Le Roux et Shinji Watanabe. *BEATs-based audio captioning model with INSTRUCTOR embedding supervision and ChatGPT mix-up*. Rapport technique, DCASE2023 Challenge, May 2023. URL : https://dcase.community/documents/challenge2023/technical_reports/DCASE2023_Wu_31_t6a.pdf.

[Wu 2023c] Yusong Wu, Ke Chen, Tianyu Zhang, Yuchen Hui, Taylor Berg-Kirkpatrick et Shlomo Dubnov. *Large-Scale Contrastive Language-Audio Pretraining with Feature Fusion and Keyword-to-Caption Augmentation*. In ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 1–5, 2023. URL : https://ieeexplore.ieee.org/abstract/document/10095969.

[Wu* 2023d] Yusong Wu*, Ke Chen*, Tianyu Zhang*, Yuchen Hui*, Taylor Berg-Kirkpatrick et Shlomo Dubnov. *Large-scale Contrastive Language-Audio Pretraining with Feature Fusion and Keyword-to-Caption Augmentation*. In IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, 2023. URL : https://ieeexplore.ieee.org/document/10095969.

[Xiao 2023] Feiyang Xiao, Qiaoxi Zhu, Haiyan Lan, Wenwu Wang et Jian Guan. *Ensemble systems with contrastive language-audio pretraining and attention-based audio features for audio captioning and retrieval*. Rapport technique, DCASE2023 Challenge, May 2023. URL : https://dcase.community/documents/challenge2023/technical_reports/DCASE2023_Guan_83_t6a.pdf.

[Xie 2019] Qizhe Xie, Minh-Thang Luong, Eduard Hovy et Quoc V. Le. *Self-training with Noisy Student improves ImageNet classification*, 2019. URL : https://arxiv.org/pdf/1911.04252.pdf.

[Xie 2020] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong et Quoc V. Le. *Unsupervised Data Augmentation for Consistency Training*. In Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. URL : https://proceedings.neurips.cc/paper_files/paper/2020/file/44feb0096faa8326192570788b38c1d1-Paper.pdf.

[Xie 2023] Zeyu Xie, Xuenan Xu, Mengyue Wu et Kai Yu. *Enhance Temporal Relations in Audio Captioning with Sound Event Detection*. In Proc. INTERSPEECH 2023, pages 4179–4183, 2023. URL : https://www.isca-speech.org/archive/pdfs/interspeech_2023/xie23d_interspeech.pdf.

[Xin 2023] Yifei Xin, Dongchao Yang et Yuexian Zou. *Improving Text-Audio Retrieval by Text-Aware Attention Pooling and Prior Matrix Revised Loss*. In ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 1–5, 2023. URL : https://ieeexplore.ieee.org/document/10096972.

[Xu 2020a] Xuenan Xu, Heinrich Dinkel, Mengyue Wu et Kai Yu. *A CRNN-GRU Based Reinforcement Learning Approach to Audio Captioning*. In Proceedings of the Detection

and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020), pages 225–229, Tokyo, Japan, November 2020. URL : https://dcase.community/documents/workshop2020/proceedings/DCASE2020Workshop_Xu_83.pdf.

[Xu 2020b] Xuenan Xu, Heinrich Dinkel, Mengyue Wu et Kai Yu. *The SJTU Submission for DCASE2020 Task 6 : A CRNN-GRU Based Reinforcement Learning Approach to Audiocaption.* Rapport technique, DCASE2020 Challenge, June 2020. URL : https://dcase.community/documents/challenge2020/technical_reports/DCASE2020_Xu_43_t6.pdf.

[Xu 2021a] Xuenan Xu, Heinrich Dinkel, Mengyue Wu, Zeyu Xie et Kai Yu. *Investigating local and global information for automated audio captioning with transfer learning.* In ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 905–909. IEEE, 2021. URL : https://ieeexplore.ieee.org/abstract/document/9413982.

[Xu 2021b] Xuenan Xu, Heinrich Dinkel, Mengyue Wu et Kai Yu. *Audio Caption in a Car Setting with a Sentence-Level Loss.* In 2021 12th International Symposium on Chinese Spoken Language Processing (ISCSLP), pages 1–5, 2021. URL : https://ieeexplore.ieee.org/abstract/document/9362117.

[Xu 2021c] Xuenan Xu, Heinrich Dinkel, Mengyue Wu et Kai Yu. *Text-to-Audio Grounding : Building Correspondence Between Captions and Sound Events.* In ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 606–610, 2021. URL : https://ieeexplore.ieee.org/document/9414834.

[Xu 2021d] Xuenan Xu, Heinrich Dinkel, Mengyue Wu et Kai Yu. *Text-to-audio grounding : Building correspondence between captions and sound events.* In ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 606–610. IEEE, 2021. URL : https://ieeexplore.ieee.org/document/9414834.

[Xu 2021e] Yi Xu, Lei Shang, Jinxing Ye, Qi Qian, Yu-Feng Li, Baigui Sun, Hao Li et Rong Jin. *Dash : Semi-Supervised Learning with Dynamic Thresholding*, 2021. URL : https://arxiv.org/pdf/2109.00650.pdf.

[Xu 2022] Xuenan Xu, Zeyu Xie, Mengyue Wu et Kai Yu. *The SJTU System for DCASE2022 Challenge Task 6 : Audio Captioning with Audio-Text Retrieval Pre-training.* Rapport technique, DCASE2022 Challenge, July 2022. URL : https://dcase.community/documents/challenge2022/technical_reports/DCASE2022_Xu_106_t6a.pdf.

[Xue 2022] Hongwei Xue, Tiankai Hang, Yanhong Zeng, Yuchong Sun, Bei Liu, Huan Yang, Jianlong Fu et Baining Guo. *Advancing High-Resolution Video-Language Representation with Large-Scale Video Transcriptions.* In International Conference on Computer Vision and Pattern Recognition (CVPR), 2022. URL : https://openaccess.thecvf.com/content/CVPR2022/papers/Xue_Advancing_High-Resolution_Video-Language_Representation_With_Large-Scale_Video_Transcriptions_CVPR_2022_paper.pdf.

[Yang 2021] Liu Yang et Bi Sijun. *The DCASE2021 Challenge Task 6 System : Automated Audio Caption.* Rapport technique, DCASE2021 Challenge, July 2021. URL : https://dcase.community/documents/challenge2021/technical_reports/DCASE2021_Yang_76_t6.pdf.

[Yang 2023] Dongchao Yang, Jianwei Yu, Helin Wang, Wen Wang, Chao Weng, Yuexian Zou et Dong Yu. *Diffsound : Discrete Diffusion Model for Text-to-Sound Generation.* IEEE/ACM

Transactions on Audio, Speech, and Language Processing, vol. 31, pages 1720–1733, 2023. URL : https://ieeexplore.ieee.org/document/10112585.

[Ye 2021a] Zhongjie Ye, Helin Wang, Dongchao Yang et Yuexian 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), pages 40–44, Barcelona, Spain, November 2021. URL : https://dcase.community/documents/workshop2021/proceedings/DCASE2021Workshop_Ye_19.pdf.

[Ye 2021b] Zhongjie Ye, Helin Wang, Dongchao Yang et Yuexian Zou. *Improving the Performance of Automated Audio Captioning via Integrating the Acoustic and Textual Information*. Rapport technique, DCASE2021 Challenge, July 2021. URL : https://dcase.community/documents/challenge2021/technical_reports/DCASE2021_Ye_21_t6.pdf.

[Ye 2022] Zhongjie Ye, Yuexian Zou, Fan Cui et Yujun Wang. *Automated audio captioning with multi-task learning*. Rapport technique, DCASE2022 Challenge, July 2022. URL : https://dcase.community/documents/challenge2022/technical_reports/DCASE2022_Zou_37_t6a.pdf.

[Yuan 2021] Weiqiang Yuan, Qichen Han, Dong Liu, Xiang Li et Zhen Yang. *The DCASE 2021 Challenge Task 6 System : Automated Audio Captioning With Weakly Supervised Pre-training and Word Selection Methods*. Rapport technique, DCASE2021 Challenge, July 2021. URL : https://dcase.community/documents/challenge2021/technical_reports/DCASE2021_Yuan_2_t6.pdf.

[Yulianto 2021] Ahmad Yulianto et Rina Supriatnaningsih. *Google Translate vs. DeepL : A Quantitative Evaluation of Close-Language Pair Translation (French to English)*. AJELP : Asian Journal of English Language and Pedagogy, vol. 9, no. 2, pages 109–127, Décembre 2021. URL : https://ojs.upsi.edu.my/index.php/AJELP/article/view/6087.

[Zagoruyko 2017] Sergey Zagoruyko et Nikos Komodakis. *Wide Residual Networks*, 2017. URL : https://arxiv.org/pdf/1605.07146.pdf.

[Zhai 2021] Shuangfei Zhai, Walter Talbott, Nitish Srivastava, Chen Huang, Hanlin Goh et Joshua M. Susskind. *An Attention Free Transformer*, 2021. URL : https://openreview.net/forum?id=pW--cu2FCHY.

[Zhang 2018] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin et David Lopez-Paz. *mixup : Beyond Empirical Risk Minimization*. In 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net, 2018. URL : https://openreview.net/forum?id=r1Ddp1-Rb.

[Zhang* 2020] Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger et Yoav Artzi. *BERTScore : Evaluating Text Generation with BERT*. In International Conference on Learning Representations, 2020. URL : https://openreview.net/forum?id=SkeHuCVFDr.

[Zhang 2021] Bowen Zhang, Yidong Wang, Wenxin Hou, Hao Wu, Jindong Wang, Manabu Okumura et Takahiro Shinozaki. *FlexMatch : Boosting Semi-Supervised Learning with Curriculum Pseudo Labeling*, 2021. URL : https://arxiv.org/pdf/2110.08263.pdf.

[Zhang 2023] Yiming Zhang, Hong Yu, Ruoyi Du, Zheng-Hua Tan, Wenwu Wang, Zhanyu Ma et Yuan Dong. *ACTUAL : Audio Captioning With Caption Feature Space Regularization*. IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 31, pages 2643–2657, 2023. URL : https://ieeexplore.ieee.org/document/10174663.

[Zheng 2022] Mingkai Zheng, Shan You, Lang Huang, Fei Wang, Chen Qian et Chang Xu. *SimMatch : Semi-supervised Learning with Similarity Matching*, 2022. URL : https://arxiv.org/pdf/2203.06915.pdf.

[Zhou 2022] Zelin Zhou, Zhiling Zhang, Xuenan Xu, Zeyu Xie, Mengyue Wu et Kenny Q. Zhu. *Can Audio Captions Be Evaluated With Image Caption Metrics ?* In ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 981–985, 2022. URL : https://ieeexplore.ieee.org/abstract/document/9746427.

[Zhu 2018] Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang et Yong Yu. *Texygen : A Benchmarking Platform for Text Generation Models*. In The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR '18, page 1097–1100, New York, NY, USA, 2018. Association for Computing Machinery. URL : https://doi.org/10.1145/3209978.3210080.

[Çakır 2020] Emre Çakır, Konstantinos Drossos et Tuomas Virtanen. *Multi-Task Regularization Based on Infrequent Classes for Audio Captioning*. In Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020), pages 6–10, Tokyo, Japan, November 2020. URL : https://dcase.community/documents/workshop2020/proceedings/DCASE2020Workshop_Cakir_52.pdf.

Résumé

Dans le domaine de l'audio, la majorité des systèmes d'apprentissage automatique se concentrent sur la reconnaissance d'un nombre restreint d'événements sonores. Cependant, lorsqu'une machine est en interaction avec des données réelles, elle doit pouvoir traiter des situations beaucoup plus variées et complexes. Pour traiter ce problème, les annotateurs ont recours au langage naturel, qui permet de résumer n'importe quelle information sonore. La Description Textuelle Automatique de l'Audio (DTAA ou *Automated Audio Captioning* en anglais) a été introduite récemment afin de développer des systèmes capables de produire automatiquement une description de tout type de son sous forme de texte. Cette tâche concerne toutes sortes d'événements sonores comme des sons environnementaux, urbains, domestiques, des bruitages, de la musique ou de parole. Ce type de système pourrait être utilisé par des personnes sourdes ou malentendantes, et pourrait améliorer l'indexation de grandes bases de données audio.

Dans la première partie de cette thèse, nous présentons l'état de l'art de la tâche de DTAA au travers d'une description globale des jeux de données publics, méthodes d'apprentissage, architectures et métriques d'évaluation. À l'aide de ces connaissances, nous présentons ensuite l'architecture de notre premier système de DTAA, qui obtient des scores encourageants sur la principale métrique de DTAA nommée SPIDEr : 24,7% sur le corpus Clotho et 40,1% sur le corpus AudioCaps.

Dans une seconde partie, nous explorons de nombreux aspects des systèmes de DTAA. Nous nous focalisons en premier lieu sur les méthodes d'évaluations au travers de l'étude de SPIDEr. Pour cela, nous proposons une variante nommée SPIDEr-max, qui considère plusieurs candidats pour chaque fichier audio, et qui montre que la métrique SPIDEr est très sensible aux mots prédits. Puis, nous améliorons notre système de référence en explorant différentes architectures et de nombreux hyperparamètres pour dépasser l'état de l'art sur AudioCaps (SPIDEr de 49,5%). Ensuite, nous explorons une méthode d'apprentissage multitâche visant à améliorer la sémantique des phrases générées par notre système. Enfin, nous construisons un système de DTAA généraliste et sans biais nommé CoNeTTE, pouvant générer différents types de descriptions qui se rapprochent de celles des jeux de données cibles.

Dans la troisième et dernière partie, nous proposons d'étudier les capacités d'un système de DTAA pour rechercher automatiquement du contenu audio dans une base de données. Notre approche obtient des scores comparables aux systèmes dédiés à cette tâche, alors que nous utilisons moins de paramètres. Nous introduisons également des méthodes semi-supervisées afin d'améliorer notre système à l'aide de nouvelles données audio non annotées, et nous montrons comment la génération de pseudo-étiquettes peut impacter un modèle de DTAA. Enfin, nous avons étudié les systèmes de DTAA dans d'autres langues que l'anglais : français, espagnol et allemand. De plus, nous proposons un système capable de produire les quatre langues en même temps, et nous le comparons avec les systèmes spécialisés dans chaque langue.

Mots-clés: description textuelle automatique de l'audio, apprentissage profond, modélisation bout-en-bout multimodale, événements sonores

Abstract

In the audio research field, the majority of machine learning systems focus on recognizing a limited number of sound events. However, when a machine interacts with real data, it must be able to handle much more varied and complex situations. To tackle this problem, annotators use natural language, which allows any sound information to be summarized. Automated Audio Captioning (AAC) was introduced recently to develop systems capable of automatically producing a description of any type of sound in text form. This task concerns all kinds of sound events such as environmental, urban, domestic sounds, sound effects, music or speech. This type of system could be used by people who are deaf or hard of hearing, and could improve the indexing of large audio databases.

In the first part of this thesis, we present the state of the art of the AAC task through a global description of public datasets, learning methods, architectures and evaluation metrics. Using this knowledge, we then present the architecture of our first AAC system, which obtains encouraging scores on the main AAC metric named SPIDEr : 24.7% on the Clotho corpus and 40.1% on the AudioCaps corpus.

Then, subsequently, we explore many aspects of AAC systems in the second part. We first focus on evaluation methods through the study of SPIDEr. For this, we propose a variant called SPIDEr-max, which considers several candidates for each audio file, and which shows that the SPIDEr metric is very sensitive to the predicted words. Then, we improve our reference system by exploring different architectures and numerous hyper-parameters to exceed the state of the art on AudioCaps (SPIDEr of 49.5%). Next, we explore a multi-task learning method aimed at improving the semantics of sentences generated by our system. Finally, we build a general and unbiased AAC system called CoNeTTE, which can generate different types of descriptions that approximate those of the target datasets.

In the third and last part, we propose to study the capabilities of a AAC system to automatically search for audio content in a database. Our approach obtains competitive scores to systems dedicated to this task, while using fewer parameters. We also introduce semi-supervised methods to improve our system using new unlabeled audio data, and we show how pseudo-label generation can impact a AAC model. Finally, we studied the AAC systems in languages other than English : French, Spanish and German. In addition, we propose a system capable of producing all four languages at the same time, and we compare it with systems specialized in each language.

Keywords: automated audio captioning, deep learning, end-to-end multimodal modelling, sound events