Apprentissage profond faiblement supervisé et semi-supervisé pour la détection d'évènements sonores

Léo Cances

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *14/12/2021* **par :**

Léo Cances

Apprentissage profond faiblement supervisé et semi-supervisé pour la détection d'évènements sonores

---

JURY

Geoffroy Peeters	Professeur titulaire à l'école d'ingénieurs Télécom Paris	Président
Alain Rakotomamonjy	Professeur de l'Université de Rouen	Membre
Sylvie Chambon	Enseignante-Chercheuse à ENSEEIHT	Membre
Thomas Pellegrini	directeur de thèse	Membre

École doctorale et spécialité :
MITT : Domaine STIC : Intelligence Artificielle
Unité de Recherche :
Institut de Recherche en Informatique de Toulouse
Directeur de Thèse :
Thomas Pellegrini
Rapporteurs :
Geoffroy Peeters et *Alain Rakotomamonjy*

Résumé

La quantité de données produite par les médias tel que Youtube est une mine d'or d'information pour les algorithmes d'apprentissage machine. Une mine d'or inatteignable tant que ces informations n'ont pas été raffinées. Pour les algorithmes dits supervisés, il est nécessaire d'associer à chaque information disponible une étiquette permettant de l'identifier et de l'utiliser. C'est un travail fastidieux, lent et coûteux, réalisé par des annotateurs humains de manière bénévole ou professionnellement. Cependant, la quantité d'information générée chaque jour excède largement nos capacités d'annotation humaine. Il est alors nécessaire de se tourner vers des méthodes d'apprentissage capables d'utiliser l'information dans sa forme brute ou légèrement travaillée. Cette problématique est au coeur de ma thèse, où il s'agit, dans une première partie, d'exploiter des annotations humaines dites « faibles », puis d'exploiter des données partiellement annotées dans une seconde partie.

La détection automatique d'évènements sonores polyphoniques est une problématique difficile à résoudre. Les évènements sonores se superposent, se répètent et varient dans le domaine fréquentiel même au sein d'une même catégorie. Toutes ces difficultés rendent la tâche d'annotation encore plus difficile, non seulement pour un annotateur humain, mais aussi pour des systèmes entraînés à la classification. La classification audio de manière semi-supervisée, c'est-à-dire lorsqu'une partie conséquente du jeu de données n'a pas été annotée, est l'une des solutions proposées à la problématique de l'immense quantité de données générée chaque jour. Les méthodes d'apprentissage profond semi-supervisées sont nombreuses et utilisent différents mécanismes permettant d'extraire implicitement des informations des données non-annotées, les rendant ainsi utiles et directement utilisables.

L'objectif de cette thèse est dans un premier temps, d'étudier et proposer des approches faiblement supervisées pour la tâche de détection d'évènements sonores, mises en oeuvre lors de notre participation à la tâche quatre du défi international DCASE. Il s'agit ici d'enregistrements audio faiblement supervisés réalistes, de type bruits domestique. Afin de résoudre cette tâche, nous avons proposé deux solutions fondées sur les réseaux de neurones convolutifs récurrents, ainsi que sur des hypothèses statistiques contraignant l'entraînement. Dans un second temps, nous nous pencherons sur l'apprentissage profond semi-supervisé, lorsqu'une majorité de l'information n'est pas annotée. Nous comparons des approches développées pour la classification d'images au départ, avant de proposer leur application

À la classification audio. Nous montrons que les approches les plus récentes permettent d'obtenir des résultats aussi bons qu'un entraînement entièrement supervisé, qui lui aurait eu accès à l'intégralité des annotations.

Remerciements

J'ai commencé ce doctorat avec beaucoup d'appréhension. Mon master portait principalement sur la programmation embarquée pour les systèmes critiques et n'avait pas grand-chose à voir avec l'apprentissage machine / profond. J'ai eu l'occasion de travailler sur ce sujet pendant mon stage de fin d'études pendant lequel j'ai pu saisir l'ampleur de cette discipline et j'ai voulu l'explorer. J'ai rencontré des personnes vraiment incroyables et talentueuses qui m'ont aidé pendant ces trois années et je leur en serai toujours reconnaissant.

Tout d'abord, mes remerciements les plus sincères à mon superviseur Thomas Pellegrini. C'est grâce à ses conseils, sa patience et ses connaissances que j'ai réussi à terminer ce doctorat.

Mes sincères remerciements aux rapporteurs et examinateurs de mon jury, Geoffroy Peeters, Alain Rakotomanojy, et Sylvie Chambon pour leurs commentaires perspicaces et leurs suggestions constructives.

Cette thèse n'aurait pas pu être faite sans le soutien financier de l'Agence Nationale de la Recherche (ANR) avec le projet LUDAU dirigé par mon directeur de thèse Thomas Pellegrini. Mais aussi sans le soutien du laboratoire dans lequel j'ai travaillé : L'Institut de Recherche en Informatique de Toulouse (IRIT) et de l'Université Paul Sabatier.

Je tiens aussi à remercier chaleureusement les chercheurs et collègues doctorants de l'équipe SAMoVA de l'IRIT : Julien Pinquier, Jérôme Farinas, Christine Senac, Isabelle Ferrané, Julie Mauclair, Etienne Labbé, Timothy Pommée, Benjamin Chamand, Lucile Gelin, Vincent Roger, Robin Vaysse, Jim petiot, Verdiana De Fino, Mathieu Balaguer, Lila Gravellier, Estelle Randria, Sebastiao Quintas. Ils sont aussi la raison pour laquelle j'ai pu relever le défi de cette thèse : leurs connaissances et leurs capacités à partager sans retenue m'ont vraiment aidé. Je tiens à m'excuser pour ma capacité infinie à parler sans m'arrêter et pour la distraction que cela a pu causer.

Par-dessus tout, je remercie ma famille d'avoir toujours été là pour moi, de m'avoir suivi dans mes difficultés et de m'avoir guidé vers un avenir meilleur, en veillant à ce que je n'ai aucun regrets. Ils n'ont jamais cessé de croire en moi.

Léo Cances

Table des matières

Table des figures

Liste des tableaux

Glossaire

ANR Agence Nationale de la Recherche. 21

AT Audio Tagging. 23, 76

BN Batch Normalization. 34

CIFAR-10 Canadian Institute For Advanced Research - 10. 13, 15, 24, 46, 85, 90–92, 94, 100, 101, 105, 132

CNN Convolutional Neural Network. 11, 12, 27, 28, 30, 38, 45, 58, 60, 61, 76, 90, 93

CRNN Convolutional Recurrent Neural Network. 11, 12, 39–41, 51–53, 55, 57–62, 65, 68, 76, 80, 131

CT Co-Training. 86

DCASE Detection and Classification of Acoustic Scenes and Events. 37, 39, 40, 45, 51, 52, 58, 59, 64, 65, 74, 75, 108, 131

DCT Deep Co-Training. 12, 13, 16, 24, 25, 45, 85–94, 99, 100, 102, 104, 105, 107, 108, 110, 122, 123, 125–129, 131, 132, 134–136

DCT-T Deep Co-Training & Teacher. 16, 133–135

DESED Domestic Environment Sound Event Detection. 40

DNN Deep Neural Network. 35, 44

EMA Exponential Moving Average. 45, 135

ESC Environmental Sound Classification Dataset. 123, 128, 129

ESC-10 Environmental Sound Classification Dataset. 15, 16, 107, 120, 124–127, 134, 135

FC Fully Connected Network. 61, 91, 93

FGSM Fast Gradient Signed Method. 88, 89, 91, 93

FM FixMatch. 46, 107, 108, 114, 116, 122, 123, 125–129, 132

GMM Gaussian Mixture Model. 52

GRU Gated Recurrent Unit. 12, 32, 33, 51, 52, 55–57, 60, 131

GSC Google Speech Command Dataset. 13, 15, 16, 96, 97, 107, 116–118, 121, 124–128, 134

HMM Hidden Markov Model. 52

ICT Interpolation Consistency Training. 42

LReLU Leaky Rectified Linear Activation. 90

LSTM Long short-term memory. 31, 32, 69

LUDAU Lightly-supervised and Unsupervised Discovery of Audio Units using Deep Learning. 21

MIL Multiple Instance Learning. 57

MLP Multi Layer Perceptron. 27, 28

MM MixMatch. 25, 43, 46, 107, 108, 114–116, 119, 122, 123, 125–129, 132

MMM Min Mean Max. 12, 15, 51, 55, 57–65, 67, 74, 76–80, 131, 132

MSE Mean Square Error. 44, 110

MT Mean-Teacher. 41, 45–47, 63, 91, 107, 108, 110, 122, 123, 125–129, 132, 134, 135

ReLU Rectified Linear Unit. 93

ResNet Residual Network. 29

RMM ReMixMatch. 46, 107, 108, 112, 114–116, 122, 123, 125–129, 132, 133

RMSE Root Mean Square Error. 103, 156

RNN Recurrent Neural Network. 31, 53, 55, 56, 68

SED Sound Event Detection. 11, 23, 40, 52, 53, 69, 131, 132

SGD Stochastic Gradient Descent. 91, 93

SL Supervised Learning. 22

SNR Signal to Noise Ration. 13, 96, 102

SSL Semi Supervised Learning. 23, 41–43, 46, 47, 86, 91, 93, 108, 119, 122, 123, 125, 127, 129, 132, 133, 136, 137

SVHN The Street View House Numbers. 46, 85

SVM Support Vector Machine. 38

UBS8K UrbanSound 8K Dataset. 13, 15, 16, 25, 85, 92–94, 97, 99–102, 105, 107, 121, 123–128, 132, 134, 135

UL Unsupervised Learning. 22

VAT Virtual Adversarial Training. 42, 43, 45

WGRU Weighted Gated Recurrent Unit. 12, 15, 24, 51, 55–57, 59–64, 131

WSL Weakly Supervised Learning. 22

Introduction

Projet LUDAU

L'apprentissage semi-supervisé est une branche de l'apprentissage profond en forte évolution. Il est principalement expérimenté sur des tâches de catégorisation d'objets dans les images, mais il est peu développé dans le traitement audio. Le projet de l'Agence Nationale de la Recherche (ANR) Lightly-supervised and Unsupervised Discovery of Audio Units using Deep Learning (LUDAU) (ANR-18-CE23-0005-01) a pour objectif de remédier à cette situation en proposant d'explorer différentes méthodes pour la classification et la détection d'évènements sonores. Pour ce faire, deux scénarios seront traités :

1. Un scénario faiblement supervisé, pour lequel des annotations faibles sont disponibles, c'est-à-dire qu'elles décrivent globalement un enregistrement audio.
2. Un scénario semi-supervisé, pour lequel les annotations ne sont disponibles que pour une fraction des enregistrements audio.

Motivation

L'apprentissage profond est de plus en plus utilisé et l'état de l'art évolue rapidement. Il existe de nombreux domaines d'application comme la classification de contenu multimédia, la traduction de documents ou encore la détection d'anomalie. Dans cette thèse, on s'intéresse à la tâche de détection et de classification d'évènements sonores. L'apprentissage profond est basé sur des réseaux de neurones artificiels qui permettent d'extraire des données qu'on leur fournit, les caractéristiques nécessaires pour résoudre une tâche spécifique. On peut noter, depuis quelques années qu'il y a un engouement de plus en plus important pour l'application de l'apprentissage profond au domaine audio. On observe l'émergence de jeux de données très larges, engendrant les mêmes problématiques d'annotations que les jeux de données d'images ont rencontrées quelques années plus tôt. En effet, le travail nécessaire pour acquérir de grandes quantités de données et les rendre exploitables nécessite énormément d'effort, de temps et d'argent. À titre d'exemple, le jeu de données le plus populaire pour la classification d'images, ImageNet [Deng et al., 2009], a mis trois ans pour voir sa première

version annotée en 2009, grâce à la contribution de 49 000 travailleurs répartie dans 167 pays par la plateforme *Amazon Mechanical Turk* [1].

Pour l'audio, on retrouvera deux jeux de données particulièrement importants. Audioset [Gemmeke et al., 2017] est composé de 2,1 millions d'enregistrements audio de 10 secondes extraits de vidéo YouTube. Il est divisé en 632 catégories, pouvant chacune contenir de multiples évènements distincts, rendant l'annotation plus fastidieuse. Les catégories sont automatiquement attribuées à partir des tags des vidéos d'origines, ainsi le rôle de l'annotateur humain consiste à vérifier la validité de ces labels. Freesound [Font et al., 2013] est un autre jeu de données de grande ampleur qui utilise une approche communautaire pour acquérir et annoter des enregistrements audio. L'initiative a démarré en 2005 et atteignait, en 2020, plus de 483 000 fichiers audio [2] grâce à 24 000 contributeurs [3].

De tels volumes de données ne sont pas directement exploitables et ne peuvent être annotés dans des temps acceptables. On travaille souvent avec des jeux de données qui ne représentent qu'une infime fraction des informations disponibles, dont les étiquettes simples peuvent parfois être générées automatiquement. Il existe alors plusieurs approches pour compenser le manque de données labélisées, et ainsi utiliser avec plus d'efficacité l'énorme quantité d'information disponible, ce sont les paradigmes d'apprentissage faiblement supervisé et semi-supervisé qui seront au coeur de ma thèse.

Paradigmes d'apprentissage faiblement et semi-supervisé

L'apprentissage supervisé (Supervised Learning (SL)) utilise des données annotées pour entraîner un réseau de neurones profond sur une tâche particulière. Chacun des exemples dans le jeu de données possède une correspondance avec un label. Lors de la phase d'entraînement, chaque erreur de prédiction du système fera l'objet d'une mise à jour des poids du modèle et permettra à ce dernier de corriger ses prédictions, améliorant ses performances petit à petit. Différentes approches supervisées ont été développées et sont devenues la norme au fil du temps, ainsi que plusieurs architectures de réseaux de neurones tel que les réseaux convolutifs et récurrents.

Avec l'apprentissage non supervisé (Unsupervised Learning (UL)), les données utilisées pour l'entraînement du système n'ont pas de label associé. L'objectif de ces systèmes est de trouver des similitudes dans les données pour les regrouper en clusters. Les tâches les plus communes de l'apprentissage non supervisé sont donc le clustering, mais aussi la réduction de dimension avec des techniques qui ne s'appuient pas sur des labels associés aux données brutes.

L'apprentissage faiblement supervisé (Weakly Supervised Learning (WSL)) est une autre

1. https ://image-net.org/static_files/files/imagenet_ilsvrc2017_v1.0.pdf
2. https ://blog.freesound.org/ ?p=1291
3. https ://blog.freesound.org/ ?p=1084

branche de l'apprentissage machine qui, par exemple, utilise des annotations incomplètes pour entraîner un système sur une tâche de classification. Les annotations peuvent être imprécises, parfois générées automatiquement par d'autres systèmes automatiques. L'objectif étant de pouvoir utiliser des jeux de données bien plus conséquents, mais avec des annotations de moins bonnes qualités. De ce fait, un certain nombre de techniques sont utilisées pour mitiger l'impact de ces données, aussi appelées « données faibles ».

L'apprentissage semi-supervisé (Semi Supervised Learning (SSL)) est un mélange d'apprentissages supervisé et non supervisé. Les méthodes SSL doivent alors composer avec des ensembles de données dont seule une petite fraction du contenu est annotée. Cette dernière sera utilisée pour faire converger le système dans la bonne direction tandis que la partie non annotée sera utilisée pour améliorer la généralisation du système, c'est-à-dire de prédire correctement des exemples qu'il n'a jamais vus pendant son entraînement.

Ces paradigmes ont une longue histoire dans l'apprentissage automatique [Bishop, 2006] et ils ont été remis au goût du jour avec l'avènement de l'apprentissage profond. De nombreux algorithmes de « l'ère pré-deep learning » ont été adaptés à l'apprentissage de réseaux de neurones, comme par exemple, l'algorithme de co-training, qui met en jeu deux ou plusieurs vues des données, qui a été adapté en une version « deep co-training ». L'apprentissage profond fera l'objet de la majeure partie de mon manuscrit.

Détection et classification d'évènements sonores

Dans cette thèse, j'ai étudié les deux tâches de classification et de détection d'évènements sonores.

Classification ou Audio Tagging

La classification d'évènements sonores (Audio Tagging (AT)) consiste à prédire la présence d'un évènement sonore dans un enregistrement audio. Cet évènement peut-être présent soit dans une partie du signal, soit dans son intégralité. Il est possible que plusieurs évènements différents soient présents dans l'enregistrement, on parlera alors de classification multilabel par opposition à monolabel.

Détection d'évènements sonores

La détection d'évènements sonores (Sound Event Detection (SED)) est décomposée en deux sous-tâches. La première est la tâche de classification (AT), et la seconde une tâche de localisation dans le temps. Dans un enregistrement audio réel, on trouvera très souvent plusieurs évènements qui se superposent, on parle alors de détection d'évènements sonores

polyphoniques.

On parlera d'annotations fortes lorsque les temps de début et de fin des évènements sont connus. En revanche, on parlera d'annotations faibles que lorsque les tags sont connus, mais pas les temps de début et de fin. Dans la première partie de cette thèse, je me suis intéressé à l'apprentissage faiblement supervisé qui utilise des tags pour l'entraînement d'un réseau de neurones, pour ensuite essayer de prédire simultanément des tags et des annotations fortes. Dans la deuxième partie de cette thèse, je me suis concentré sur l'amélioration de modèles de prédiction de tags, à l'aide de données non annotées. Il s'agit dans ce cas d'apprentissage profond semi-supervisé.

Organisation

Ce manuscrit est organisé en deux parties. Après le chapitre état de l'art, la première partie porte sur l'apprentissage faiblement supervisé pour l'audio tagging et la détection, avec les chapitres 2 et 3. La deuxième partie, plus conséquente, s'intéresse aux algorithmes d'apprentissage profond semi-supervisé pour l'audio tagging, avec les chapitres 4 et 5.

Le contenu global des chapitres est le suivant :

- Dans le chapitre 1, nous donnons un aperçu général des réseaux de neurones profonds et des architectures les plus répandues, avant de continuer avec un état de l'art sur l'apprentissage faiblement supervisé puis semi-supervisé.
- Dans le chapitre 2, nous présentons les expériences de classification et de localisation d'évènements sonores, réalisées dans le cadre d'un apprentissage faiblement supervisé. Ces travaux ont abouti à une première contribution, nommée Weighted Gated Recurrent Unit (WGRU), qui désigne une modification faite à une cellule récurrente en phase d'inférence. Les résultats obtenus à notre participation au challenge DCASE 2018 seront décrits.
- Dans le chapitre 3, nous présentons une autre contribution faisant suite au chapitre 2, consistant à fournir une comparaison de plusieurs méthodes de post-processing des prédictions temporelles d'un réseau pour obtenir une meilleure localisation des évènements sonores. Ces travaux ont abouti à la mise à disposition open-source d'une boite à outils permettant d'effectuer ce post-processing de manière optimale.
- Dans le chapitre 4, nous nous concentrons sur l'étude d'une méthode d'apprentissage nommée Deep Co-Training (DCT) qui, en 2019, était l'état de l'art de la classification semi-supervisée appliquée à la détection d'objets dans les images. Après la description théorique de la méthode et le fait d'avoir reproduit les résultats sur le jeu de données d'images Canadian Institute For Advanced Research - 10 (CIFAR-10), nous décrivons l'adaptation qui a été faite pour l'application à des données audio, en particulier sur

le jeu UrbanSound 8K Dataset (UBS8K). Nous avons ensuite amélioré l'algorithme original en ajoutant des augmentations de données audio en plus de la génération d'exemples adversaires.

- Enfin dans le chapitre 5, nous présenterons une comparaison des performances de DCT avec d'autres approches semi-supervisées proposées après DCT, que nous avons dû adapter à la classification audio. Les résultats obtenus permettent de démontrer l'efficacité de ces méthodes en général, pour la modalité audio. De plus, les méthodes plus récentes que le DCT, et n'utilisant qu'un seul modèle au lieu de deux comme c'est le cas pour le DCT, se sont révélées bien plus performantes, en particulier le MixMatch (MM).

Chapitre 1

État de l'art

Dans ce chapitre, nous donnons dans un premier temps un aperçu général des réseaux de neurones profonds et des architectures les plus répandues. Nous commencerons par présenter très brièvement les réseaux entièrement connectés, les réseaux convolutifs, résiduels, et pour finir les réseaux récurrents. Dans un second temps, nous faisons un état de l'art de l'apprentissage faiblement supervisé pour la détection d'évènements sonores ainsi que l'apprentissage semi-supervisé pour la classification d'évènements (*audio tagging*).

1.1 Bref aperçu des réseaux convolutifs et récurrents

Dans cette section, nous présentons les réseaux de neurones artificiels ainsi que les architectures les plus populaires. Les réseaux résiduels et récurrents seront particulièrement détaillés car je les ai utilisé tout au long de ma thèse.

1.1.1 Les réseaux convolutifs

Les réseaux convolutifs sont très utilisés encore de nos jours, à la fois pour le traitement d'images et l'analyse d'enregistrements sonores. [Fukushima and Miyake, 1982] ont proposé les Convolutional Neural Network (CNN) pour surmonter les limitations du perceptron multi-couche (Multi Layer Perceptron (MLP)). Inspiré par le cortex visuel des animaux, l'idée est que chaque neurone reçoive des signaux provenant de champs réceptifs qui se chevauchent et couvrent l'ensemble du champ « visuel » . En pratique, nous utilisons une quantité définie de champs réceptifs, également appelés filtres. Les nombreux neurones partageront alors les mêmes filtres (ainsi que les poids et le biais de ces derniers). Au fil des années, cette approche a été améliorée et affinée par [Lecun et al., 1998].

Un CNN comporte généralement plusieurs couches de convolutions successives. Entre elles on retrouvera souvent une fonction d'activation puis une couche de *pooling* dont l'objectif est de sous-échantillonner leur sortie avant de la fournir à la couche suivante. Cette

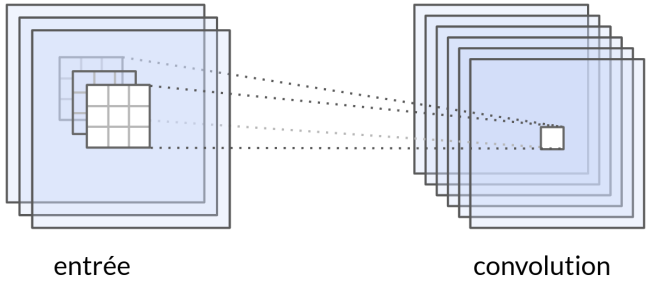

Figure 1.1 – Illustration de deux couches de convolution dans un CNN. La couche d'entrée correspond à une image avec ses trois matrices de couleur. La couche suivante est une couche de convolutions composée de 6 filtres et un champ réceptif de 3x3.

première série de couches permet de réaliser l'extraction des caractéristiques de l'entrée. Ensuite on retrouve le second bloc de couches entièrement connectées qui permettront de faire la classification. L'image 1.2 représente un CNN avec trois couches convolutives suivies de deux couches entièrement connectées.

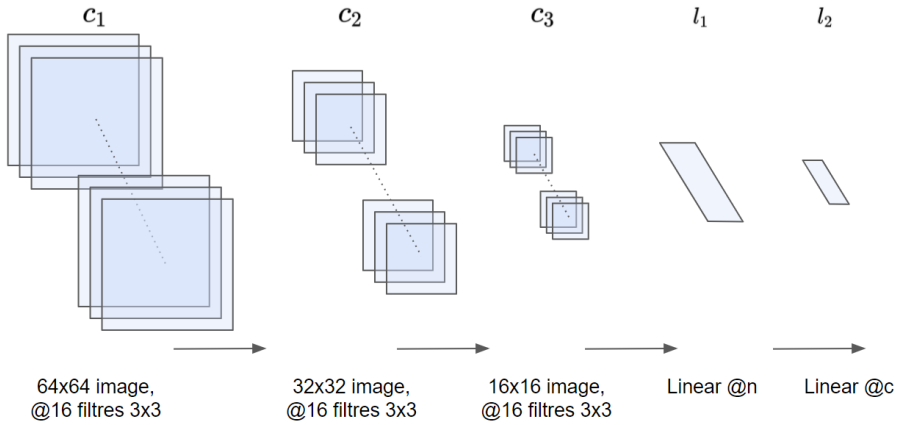

Figure 1.2 – Illustration simplifiée d'un CNN avec trois couches convolutives suivies de deux couches entièrement connectées. Entre chaque couche de convolution c_i on retrouvera une fonction d'activation suivi d'un *pooling*. Les couches l_i sont des denses.

De nos jours, les jeux de données sont de plus en plus grands avec un très grand nombre de catégories. Par conséquent, il est nécessaire d'augmenter les capacités des CNN. Pour cela, on augmente le nombre de couches du modèle, le rendant plus « profond » et, on augmente le nombre de filtres des couches. Avec l'augmentation du nombre de couches et de filtres, le nombre de paramètres entraînables augmente tout aussi rapidement que le MLP. Deux problèmes apparaissent alors :

— Le gradient évanescent qui est particulièrement présent dans les réseaux de neurones très profonds. Le gradient d'erreur dans les couches supérieures du modèle est trop petit pour permettre aux paramètres de ces couches d'être mis à jour de manière significative durant l'entraînement, les rendant « fixes » et par conséquent inutiles.

— Le sur-apprentissage des réseaux de neurones possédant un trop grand nombre de paramètres comparés à la taille du jeu de données utilisé pour l'entraîner. Pour ce dernier, le modèle sera performant, mais incapable de généraliser et prédire de nouvelles données.

Il existe de nombreuses architectures de modèles tel que AlexNet [Krizhevsky et al., 2017], VGG [Simonyan and Zisserman, 2015], ou encore Inception [Szegedy et al., 2014]. Certaines architectures sont construites pour pallier au problème du gradient évanescent, par exemple l'architecture ResNet [He et al., 2015], Wide ResNet [Zagoruyko and Komodakis, 2016], ou DenseNet [Huang et al., 2018]. Dans la section 1.1.2 nous décrivons l'architecture ResNet et WideResNet en détail.

Concernant le sur-apprentissage, de nouvelles techniques de régularisation sont apparues au fils des années tel que le *weight decay* [Krogh and Hertz, 1991], le dropout [Srivastava et al., 2014], la *Batch Normalisation* [Ioffe and Szegedy, 2015], ou encore l'augmentation de donnée. Ces méthodes sont décrites dans la section 1.1.4.

1.1.2 Les réseaux résiduels

Le problème de gradient évanescent est directement lié au nombre de couches qui composent un réseau de neurones. L'idée derrière les réseaux résiduels est d'insérer des connexions intermédiaires entre deux couches distantes. Ces liaisons *skip* sont représentées par la figure 1.3.

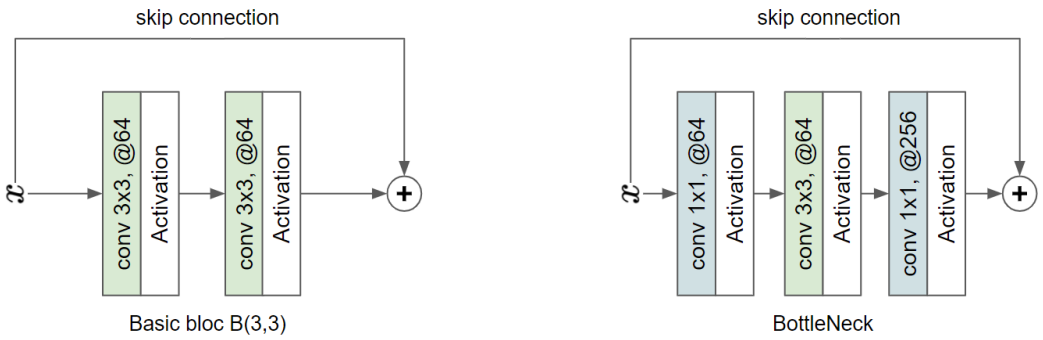

Figure 1.3 – Représentation d'une connexion *skip* permettant de « sauter » deux couches de convolutions avec leur activation.

Il est alors possible de construire des modules contenant n couches de convolutions et de les empiler, permettant la construction de réseaux de neurones plus profonds qu'on nomme alors, Residual Network (ResNet) [He et al., 2015] et peuvent être composés de milliers de couches (1202). Le bloc de droite est utilisé pour augmenter encore la profondeur des réseaux. Cette liaison *skip* permet de diminuer le nombre de paramètres entraînables et ainsi la complexité du modèle. Les couches de convolutions 1x1 permettent de réduire puis

d'augmenter la dimension en entrée et sortie du bloc résiduel, évitant ainsi une explosion du nombre de paramètres entraînables.

Ainsi, pour la première couche du module résiduel, si le gradient d'erreur vient du module précédant via une série de multiplications au travers des couches intermédiaires, on retrouve aussi un autre gradient d'erreur, transféré directement à la première couche du module grâce à la connexion *skip*. De cette façon, on peut éviter le problème du gradient évanescent.

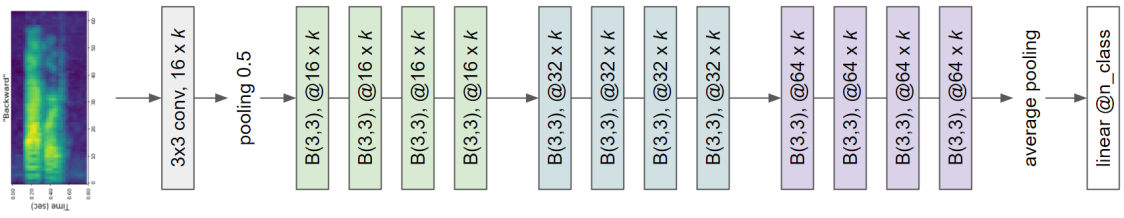

Figure 1.4 – Représentation d'un réseau résiduel de 28 couches. $B(3,3)$ représente un bloc résiduel contenant 2 couches de convolutions ayant un champ réceptif de 3×3.

Dans leurs travaux, [Zagoruyko and Komodakis, 2016] proposent d'utiliser *Wide ResNet* qui est une amélioration du *ResNet*. Ils montrent qu'augmenter la largeur d'un réseau (nombre de filtres de convolutions) est plus efficace que d'augmenter sa profondeur. Un *Wide ResNet* profond de 16 couches, mais 8 fois plus large, permettrait alors d'obtenir de meilleures performances qu'un *ResNet* profond de 110 couches pour la même tâche et sur les même jeux de données. La figure 1.4 permet de représenter un *Wide Resnet* de 28 couches dont l'architecture sera utilisée pour réaliser plusieurs de nos expériences dans cette thèse.

1.1.3 Les réseaux récurrents

Les CNN sont très performants lorsqu'il s'agit d'apprendre des liens entre une entrée et la sortie attendue. Cependant, ce type de modèle n'est pas bien adapté aux tâches qui nécessitent un contexte passé pour faire des prédictions futures, et lorsque les enregistrements sont de durée variable.

C'est pour ce genre de tâches qu'interviennent les réseaux récurrents. Ces derniers prennent en compte, en plus de la donnée d'entrée, l'état dans lequel il se trouvait avant. L'image 1.5 montre une représentation d'un réseau de neurones récurrent avec, À gauche, une cellule récurrente classique dans sa représentation la plus simple, et à droite sa forme dépliée. On peut voir qu'à chaque instant t, la cellule prend en compte, à la fois l'entrée actuelle x_t et le contexte des entrées précédentes, ici appelé v. C'est cette structure qui permet aux modèles récurrents de garder en « mémoire » le passé pour pouvoir prédire ce qui va suivre. Ils capturent le comportement temporel des données séquentielles [Hochreiter and Schmidhuber, 1997] et on peut les formuler de la façon suivante :

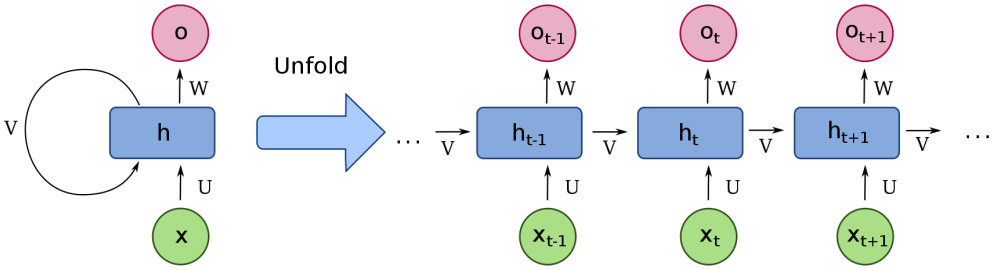

Figure 1.5 – Diagramme d'un réseau neuronal récurrent (Recurrent Neural Network (RNN)). De bas en haut : *x* état d'entrée, *h* état caché et *o* état de sortie. U, V, W sont les poids du réseau. Diagramme compressé à gauche et version dépliée à droite (à chaque instant *t*). – (https ://commons.wikimedia.org/w/index.php ?curid=60109157)

$$h_t = g(Wx_t + Uh_{t-1} + b) \tag{1.1}$$

où g est une fonction d'activation, et W et U sont des matrices de poids à apprendre avec le biais b.

Long Short-Term Memory LSTM

En théorie, les RNN classiques peuvent garder en mémoire les contextes des évènements qui se sont produits longtemps dans le passé. Cependant, un phénomène de disparition (*gradient vanishing*) ou d'explosion du gradient apparaît lorsqu'un trop grand nombre de rétro-propagations entre en jeu.

La cellule récurrente Long short-term memory (LSTM) [Hochreiter and Schmidhuber, 1997], schématisée par la figure 1.6 et autrement plus complexe, permet de résoudre une partie du problème ci-dessus. Elle est complétée par des « portes » récurrentes appelées « porte d'oubli » (*forget gate* f), « porte de sortie » (*output gate* o), et la « porte de réinitialisation » (*reset gate* r) qui permettent de sélectionner parmi le contexte passé, l'information la plus pertinente qui doit être conservée. Les gradients d'erreurs peuvent refluer à travers un nombre virtuellement illimité de couches. Les LSTM sont alors de bons candidats pour apprendre des tâches qui nécessitent la mémoire d'évènements avec des dépendances temporelles longues.

Soit W_g et U_g les matrices de poids entraînables des connexions d'entrées et récurrentes, h_t l'état de la couche à l'instant précédant, x_t l'entrée à l'instant t, et b_g le biais. L'indice g représente la porte considérée, soit f pour *forget gate*, i pour *input gate*, o pour *output gate*, ou c pour la mémoire de la cellule. Les trois portes peuvent être modélisées par les formules 1.3 :

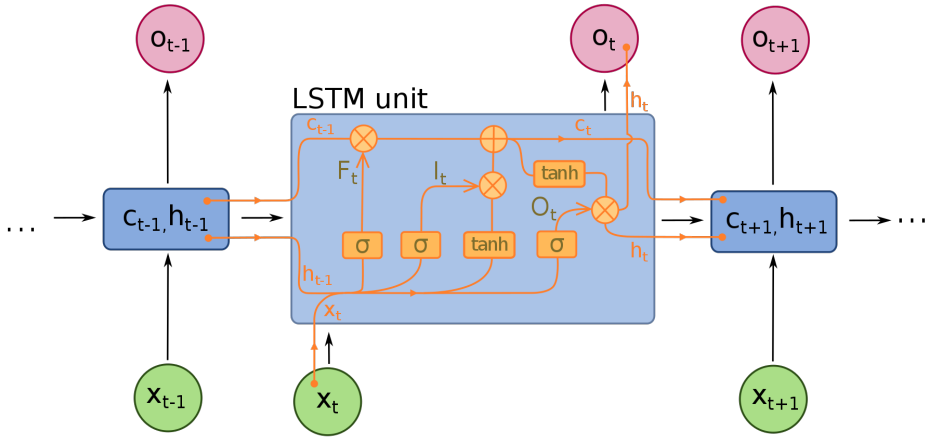

Figure 1.6 – Diagramme d'une mémoire à long court terme (LSTM) d'une cellule. De bas en haut : x état d'entrée, x état caché et c état de la cellule, o état de sortie. Les portes sont des sigmoïdes (σ) ou des tangentes hyperboliques. Autres opérateurs : addition et multiplication par éléments. Les poids ne sont pas affichés. – (https ://commons.wikimedia.org/w/index.php ?curid=60149410)

$$
\begin{aligned}
f_t &= \text{Sigmoid}(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \text{Sigmoid}(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \text{Sigmoid}(W_o x_t + U_o h_{t-1} + b_o)
\end{aligned} \tag{1.2}
$$

Ces trois portes permettent de calculer les *cell state* et les *hidden state*. Ces deux sorties sont définies dans les équations 1.5

$$c'_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{1.3}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot c'_t \tag{1.4}$$

$$h_t = o_t \times \tanh(c_t) \tag{1.5}$$

Gated Recurrent Unit GRU

Les cellules récurrentes Gated Recurrent Unit (GRU) [Chung et al., 2014] ont été introduites en 2014, elles sont une simplification des cellules LSTM tout en étant aussi performantes [Chung et al., 2014] sur des tâches de classification d'image ou de texte [Dey and Salem, 2017; Heck and Salem, 2017]. L'absence de « porte de sortie » (*output gate*) est la différence principale avec la cellule LSTM et lui permet d'avoir moins de paramètres entraînables. Il existe aussi des variantes simplifiées, qui ne seront pas abordées ici. Les deux autres portes sont : R_t (*Reset gate*) et Z_t (*Update Gate*)

L'équation de la cellule GRU est plus simple, car il y a une porte de moins, et il n'y pas de différenciation entre les *cell state* et les *hidden state* :

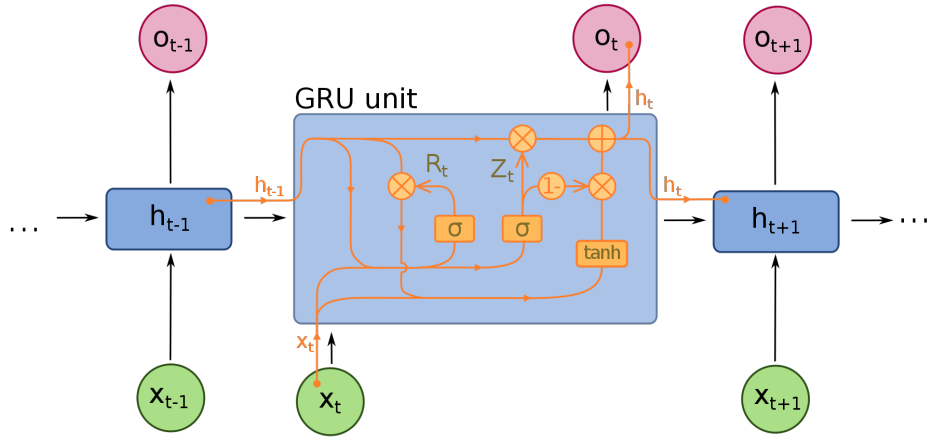

Figure 1.7 – Diagramme d'un modèle récurrent (GRU) à une cellule. De bas en haut : *x* état d'entrée, *h* état caché, *o* état de sortie. Les portes sont des sigmoïdes (σ) ou des tangentes hyperboliques. Autres opérateurs : addition et multiplication par éléments. Les poids ne sont pas affichés. – (https ://commons.wikimedia.org/w/index.php ?curid=60149410)

Soit W_g et U_g les matrices de poids entraînables des connexions d'entrées et récurrentes, h_t l'état de la couche à l'instant précédant, x_t l'entrée à l'instant t, et b_g le biais. L'indice g représente la porte considérée, soit r pour *reset gate*, et z pour *update gate*. Les deux portes peuvent être modélisées par les formules 1.7 :

$$z_t = \text{Sigmoid}(W_z x_t + U_z h_{t-1} + b_z) \tag{1.6}$$

$$r_t = \text{Sigmoid}(W_r x_t + U_r h_{t-1} + b_r) \tag{1.7}$$

Ces portes calculent les candidats h'_t qui permettent de sélectionner les *hidden state* h_t de l'état précédant qui doivent être remis à zéro et mis à jour. La sortie de la couche GRU est alors définie par l'équation 1.10

$$h'_t = \tanh\left(W_h \cdot x_t + U_h(r_t \times h_{t-1}) + b_h\right) \tag{1.8}$$

$$h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot h'_t \tag{1.9}$$

$$o_t = h_t \tag{1.10}$$

1.1.4 Modules complémentaires

En plus des couches convolutives, récurrentes ou entièrement connectées, on retrouve un ensemble de couches intermédiaires ayant différents objectifs tel que le *dropout* ou la *batch normalisation*. Cette section décrit brièvement ces couches, leur fonctionnement et leur rôle au sein du modèle.

Pooling

Cette couche intermédiaire effectue une opération de *pooling* qui consiste en une réduction des dimensions de la couche précédante tout en préservant les caractéristiques importantes. La couche de *pooling* prend en compte un champ à deux dimensions, souvent carré, de la couche précédente pour le réduire en un point. L'algorithme de réduction utilisé peut être un Max, une moyenne ou une somme, bien qu'on utilise très majoritairement une réduction max. Cette couche permet de réduire le nombre de paramètres et de calculs dans le réseau.

Dropout

[Srivastava et al., 2014] ont introduit la couche de *dropout*. Il s'agit d'un composant de la plupart des réseaux de nos jours. Son rôle est de mettre à zéro un nombre aléatoire d'éléments de la sortie de la couche précédente. Le *dropout* est un moyen de régularisation puissant permettant de virtuellement augmenter le nombre d'exemples présentés au système. Il est activé seulement lors de l'entraînement du modèle.

Batch Normalization

La Batch Normalization (BN) [Ioffe and Szegedy, 2015] amène plusieurs bénéfices. Le premier est de rendre le système plus robuste au changement de distribution des entrées de chaque couche, et donc une meilleure généralisation. Il participe aussi à résoudre le problème du gradient évanescent. Lorsque la BN est utilisée directement avant l'activation, la distribution des données est modifiée, poussant les valeurs plus proches de zéro, résultant en des valeurs de dérivation plus importantes.

Pour effectuer la normalisation, il faut calculer la moyenne et la variance du batch. Soit B un batch contenant m exemple x_i, la moyenne et la variance sont calculées comme suit :

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{1.11}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 \tag{1.12}$$

On peut alors normaliser les données d'entrées x_i en y_i en utilisant ces deux informations statistiques ainsi que les paramètres entraînables γ et β.

$$x'_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{1.13}$$

$$y_i = \gamma x'_i + \beta \tag{1.14}$$

Cependant la batch normalisation vient avec son lot de nouveaux problèmes. Elle n'est pas compatible avec les réseaux récurrents, et il est difficile de faire un entraînement distribué sur plusieurs machines ou GPU car il y aura alors de multiple noyaux de normalisation pour les mêmes couches.

1.1.5 Techniques d'entraînement

Lors de l'entraînement d'un réseau de neurones profond, certains mécanismes peuvent être mis en place pour améliorer les performances de ce dernier, converger plus vite, l'aider à mieux généraliser ou éviter des problèmes tel que le gradient évanescent ou explosif.

Décroissance des poids (Weight Decay)

Le *weight decay* [Krogh and Hertz, 1991] est un mécanisme qui empêche les poids entraînables du modèle de devenir trop gros si ce n'est pas nécessaire, réduisant les problèmes de sur-apprentissage. Pour cela, il impacte l'entraînement de deux façons. Premièrement, il peut améliorer la généralisation du système [Hinton, 1987] en réduisant les prédictions parasites (le bruit dans les prédictions). Deuxièmement, il permet de supprimer une partie du bruit statique sur les classes prédites.

Taux d'apprentissage variable (Learning rate scheduler)

Lors de la mise à jour des poids d'un modèle, le gradient est pondéré par un coefficient appelé « taux d'apprentissage » (ou *learning rate* en anglais). Le *learning rate scheduler* a pour objectif d'ajuster le taux d'apprentissage pendant les différentes étapes de l'entraînement du Deep Neural Network (DNN) en utilisant un certain nombre de méthodes. Les travaux de [Wu et al., 2019] présentent en détail 13 différentes fonctions de décroissance du *learning rate* ou de *learning rate* cyclique.

Durant cette thèse, nous avons utilisé la règle du cosinus décroissant définie comme suit et représentée par l'image 1.8 :

$$\mathrm{lr} = \frac{1}{2}\Big(1,0 + \cos\Big((t\text{ - }1)\frac{\pi}{e}\Big)\Big) \tag{1.15}$$

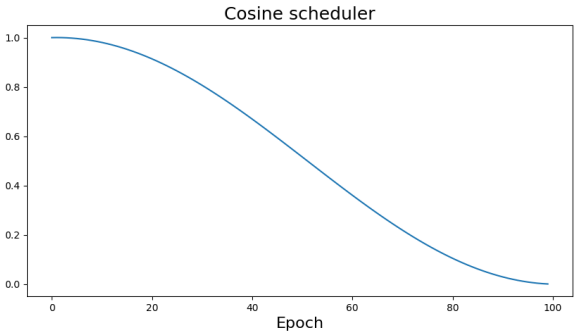

Figure 1.8 – Facteur de décroissance appliqué au *learning rate*, suivant une courbe sinusoïdale.

Échauffement (*Warmup*)

Comme nous allons le voir, lors d'un apprentissage semi-supervisé, la fonction d'entraînement est souvent décomposée en de multiples fonctions de coût. Essentiellement, une pour entraîner le modèle sur des données annotées et au moins une autre pour exploiter les données non-annotées. L'échauffement, (ou *warmup* en anglais) permet de pondérer l'impact des fonctions de coût lors de l'entraînement pour, par exemple, favoriser les données supervisées au début de l'entraînement et incorporer la composante non-supervisée au fur et à mesure de l'entraînement.

Augmentation de données

Le concept de l'augmentation de données est de virtuellement augmenter le nombre d'exemples pour l'entraînement du modèle. Les augmentations utilisées dépendent de la modalité utilisée. Pour un jeu de données d'images, on appliquera des rotations ou des translations aléatoires, on pourra aussi changer le contraste ou les couleurs par exemple. Pour un jeu de données de parole ou d'audio, on modifiera les fréquences du signal, ou on pourra encore le ralentir ou l'accélérer. Il est possible de cette façon d'augmenter les variantes d'exemples pour une même classe, rendant le réseau plus robuste et avec un meilleur pouvoir de généralisation.

Chaque méthode d'augmentation utilisée sera détaillée dans la manuscrit.

1.2 Détection d'évènements sonores avec annotations faibles

Quelle que soit la tâche à accomplir, un apprentissage supervisé est idéal, cependant cela est de moins en moins possible à cause à la taille des jeux de données qui ne cesse de grossir. De plus, les annotations sont souvent créées pour une tâche spécifique et on peut alors retrouver de multiples annotations différentes pour un même jeu de données. Les

étiquettes faibles permettent de réduire l'effort nécessaire pour l'annotation des grands jeux de données et pourraient être utilisées pour plusieurs types de tâches.

Au cours de la première moitié de ma thèse, j'ai travaillé sur la reconnaissance d'évènements sonores faiblement supervisée. C'est justement le sujet d'une thèse publiée très récemment au moment de l'écriture de ce manuscrit, sur des évènements sonores enregistrés en environnement domestique. L'auteur, Nicolas Turpault, propose de générer des prédictions fortes en utilisant des données faiblement annotées. Il décrit les problèmes qui peuvent survenir du fait de l'utilisation de tags à la place d'annotations fortes. Parmi ces problématiques, la plus importante pour une tâche de détection, est que les annotations faibles n'indiquent pas explicitement quelles parties d'un enregistrement le modèle peut utiliser pour apprendre à localiser les différents évènements présents. Il détaille aussi les solutions proposées lors du défi Detection and Classification of Acoustic Scenes and Events (DCASE) dont il est l'un des organisateurs (tâche 4). Les résultats au challenge sont meilleurs d'année en année et tendent à montrer la faisabilité d'utiliser des annotations faibles pour réaliser des prédictions fortes.

1.2.1 Annotations faibles et fortes

L'apprentissage faiblement supervisé a pour objectif d'utiliser des annotations incomplètes (ou « faibles ») contenant une information globale sur un exemple d'apprentissage, pour effectuer une tâche plus précise. Prenons l'exemple concret d'une tâche de détection d'évènements sonores telle que celle présentée dans le chapitre 2. Contrairement à un apprentissage supervisé où chaque occurrence de l'évènement sera correctement fournie (annotations fortes), on entraîne alors un modèle à prédire la segmentation d'évènements sans lui fournir les annotations temporelles. La figure 1.9 permet de visualiser la différence entre des annotations faibles et fortes pour une tâche de détection d'évènements sonores.

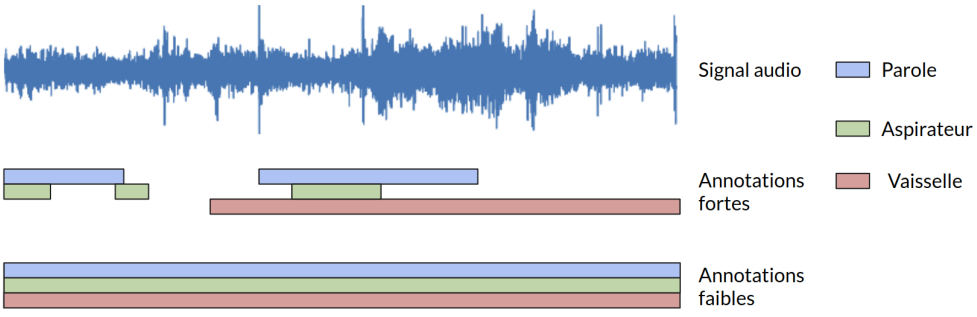

Figure 1.9 – De haut en bas, enregistrement audio, annotations fortes, annotations faibles.

Les annotations faibles ont pour avantage de permettre la construction de jeux de données plus rapidement et à frais réduits. Il est alors possible d'utiliser le *crowdsourcing* [Fonseca et al., 2020], ou des systèmes pté-entraînés, pour automatiquement annoter des

jeux de données qui seraient autrement impossibles à labéliser à cause de leur taille [Gemmeke et al., 2017]. Dans ce dernier cas, l'intervention humaine est alors cantonnée à un rôle de vérification.

1.2.2 Méthodes

Les travaux de [Lecomte, 2013] s'intéressent à la détection d'évènements sonores audio, et explorent les différentes méthodes de Support Vector Machine (SVM) partiellement supervisées pour une application de surveillance des infrastructures publiques. L'idée est d'utiliser de multiples SVM mono-classes pour détecter les évènements anormaux et automatiquement les regrouper en clusters.

Pour une tâche de classification musicale faiblement supervisée, [Liu and Yang, 2016] proposent une approche d'entraînement multi-instance basée sur un CNN. L'idée principale est de fournir au modèle de multiples versions redimensionnées de l'exemple de départ, afin de localiser l'évènement. Ceci permet alors de faire des prédictions temporelles plus précises que la précision fournie par les données d'entraînement. Le CNN possède autant d'entrées que de dimensions et, après une série de convolutions indépendantes, les différentes branches du réseau se regroupent pour les convolutions finales. Enfin, deux sorties permettent de fournir, à la fois une classification globale, et une prédiction temporelle. La figure 1.10 représente l'architecture du CNN qu'ils ont utilisée dans leurs travaux.

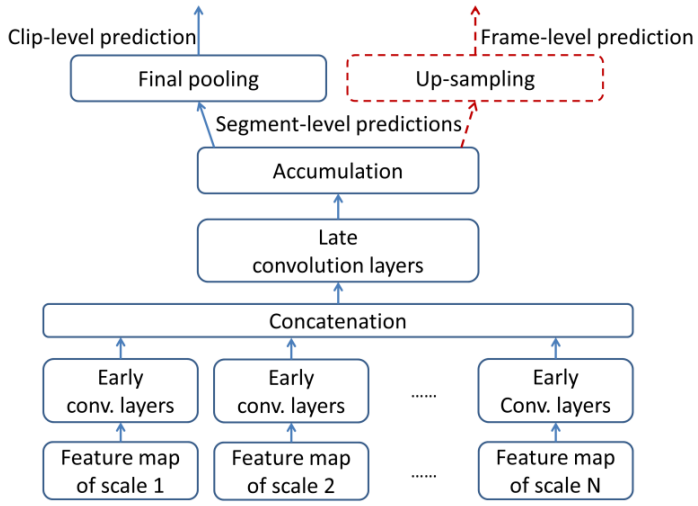

Figure 1.10 – L'architecture CNN proposée par Liu and Yang [2016]. Étant donné un réseau entraîné pour la prédiction au niveau du clip (en bleu), ils rajoutent une couche de sur-échantillonnage pour la prédiction au niveau de la trame temporelle (en rouge) permettant d'augmenter la précision.

[Kumar and Raj, 2016] et [Su et al., 2017] ont aussi exploré cette approche du redimensionnement des entrées. Dans leurs travaux, ils ajoutent également des mécanismes tels que l'augmentation de données et l'utilisation de filtres gaussiens spécialisés pour chaque classe.

En 2017, le défi du DCASE propose une tâche de détection d'évènements sonores en utilisant des labels faibles pour l'ensemble de données d'entraînement et des annotations fortes pour l'évaluation (tâche 4). L'apprentissage multi-instance présenté au-dessus est alors repris par Chou et al. [2017], et Lee et al. [2017a,b]. Pour prendre en compte l'information temporelle, une des approches est de découper le signal en multiples segments se superposant légèrement et faire une classification indépendante de ces derniers. Ces prédictions sont alors combinées pour fournir la prédiction temporelle sur l'ensemble de l'enregistrement audio.

Parmi les systèmes atteignant le top 10 du défi, [Adavanne and Virtanen, 2017] utilisent une approche différente. Ils utilisent une combinaison de couches convolutives suivies par une ou plusieurs couches récurrentes pour traiter la dimension temporelle de l'enregistrement audio de manière plus naturelle. Les annotations faibles sont alors considérées comme fortes pour l'intégralité de l'échantillon, c'est-à-dire qu'elles sont propagées à l'ensemble des trames temporelles.

Cette architecture combinant à la fois un réseau convolutif et un réseau récurrent sera appelé eConvolutional Recurrent Neural Network (CRNN). Nous utiliserons ce type d'architecture lors de nos deux participations à la tâche 4 du défi DCASE 2018 et 2019 (Chapitre2 et 3). Leur modèle est illustré dans la figure 1.11. C'est aussi une approche moins complexe à mettre en place, car elle ne nécessite pas de multiples versions de l'entrée ni de découpage en patchs qui devront être recombinés ultérieurement.

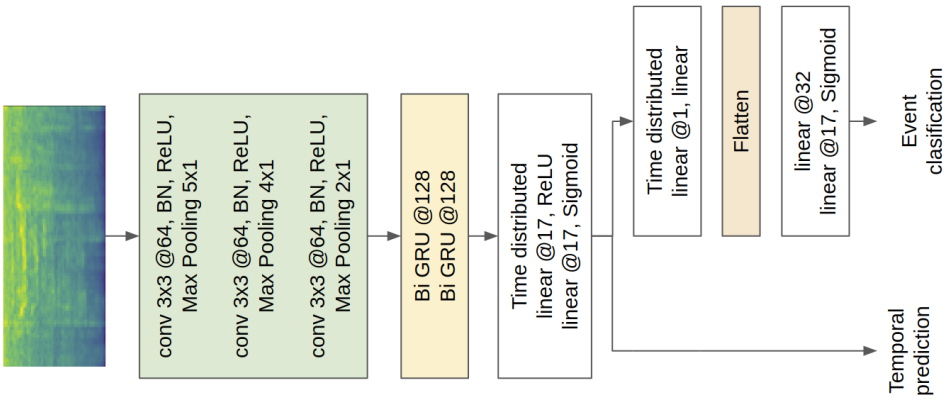

Figure 1.11 – Réseau de neurones convolutif et récurrent (CRNN) pour la prédiction d'annotations fortes à partir d'annotations faibles. Par Adavanne and Virtanen [2017]

Le modèle CRNN proposé se compose de trois couches de convolutions successives suivies de deux couches récurrentes bidirectionnelles ainsi qu'une couche linéaire appliquée à chaque trame sortie de la partie récurrente. La sortie de cette couche entièrement connectée est divisée en deux branches, une première est utilisée pour fournir directement les prédictions temporelles, tandis que la seconde permet de réaliser la tâche de classification globale.

Pour l'édition suivante du défi, le CRNN sera alors l'architecture de référence pour la plupart des systèmes proposés[JiaKai, 2018; Liu et al., 2018; Kothinti et al., 2018]. D'autres

proposeront alors de combiner les deux systèmes pour obtenir un CRNN utilisant de multiples entrées redimensionnées [Guo et al., 2018].

Lors des éditions 2019 et 2020 du défi, la tâche évolue pour intégrer des données synthétiques fortement annotées dans leur jeu de données Domestic Environment Sound Event Detection (DESED) [Turpault et al., 2019]. C'est une approche qui permet de générer des données réalistes, en grande quantité et sans intervention d'un annotateur. Combiné avec des annotations faibles pour l'apprentissage, il est ainsi possible d'améliorer les performances des systèmes de manière significative. Avec l'apparition de cet ensemble d'entraînement synthétique, l'utilisation d'un mécanisme d'attention se généralise, et une grande majorité des systèmes proposés utilisent des mécanismes d'attention [Lin and Wang, 2019; Delphin-Poulat and Plapous, 2019; Shi, 2019], en particulier le *context gating*.

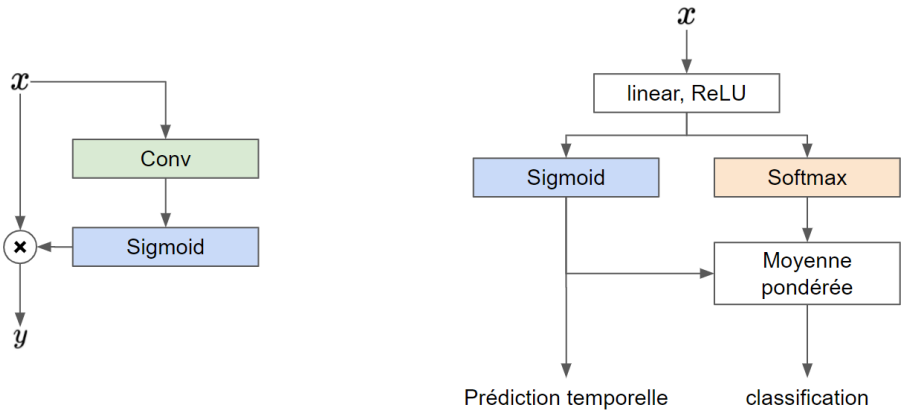

Figure 1.12 – Représentation du *context gating* à gauche, et à droite, une sortie avec un mécanisme d'attention.

Le *context gating* [Miech et al., 2018], illustré dans la partie gauche de l'image 1.12, est un mécanisme d'attention qui est réalisé au niveau des représentations en sortie des couches de convolutions, en multipliant terme à terme la sortie d'une couche de convolution suivie d'une activation sigmoide avec son entrée x. La sortie de la couche sigmoide produit des nombres entre 0 et 1 qui sont vus comme des poids. Une autre méthode, illustrée dans la partie droite de l'image s'effectue au niveau des décisions du modèle, et combine deux activations en sortie de la dernière couche linéaire du modèle. Il est ainsi possible d'obtenir une meilleure classification grâce à la combinaison de ces deux sorties.

En 2020, le meilleur système [Miyazaki et al., 2020] de la tâche 4 du défi, utilise des blocs de type « *conformer* », qui combinent des mécanismes d'auto attention (transformer) avec des réseaux de convolution, pour capturer le contexte global et temporel de l'enregistrement audio.

Pour conclure cet état de l'art sur la Sound Event Detection (SED) faiblement supervisée, il nous semble intéressant de regarder l'évolution des performances sur la tâche 4 de DCASE. Le tableau 1.1 montre cette évolution depuis l'édition 2018 jusqu'en 2020. En 2021, les

	F-score (%)	Méthode	Année
[Miyazaki et al., 2020]	51,1	Conformer + MT	DCASE 2020
[Hao et al., 2020]	50,7	CRNN	
[Ebbers and Haeb-Umbach, 2020]	47,8	CRNN + PL	
[Koh et al., 2020]	46,6	FP-CRNN + MT	
[Lin and Wang, 2019]	42,7	CNN + MT	DCASE 2019
[Delphin-Poulat and Plapous, 2019]	42,1	CRNN + MT	
[Shi, 2019]	42,0	CRNN + ICT	
[Cances et al., 2019]	39,7	CRNN	
[JiaKai, 2018]	32,4	CRNN + MT	DCASE 2018
[Liu et al., 2018]	29,9	CRNN	
[Kong et al., 2018]	24,0	CNN	
[Kothinti et al., 2018]	22,4	CRNN	

Table 1.1 – Évolution du F-score lors des éditions 2018 à 2020 du défi DCASE. Les quatre meilleurs scores de chaque année sont restitués.

organisateurs ont changé la métrique d'évaluation, qui n'est plus le F-score évènements, mais une nouvelle métrique indépendante des seuils de détection appelée *Polyphonic Sound Event Detection Scores* (PSDS) Bilen et al. [2020] [1].

Nous pouvons voir dans le tableau que chaque année, le meilleur système a un score environ 10 points supérieurs à celui de l'édition précédente. Les méthodes évoluent également chaque année pour incorporer de nouvelles techniques. Ainsi l'architecture CRNN + Mean-Teacher (MT) qui avait obtenu le meilleur résultat lors de l'édition 2018 est majoritaire lors de l'édition 2019. Une architecture de type Conformer a donné le meilleur score lors de l'édition 2020. En 2021, les meilleurs systèmes sont à nouveau des CRNN + MT. Le modèle conformer gagnant en 2020 ne semble pas avoir été encore adopté, peut-être à cause du fait que les auteurs n'ont pas rendu leur code public.

Il est important de noter que le jeu de données utilisé a aussi évolué. En 2019, un ensemble d'entraînements synthétiques fortement annotés est rajouté et en 2020, il était possible de générer soit même des données synthétiques, ce qui a sûrement contribué à l'amélioration des performances. En revanche, le jeu d'évaluation est le même sur les trois années.

1.3 Apprentissage profond semi-supervisé

L'apprentissage semi-supervisé (Semi Supervised Learning (SSL)) se différencie de l'apprentissage faiblement supervisé par l'absence pure et simple d'annotations pour une partie des exemples qui composent le jeu de données. On se rapproche plus d'une combinaison

1. http://dcase.community/challenge2021/task-sound-event-detection-and-separation-\in-domestic-environments

d'apprentissages supervisés et non-supervisés. Dans cette section, nous décrivons les différents mécanismes sous-jacents, visant à exploiter les données non-supervisées de manière efficace, avant de dresser un panorama des différentes méthodes de SSL, en particulier dans le cadre de l'apprentissage profond.

1.3.1 Les mécanismes de l'apprentissage semi-supervisé

On peut les regrouper dans trois catégories, à savoir, le maintien de la consistance, la minimisation d'entropie, et la régularisation. Ces mécanismes peuvent être combinés pour plus d'efficacité.

Maintien de la consistance

Soit x un exemple sans label et $\mathrm{Augment}(x)$, une version augmentée de lui-même. Le maintien de la consistance (*Consistency*) vise à minimiser la différence entre les prédictions du modèle sur x et sur $\mathrm{Augment}(x)$, à l'aide d'une pénalité fondée sur une distance euclidienne par exemple [Laine and Aila, 2017; Sajjadi et al., 2016] :

$$\text{Consistency} = \|P(y \mid x; \theta) - P(y \mid \mathrm{Augment}(x); \theta)\|_2^2. \tag{1.16}$$

Un inconvénient de cette approche est l'utilisation de stratégies d'augmentation de données spécifiques au domaine d'application. Il est possible que certaines méthodes d'augmentation fonctionnent pour une tâche ou un jeu de données spécifiques mais pas pour d'autres. Le *Virtual Adversarial Training* Virtual Adversarial Training (VAT) [Miyato et al., 2018] répond à cet inconvénient, et aborde ce problème en calculant plutôt une perturbation additive à appliquer à l'entrée, qui tente de modifier la distribution des prédictions faites par le modèle.

Une autre méthode de maintien de la consistance est utilisée par Verma et al. [2020]. Il s'agit de la Interpolation Consistency Training (ICT) qui encourage la distribution des prédictions des fichiers non-annotés à être proche de la distribution des prédictions des fichiers annotés.

Minimisation de l'entropie

Une hypothèse sous-jacente commune à de nombreuses méthodes d'apprentissage semi-supervisé est que le classifieur ne doit pas prendre de décision ayant une forte entropie, c'est-à-dire avec une confiance faible. La figure 1.13 illustre une prédiction avec une entropie forte (à gauche, où les prédictions pour les différentes classes sont toutes proches les unes des autres), et une prédiction avec une entropie faible (à droite, où la prédiction d'une classe se démarque bien des autres). Une façon de faire respecter cette hypothèse est de

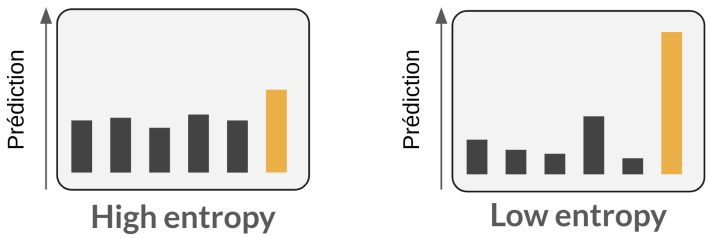

Figure 1.13 – À gauche : un exemple de prédiction à forte entropie, les valeurs des prédictions sont proches les unes des autres. À droite : Un exemple de prédiction à faible entropie.

forcer le système à produire des prédictions à faible entropie sur les données non-étiquetées. Cela peut être fait de manière explicite comme dans [Grandvalet and Bengio, 2005] avec un terme de perte qui minimise l'entropie de $P(y \mid x; \theta)$ pour les données non-étiquetées x. Cette forme de minimisation de l'entropie a été combinée avec la VAT dans [Miyato et al., 2018] pour obtenir des résultats plus robustes. L'utilisation de labels artificiels, ou « pseudo-labels » [Lee, 2013] en anglais, permet de favoriser la minimisation d'entropie implicitement en créant des étiquettes artificielles à partir de prédictions de haute confiance sur des données non-étiquetées et en les utilisant comme cibles d'entraînement. MixMatch (MM) [Berthelot et al., 2019] parviennent également à minimiser l'entropie de manière explicite grâce à l'utilisation d'une fonction d'« affinage » ($\mathrm{Sharpen}$) de la distribution cible pour les données non-étiquetées, décrites dans la section 5.1.3.

Régularisation traditionnelle

Les mécanismes classiques de régularisation consistent à imposer une contrainte à un modèle pour qu'il ait plus de mal à apprendre par coeur les données d'apprentissage et, par conséquent, pour qu'il puisse mieux généraliser sur de nouvelles données [Hinton and van Camp, 1993]. Nous pouvons utiliser la décroissance de poids (*weight decay*) qui pénalise la norme ℓ_2 des paramètres du modèle [Loshchilov and Hutter, 2019; Zhang et al., 2018a]. Mixup [Zhang et al., 2018b] dans MM est aussi utilisé pour la création de nouveaux exemples à partir de paires d'enregistrements que l'on mélange entre eux. Mixup est utilisé, à la fois comme un régularisateur (appliqué aux données étiquetées), et comme une méthode d'apprentissage semi-supervisée (appliquée aux données non-étiquetées).

1.3.2 Méthodes

Le SSL intervient lorsque seulement une fraction du jeu de données utilisée pour l'entraînement a été annotée. Les méthodes SSL tentent d'exploiter également les exemples non-annotés lors de l'entraînement. Ces derniers peuvent aider à régulariser le modèle et ainsi améliorer les résultats du système pour la tâche à résoudre.

Au fil des années, de nombreuses méthodes ont été développées pour pourvoir utiliser les exemples non-annotés. On peut ainsi parler de la génération de « pseudo » étiquettes, l'utilisation de tâches secondaires ayant des objectifs bien précis tels que la minimisation d'entropie, le maintien de la consistance, la corruption ou encore l'apprentissage auto-supervisé. Plus récemment sont apparues les méthodes dites « Holistiques ».

[Lee, 2013] et ses collègues proposent d'utiliser le *pseudo-labelling* dans une tâche de classification avec un DNN. Le pseudo-labelling consiste en l'utilisation d'un modèle déjà entraîné pour générer des annotations pour les données qui n'en possèdent pas. De cette façon, il est possible d'agrandir le jeu de données d'entraînement pour réentraîner un modèle. [Nguyen et al., 2018; Dorfer and Widmer, 2018] utilisent également le *pseudo labelling* pour automatiquement générer des labels sur les exemples non-annotés au cours de l'entraînement, et ils ne gardent que les labels qui sont prédits avec une haute confiance. Les prédictions sont faites à chaque itération de l'entraînement et cela permet de s'affranchir de l'aspect itératif de l'algorithme.

[Valpola, 2015] propose un réseau « en échelle » (ou *Ladder Network*) pour rendre les réseaux de neurones plus robustes au bruit. Ce modèle a été rapidement repris par [Rasmus et al., 2015] pour proposer le Γ-modèle et utiliser le débruitage comme une fonction secondaire pour intégrer les données non-annotées lors de l'entraînement. Le Γ-modèle consiste en deux auto-encodeurs identiques dont l'un est « corrompu » , c'est-à-dire qu'un bruit Gaussien est appliqué sur les poids de chacune des couches de l'encodeur. Chaque encodeur a pour objectif secondaire de produire des prédictions cohérentes pour une même entrée x. La fonction de coût non-supervisée $\mathcal{L}_u$ est calculée comme la Mean Square Error (MSE) entre la sortie des deux modèles.

$$\mathrm{MSE}(a, b) = \|a - b\|_2^2 \tag{1.17}$$

$$\mathcal{L}_u = \mathrm{MSE}\left(P(y \mid x; \theta), P(y \mid x; \hat{\theta})\right) \tag{1.18}$$

Quelque temps après, [Laine and Aila, 2017] proposent le Π-modèle comme étant une simplification du Γ-modèle en supprimant la partie « corrompue ». Le bruitage est finalement réalisé par l'utilisation de *dropout* et de l'augmentation de données lors d'une deuxième passe. La fonction de coût non-supervisée $\mathcal{L}_u$ devient alors la MSE entre la première prédiction sur l'exemple original $y = P(y \mid x; \theta)$, et la prédiction faite avec le *dropout* et/ou l'augmentation de donnée $\hat{y} = P(y \mid x', \theta')$.

$$\mathcal{L}_u = \mathrm{MSE}(y, \hat{y}) \tag{1.19}$$

Toujours avec l'objectif d'améliorer leur système, [Laine and Aila, 2017] proposent le *Temporal ensembling*. Au lieu d'utiliser la prédiction du modèle sur les données corrompues comme

pour le Γ-modèle, le *temporal ensembling* agrège toutes les prédictions précédentes, en utilisant la Exponential Moving Average (EMA), et les considère comme la vérité terrain de la fonction de coût non-supervisée. Cela permet de prendre en compte aussi les irrégularités lors de l'entraînement du modèle. Chaque prédiction y_i est alors accumulée dans un ensemble Y en utilisant la EMA. La fonction de coût non-supervisée $\mathcal{L}_u$ utilise Y alors comme vérité terrain.

$$Y = \alpha Y + (1 - \alpha) y_i \tag{1.20}$$

$$\mathcal{L}_u = \mathrm{MSE}(y, Y) \tag{1.21}$$

Dans cette configuration, si le poids du coût supervisé est supérieur à celui du coût non-supervisé, le système n'apprend pas de nouvelles informations, prédit la même cible pour une entrée donnée et tombe dans une boucle de biais de confirmation.

Comme solution à ce problème, [Tarvainen and Valpola, 2017] proposent d'isoler la prédiction des données corrompues dans un modèle « professeur » MT. Un modèle « étudiant » est entraîné pour minimiser la perte supervisée sur les exemples étiquetés et la perte de maintien de la cohérence sur les exemples non-étiquetés. Les poids du modèle « professeur » sont mis à jour grâce à une EMA des poids du modèle « étudiant » à chaque itération de l'entraînement. La fonction de coût non-supervisée compare les prédictions du modèle étudiant y_s avec celle du modèle professeur y_t. Nous pouvons trouver des applications du MT dans le défi de la tâche 4 du DCASE depuis 2018 dans les systèmes de détection d'évènements sonores faiblement supervisés (voir chapitre 2). Les gagnants de 2018 ont entraîné des CNN sur un petit sous-ensemble étiqueté et un plus grand sous-ensemble non-étiqueté [JiaKai, 2018]. Ils ont utilisé une approche MT, dans laquelle deux modèles sont construits de manière à les rendre complémentaires et plus robustes. Dans les éditions suivantes, une grande majorité des systèmes utilisent alors le MT [Shi, 2019; Delphin-Poulat and Plapous, 2019; Miyazaki et al., 2020; Yao et al., 2020]. Une description complète du MT est disponible à la section 5.1.3 car il s'agit d'un système que j'ai utilisé dans cette thèse.

Une approche nommée Deep Co-Training (DCT) [Qiao et al., 2018] utilise elle aussi deux modèles entraînés en parallèle ainsi que le VAT, et permet d'obtenir de très bons résultats pour la tâche de classification d'images. À notre connaissance, il n'y pas d'application de DCT pour une tâche de classification d'évènements sonores. De plus, elle a été proposée au début de ma thèse et semblait être la meilleure méthode en détection d'objets dans les images. C'est pourquoi nous l'avons choisi comme méthode centrale de ce travail de thèse. Vous retrouverez sa description complète dans le chapitre 4, ainsi que toutes les expériences menées sur la classification d'évènements sonores (*audio tagging*) dans les chapitres ultérieurs.

En 2019, les méthodes appelées « holistiques » apparaissent. Ces approches ont pour objectif de simplifier l'apprentissage semi-supervisé en combinant les différents mécanismes d'apprentissage SSL décrits plus haut. Comme amélioration notable, on notera l'utilisation d'un unique réseau et l'utilisation extensive de l'augmentation comme mécanisme de régularisation. Ainsi, [Berthelot et al., 2019] proposent l'approche MixMatch MM qui combine le mécanisme de minimisation d'entropie, le *pseudo labelling* et l'augmentation de données pour exploiter des données non-étiquetées et fournir une meilleure capacité de généralisation. L'année suivante, les mêmes auteurs [Berthelot et al., 2020] proposent une extension à MM nommée ReMixMatch (RMM). Ils rajoutent des augmentations plus fortes, un mécanisme supplémentaire de minimisation d'entropie et une branche pour l'auto-apprentissage. Cette approche augmente en complexité mais gagne apparemment en performance. La même année, FixMatch (FM) [Sohn et al., 2020], une simplification de RMM est aussi proposée, supprimant l'un des mécanismes de minimisation d'entropie maintenant redondant dans MM ainsi que la branche d'auto-apprentissage du RMM. Les méthodes MM, RMM et FM sont toutes détaillées dans le chapitre 5 de cette thèse.

Méthode	CIFAR-10	Modèle	Paramètres (Million)
Supervisé augmenté (2020)	12,74 $\pm$ 0,29	Wide ResNet28-2	1,5
pseudo labelling (2013)	16,09 $\pm$ 0,28	Wide ResNet 28-2	1,5
Γ-model (2015)	20,40 $\pm$ 0,47	ConvNet-13	3,1
Πmodel (2017)	14,01 $\pm$ 0,38	Wide ResNet 28-2	1,5
Temporal Ensembling (2017)	12,16 $\pm$ 0,24	ConvNet-13	3,1
Mean-Teacher (2017)	9,19 $\pm$ 0,19	Wide ResNet 28-2	1,5
Deep Co-Training (2018)	8,35 $\pm$ 0,06	ConvNet-13	3,1
MixMatch (2019)	6,42 $\pm$ 0,10	Wide ResNet 28-2	1,5
ReMixMatch (2020)	4,72 $\pm$ 0,13	Wide ResNet 28-2	1,5
FixMatch (2020)	4,26 $\pm$ 0,65	Wide ResNet 28-2	1,5

Table 1.2 – Résultats retranscrits à partir des résultats des articles de [Rasmus et al., 2015; Qiao et al., 2018; Sohn et al., 2020; Berthelot et al., 2019, 2020].

Toutes ces méthodes ont été testées et validées pour des tâches de classification d'images en utilisant les jeux de données dits de *benchmark*, comme : Canadian Institute For Advanced Research - 10 (CIFAR-10) [Ho-Phuoc, 2019], The Street View House Numbers (SVHN) [Goodfellow et al., 2014] ou ImageNet [Deng et al., 2009]. Dans le tableau 1.2, j'ai synthétisé l'évolution chronologique des performances des méthodes SSL sur le jeu de données CIFAR-10 lorsque seulement 4000 exemples annotés sont utilisés, les autres (36 000 exemples) étant utilisés sans leur label. On peut voir que depuis le *pseudo-labelling* en 2003, les méthodes ont évolué pour gagner en performance. La méthode MT, résultat de plusieurs évolutions depuis le Γ-model, permet de gagner 6,9 points (42,88 % relatif) comparée au *pseudo-labelling*. FM, la dernière méthode en date permet de gagner 11,83 points (73,52 % relatif) comparé

au pseudo-labbeling, ou 4,93 (53,64 % relatif) comparée à MT. De plus, les méthodes MT et suivantes donnent de meilleurs résultats qu'un apprentissage supervisé n'utilisant que les mêmes 4000 images, avec augmentation (ligne "Supervisé augmenté") dans le tableau.

Il existe une littérature beaucoup plus large sur les approches SSL que je n'ai pas explorée durant ma thèse et donc que je n'aborderais pas ici. Sans être exhaustif, on peut trouver des méthodes basées sur les graphes [Zhu et al., 2003; Bengio et al., 2006; Liu et al., 2019], les systèmes génératifs [Belkin and Niyogi, 2001; Hinton and Salakhutdinov, 2008; Odena, 2016] ainsi que les modèles transductifs [Gammerman et al., 1998; Joachims, 1999, 2003].

Les branches de l'apprentissage faiblement supervisé et semi-supervisé sont en plein essor, avec de nouvelles méthodes chaque année repoussant l'état de l'art de plus en plus loin. Certaines permettent même d'atteindre les mêmes performances que celles d'une approche entièrement supervisée, en utilisant seulement une fraction des annotations fournies. L'auto-apprentissage (*self-supervised learning*) devient aussi une approche de prédilection pour pré-entraîner des modèles qui peuvent ensuite être affinés (*fine-tuning*) sur une tâche et des données autres que celles utilisées lors du pré-entraînement. La discipline semble s'orienter donc vers l'apprentissage de représentation.

Dans ce chapitre, nous avons brièvement introduit quelques notions en apprentissage profond, sur les réseaux convolutifs, récurrents et résiduels, que j'ai utilisés au cours de la thèse. Nous avons ensuite essayé de dresser un état de l'art pour la tâche de détection d'évènements sonores (SED) à partir d'annotations faibles, introduisant la différence entre annotations fortes et faibles ainsi que l'évolution des méthodes depuis 2013. Mes premiers travaux sur ce sujet composeront la première partie de ma thèse. Pour finir, nous avons décrit les mécanismes principaux de l'apprentissage (profond) semi-supervisé et avons dressé un état de l'art des méthodes appliquées à l'image. Il existe peu d'approches semi-supervisées actuellement mises en oeuvre pour l'audio tagging. Il s'agira du sujet principal de la seconde partie de cette thèse.

Première partie

Détection d'évènements sonores faiblement supervisée

Chapitre 2

Détection d'évènements sonores faiblement supervisée

Dans ce chapitre, je présente les premiers travaux de ma thèse sur l'apprentissage faiblement supervisé lors d'une tâche de classification et de segmentation d'évènements sonores. Ces travaux ont été réalisés dans le cadre du défi international Detection and Classification of Acoustic Scenes and Events (DCASE). La particularité de ce défi vient de l'utilisation d'annotations dites « faibles » qui informent de la présence d'évènements dans un enregistrement audio, sans fournir leurs localisations dans le temps. L'objectif est double : en utilisant seulement ces annotations « faibles », il faut détecter les évènements audio (*audio tagging*) au niveau de l'enregistrement, et les localiser précisément en fournissant des informations sur le début et la fin de chacun d'entre eux. Pour répondre à cette problématique, j'ai travaillé sur une première solution, consistant en la modification des couches récurrentes Gated Recurrent Unit (GRU) dans un Convolutional Recurrent Neural Network (CRNN) pour introduire un mécanisme d'oubli lors de l'utilisation du réseau en inférence. Nous avons appelé cette variante Weighted Gated Recurrent Unit (WGRU). Je décrirai les résultats obtenus au challenge DCASE 2018 (Tâche 4). Je mentionnerai également une seconde approche mise en oeuvre par mon directeur de thèse Thomas Pellegrini, qui a donné de meilleurs résultats que la variante WGRU. Inspirée des travaux de [Morfi and Stowell, 2018], cette méthode utilise une fonction de coût multi-objectif appelée Min Mean Max (MMM). Il faut mentionner d'ores et déjà que l'approche WGRU n'a pas démontré de capacité à bien généraliser sur le jeu de données d'évaluation du challenge.

Comme décrit en introduction, la tâche combinée de classification et de localisation d'évènements sonores est appelée en anglais Sound Event Detection (SED). Cette dernière a fait l'objet de recherches approfondies [Virtanen et al., 2018] et couvre un large éventail d'applications comme par exemple en écologie (bio-acoustique) [Stowell et al., 2016] et en surveillance [Crocco et al., 2016]. La détection de sons dits domestiques peut fournir des indices intéressants pour des applications liées à la santé [Guyot et al., 2013], mais aussi pour les assistants virtuels intelligents de plus en présents dans notre quotidien.

Initialement, la recherche autour de la classification d'évènements était plus concentrée sur la SED monophonique. On retrouve dans la littérature des approches supervisées utilisant des techniques empruntées du traitement de la parole et de la musique avec les Gaussian Mixture Model (GMM), les Hidden Markov Model (HMM) [Mesaros et al., 2010]. L'objectif était de modéliser chaque évènement individuel avec les GMM, et de détecter chaque occurrence grâce aux HMM. Avec l'apparition de jeux de données publics plus proches de la réalité (« authentiques »), la tâche est rapidement devenue polyphonique et donc plus complexe.

La question de la détection d'évènements audio polyphoniques à l'aide de données faiblement étiquetées a été abordée dans le défi DCASE depuis 2018. L'utilisation d'annotations faibles induit une nouvelle difficulté consistant à entraîner un réseau à localiser un ou plusieurs évènements dans le temps, sans lui fournir d'annotations temporelles (annotations fortes) lors de l'apprentissage. Aujourd'hui, les CRNN sont toujours la référence pour la SED polyphonique faiblement supervisée [Kim and Kim, 2020; Ebbers and Haeb-Umbach, 2020; Zheng et al., 2021], ce qui change est l'algorithme d'entraînement ainsi que les différents mécanismes permettant d'extraire cette information temporelle manquante. Dans les travaux de [Kim and Kim, 2020], trois fonctions de coût spécialisées sont utilisées, une première pour les exemples fortement annotés s'il y en a, une seconde pour ceux faiblement annotés, et finalement une dernière pour ceux sans annotation aucune.

Dans ce chapitre, j'introduis la tâche de détection d'évènements sonores polyphoniques et compare le comportement d'un réseau récurrent lorsqu'il est entraîné à partir d'annotations fortes, puis faibles. Je parle ensuite des modifications apportées à la cellule récurrente GRU (1.1.3) pour introduire un mécanisme d'oubli et pourquoi ce dernier est important lorsqu'on utilise des annotations faibles pour l'entraînement. Finalement, je compare les résultats obtenus entre plusieurs approches et les nôtres lors de la participation à la tâche 4 du défi DCASE 2018.

2.1 Systèmes de Détection d'Évènements Sonores (SED) Polyphoniques

Une illustration générale d'un système SED est présentée dans la figure 2.1. Le système vise à prédire les classes présentes dans l'enregistrement ainsi que les prédictions temporelles correspondantes à ces classes.

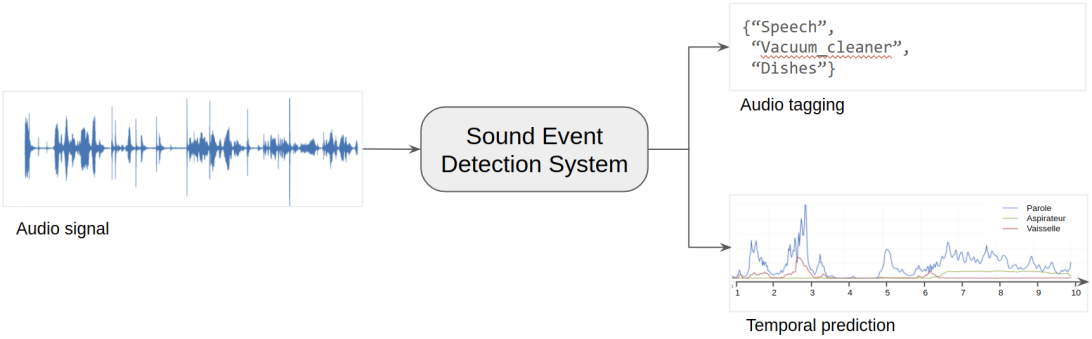

Figure 2.1 – Une représentation des deux sorties attendues d'un système de SED : d'une part des tags, d'autre part les prédictions temporelles des évènements.

Pour remplir les deux tâches, un système fait donc intervenir soit deux modèles différents, soit un seul avec deux sorties (deux couches de sortie) différentes. Le premier composant identifie les tags et le second localise ces évènements dans l'enregistrement. Les deux composants du système peuvent être exécutés simultanément, ou l'un après l'autre, en sélectionnant les prédictions temporelles uniquement des évènements qui ont été prédits.

Les implémentations les plus courantes parmi les participants utilisent un unique modèle, sous la forme d'un CRNN, qui fait les deux tâches. L'hypothèse sous-jacente de ces architectures, qui combinent couches de convolution et couches récurrentes, est que l'information temporelle requise pour localiser les évènements est toujours présente dans la dernière couche dense du modèle. On peut alors, en outre-passant la couche de classification et, en allant récupérer directement la sortie de la couche dense, obtenir les prédictions temporelles du système. Plus de détails de cette architecture sont fournis dans la section 2.4.3 de ce chapitre.

Les prédictions temporelles prennent alors la forme de courbes évoluant entre 0 (absent) et 1 (présent) pendant toute la durée de l'enregistrement. Il y a autant de courbes que de catégories à classifier. Cependant, grâce au classificateur, on peut filtrer ces courbes pour ne garder que celles qui sont pertinentes. L'image 2.2 est une illustration de courbes réelles, obtenues à la sortie de cette couche dense.

Les Recurrent Neural Network (RNN) et CRNN se sont montrés très efficaces lorsqu'ils sont entraînés avec des annotations « fortes », mais est-ce le cas avec des annotations « faibles » ? Pour pouvoir répondre à cette question, il est nécessaire de comprendre le

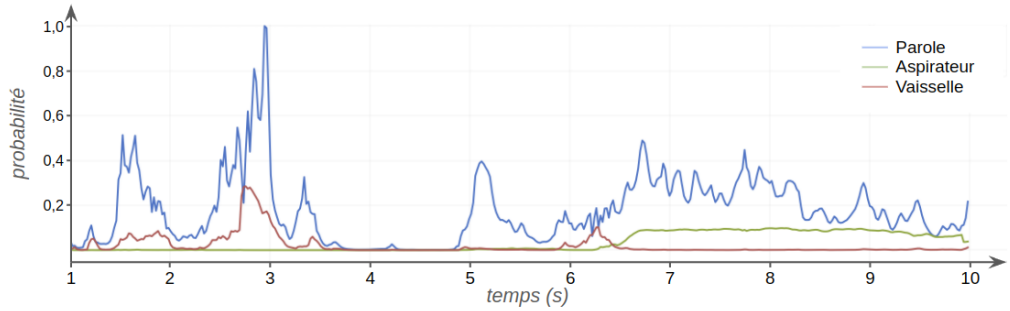

Figure 2.2 – Courbes de prédictions temporelles d'un enregistrement audio contenant trois catégories, *Speech*, *Vacumm cleaner*, *Dishes*

fonctionnement des réseaux récurrents ainsi que l'implication d'avoir des labels « faibles ». Dans un premier temps, on regardera le comportement d'un réseau récurrent entraîné sur des labels « forts ». Puis on parlera de l'entraînement avec des annotations « faibles ».

2.1.1 L'entraînement avec des annotations fortes

Comme mentionné dans la section 1.1.3, les réseaux récurrents sont particulièrement efficaces pour traiter des données ayant une cohérence temporelle. Pour cela ils sont très souvent utilisés dans des systèmes de transcription de la parole, de classification musicale, ou encore de détection d'évènements sonores [Cakir et al., 2017].

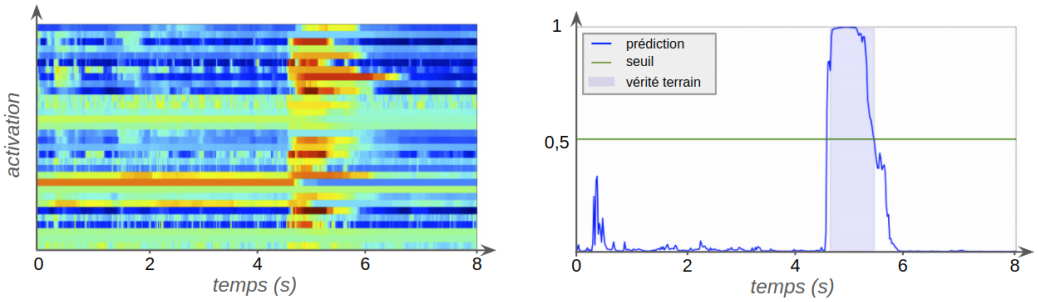

Figure 2.3 – Visualisation de la prédiction d'un réseau récurrent entraîné avec des annotations fortes.

La figure 2.3, extraite de l'article [Cakır and Virtanen, 2019], montre clairement l'efficacité des réseaux récurrents pour la localisation d'évènements sonores à partir d'annotations « fortes ». La prédiction temporelle illustrée ici est presque parfaite. Avant et après l'évènement, les prédictions sont proches de 0, tandis que pendant l'évènement, elles sont proches de 1.

2.1.2 L'entraînement avec des annotations faibles

Dans le cadre d'un entraînement faiblement supervisé, les labels indiquent seulement si l'évènement se trouve dans la séquence. Cela est suffisant pour une tâche de classification mais rend la localisation difficile. Si on utilise les tags seulement lors de l'entraînement, le

réseau récurrent s'active à partir du moment où il rencontrera la première occurrence de l'évènement, et gardera une activité élevée ensuite. Si l'évènement continue jusqu'à la fin de l'enregistrement audio, cela n'est pas problématique, car on a bien trouvé le début de l'évènement et sa fin. Mais cela n'est pas vrai pour de nombreuses catégories d'évènements. Il y a une pause significative entre chaque aboiement d'un chien par exemple, ou une personne peut s'arrêter de parler pendant plusieurs secondes entre deux phrases. Que se passe-t-il dans ce cas ?

L'image 2.4 illustre très bien ce comportement, avec la prédictions temporelle obtenue avec un CRNN entraîné avec des labels faibles.

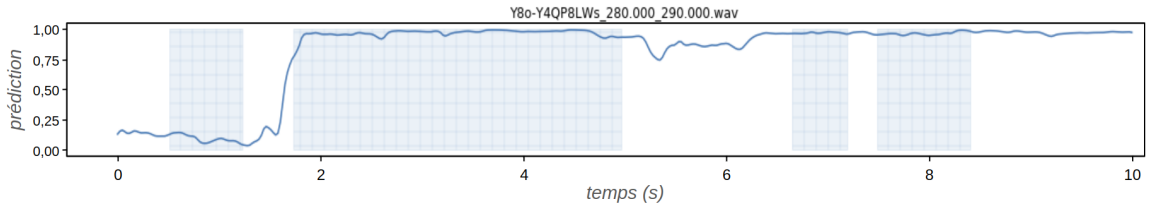

Figure 2.4 – Visualisation de la prédiction d'un réseau récurrent GRU entraîné avec des annotations faibles.

Par souci de clarté, une seule courbe est représentée. Les blocs verticaux correspondent à la vérité terrain. La prédiction passe de 0 a 1 très rapidement au moment de la seconde occurrence de l'évènement. Après quelques secondes, l'évènement s'arrête pendant une seconde, mais la prédiction ne chute pas. La même chose se produit avec les deux derniers évènements. À partir du moment où le système a détecté une classe, il continuera à la prédire.

Au regard de ce comportement, la question à se poser maintenant est, comment faire pour que le modèle ne maintienne pas sa prédiction après la détection de la première occurrence de l'évènement ? Peut-on forcer le modèle à « oublier » une partie du contexte passé et ainsi ne plus prédire l'évènement lorsque ce dernier est fini ? Ou bien, peut-on appliquer des contraintes lors de l'entraînement pour indiquer que le l'évènement n'est pas constant, forçant le modèle à ne pas le prédire pendant toute la durée de l'enregistrement ?

Nous allons tenter de répondre à ces questions grâce à la modification d'une cellule récurrente, que j'ai nommée WGRU, et l'utilisation d'une autre méthode d'apprentissage, le MMM, proposée en 2018 pour la localisation d'oiseaux dans [Morfi and Stowell, 2018].

2.2 Réseau récurrent pondéré à l'inférence WGRU

Les RNN se sont avérés efficaces pour modéliser les évènements sonores car ceux-ci ont souvent une structure séquentielle sous-jacente [Parascandolo et al., 2016a]. Cependant, certaines classes d'évènements sonores ont des durées typiques différentes, comme décrit dans [Serizel et al., 2018]. Les sons de longue durée, comme par exemple les sons d'aspirateur

(*vacuum cleaner*), d'eau courante (*running water*) et de mixeur (*blender*), devraient être plus faciles à reconnaître par un RNN simple que les sons « courts » comme des aboiements (*dog*), des miaulements (*cat*) ou de la parole (*speech*).

2.2.1 Introduction d'un mécanisme d'oubli à l'inférence

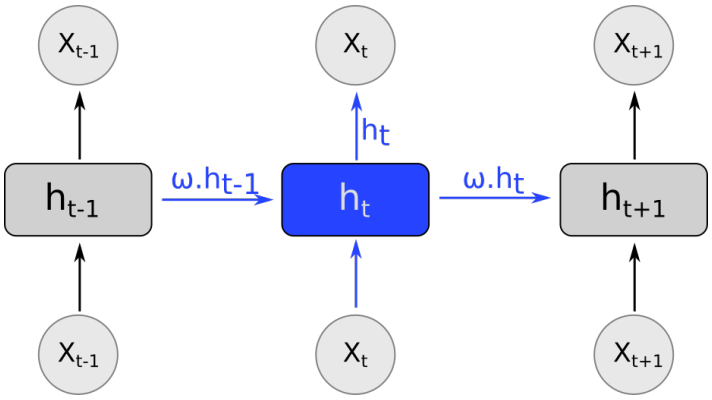

Figure 2.5 – Introduction d'un poids ω sur l'état caché d'une cellule récurrente WGRU.

Afin d'ajuster un modèle à différents types de sons, nous avons imaginé une nouvelle adaptation de RNN. Cette approche vise à configurer différents types de comportement temporel en fonction de la classe de l'évènement sonore à reconnaître. Comme première tentative dans cette direction, nous avons essayé de pondérer l'influence de l'état caché des cellules récurrentes par un facteur $\omega \leq 1$, que nous fixons globalement pour les classes sur un sous-ensemble de développement. Ceci modifie l'impact de la temporalité dans la couche récurrente GRU, tel que formulé dans l'équation 2.1 et représenté par la figure 2.5.

$$h_t = \boldsymbol{\omega} h_{t-1} \tag{2.1}$$

2.2.2 Comment fixer le poids ω ?

Le poids ω permet d'ajuster la force du mécanisme d'oubli, c'est-à-dire la quantité d'information du contexte passé que le modèle garde en mémoire pour fournir une prédiction. Mais comment déterminer la valeur de ω ? Avant toutes choses, il est important de spécifier qu'il n'est pas nécessaire d'entraîner à nouveau le modèle. Ce mécanisme n'est utilisé que lors de la phase d'inférence, il suffit de remplacer, tout en gardant ses poids, la couche GRU du modèle déjà entraînée par une WGRU au moment de l'inférence sur les exemples de validation ou d'évaluation. Ainsi, en effectuant deux passes successives, une première avec le GRU, puis une seconde avec le WGRU, il est possible d'effectuer la classification avec le premier et la localisation des évènements sonores avec le deuxième.

Un poids de $\omega = 1$ correspond à un GRU standard. Un poids plus faible est censé être plus adapté aux évènements de courte durée. Il est entre outre possible d'effectuer plusieurs

passes de localisation en utilisant différentes valeurs de ω. Nous pouvons alors combiner les résultats obtenus pour éventuellement améliorer les prédictions temporelles.

2.2.3 Prédictions temporelles avec le WGRU

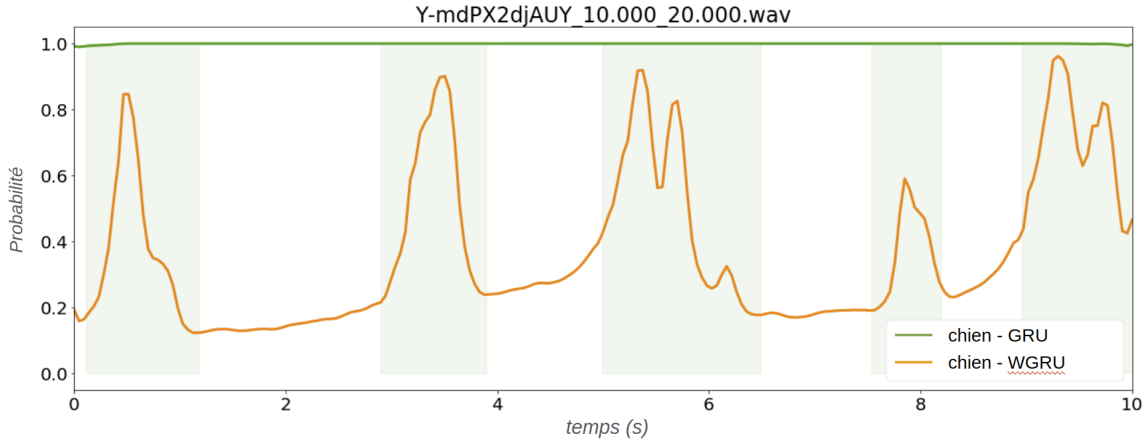

Figure 2.6 – Prédictions d'un CRNN utilisant une couche GRU classique en vert et une couche WGRU en orange ($\omega = 0,25$). Les rectangles verticaux indiquent la vérité du terrain et représentent les segments où *Dog* devrait être détecté.

La figure 2.6 donne un exemple dans lequel le CRNN standard (utilisant des cellules récurrentes GRU) ne permet pas de localiser l'évènement sonore *Dog* qui, pourtant, a été correctement classifié pour cet enregistrement particulier. Lorsqu'on replace, lors de l'inférence, la couche GRU par une WGRU où $\omega = 0,25$, le modèle devient alors capable de localiser avec une certaine précision les cinq occurrences de l'évènement présent dans l'exemple. On peut voir l'effet du mécanisme d'oubli qui se manifeste entre chaque occurrence de l'évènement.

2.3 Fonction de coût Min Mean Max

[Morfi and Stowell, 2018] a proposé une fonction de coût inspirée du paradigme Multiple Instance Learning (MIL) [Dietterich et al., 1997], définie dans l'équation 2.2, ici pour un seul exemple d'apprentissage pour simplifier la notation. BCE désigne la cross-entropie binaire, p les prédictions temporelles du modèle MMM, y la vérité terrain faible, et les indices k et j correspondent aux indices sur les K classes et sur les trames acoustiques respectivement.

L'idée est de forcer les prédictions à tendre vers 1 pour les trames avec les valeurs de score les plus élevées (premier terme), ou vers 0 pour les trames avec les scores les plus faibles (troisième terme). Le deuxième terme utilise la moyenne des scores des trames et tend à considérer qu'un évènement spécifique sera présent dans la moitié des trames acoustiques en général. Ils ont appliqué cette idée avec succès à un problème de classification binaire

pour détecter la présence ou l'absence d'oiseaux dans des enregistrements audio. Nous avons généralisé la fonction de perte binaire à une tâche de classification multiclasse en sommant les contributions de chaque classe.

$$\mathcal{L}_{\text{MMM}} = \sum_{k=1}^{K} \text{BCE}(y_k, \max_j \hat{y}_{kj}) + \text{BCE}(y_k/2, \text{mean}_j \hat{y}_{kj}) + \text{BCE}(0, \min_j \hat{y}_{kj}) \tag{2.2}$$

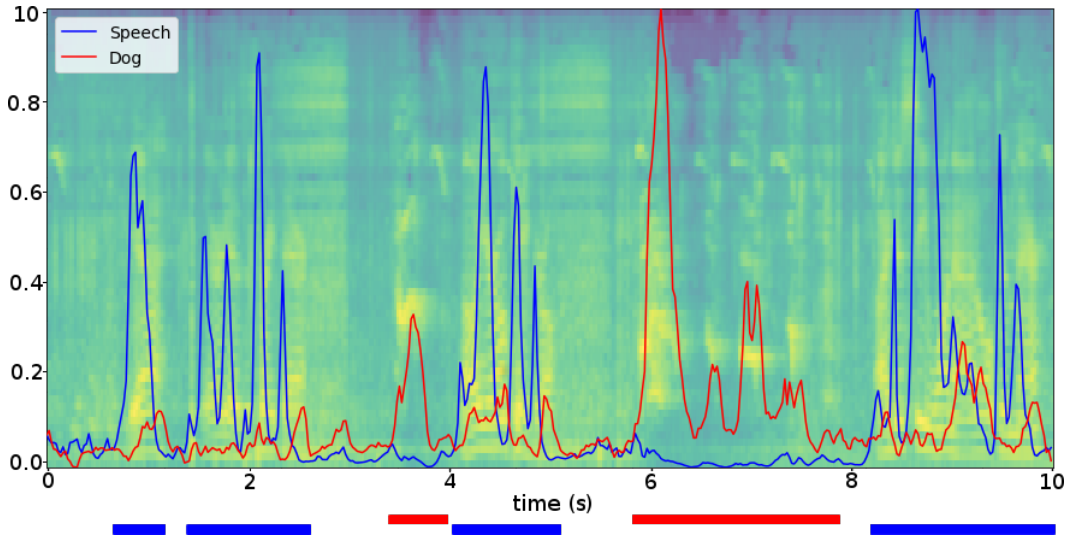

Figure 2.7 – Courbes de scores obtenu avec un modèle MMM pour deux classes correctement détectées *Speech* (bleu) et *Dog* (rouge). En dessous du spectrogramme est représentée la vérité terrain avec les mêmes couleurs.

La figure 2.7 montre un exemple de prédictions faites avec un modèle MMM, pour un exemple de test qui contient de la parole (*Speech*) et des aboiements de chien (*Dog*) dans les segments donnés en couleur sous le spectrogramme. Le premier Convolutional Neural Network (CNN) de classification a correctement identifié ces deux classes au niveau de l'exemple, et le second CRNN, entraîné avec la fonction MMM a donné les deux courbes bleu pour *Speech*, et rouge pour *Dog*. On observe bien la présence de pics sur les bons segments pour les deux classes. Il convient ensuite de seuiller et lisser ces pics pour obtenir une segmentation binaire. L'optimisation de ces seuils sera l'objet du prochain chapitre.

2.4 Expérimentations dans le cadre du défi DCASE 2018

La tâche 4 du défi DCASE est une tâche de détection d'évènements sonores enregistrés dans un environnement domestique [Serizel et al., 2018]. Elle met en jeu dix classes d'évènements différents. L'objectif est de fournir des temps de début et de fin des évènements

(étiquettes fortes), alors que l'ensemble d'entraînement repose uniquement sur des annotations « faibles ». Comme mentionné dans [Serizel et al., 2018], la durée des sons cibles dépend fortement de leur classe. Par exemple, la classe *vacuum cleaner* (aspirateur) contient principalement des évènements d'une durée de 10 secondes, tandis que la classe *dog* est principalement composée de plusieurs occurrences de moins d'une demi-seconde dans un même enregistrement. Cette information est importante pour la suite de ce chapitre.

La méthode de base fournie par les organisateurs du défi reposait sur deux CRNN : le premier est utilisé pour la classification des évènements présents dans l'enregistrement (étiquettes faibles), et le second pour leur localisation dans le temps. Nous avons participé au défi avec deux approches distinctes : l'une avec un WGRU et l'autre fondée sur l'approche MMM.

Dans cette section, nous décrivons des données fournies par le défi, les architectures des modèles utilisés, les étapes d'entraînement de ces systèmes ainsi que les pré et post-traitements effectués sur les prédictions temporelles.

2.4.1 Jeux de données

La tâche 4 du DCASE 2018 est liée à la découverte d'évènements audio à partir d'un ensemble de 10 catégories :*Speech*, *Dog*, *Cat*, *Alarm bell/Ring*, *Dishes*, *Frying*, *Blender*, *Running water*, *vacuum cleaner*, et *electric shaver/toothbrush*. Tous les enregistrements sont des clips de 10 secondes extraits de vidéos d'utilisateurs Youtube et font partie du corpus Audioset [Gemmeke et al., 2017]. Les enregistrements contiennent le plus souvent de multiples catégories d'évènements qui peuvent se chevaucher et se répéter. Le corpus du défi était divisé en plusieurs sous-ensembles de données :

- ($\mathcal{T}$) Un ensemble de test composé de 279 exemples fortement annotés.
- ($\mathcal{V}$) Un ensemble d'évaluation composé de 880 exemples fortement annotés.
- ($\mathcal{E}_l$) Un sous-ensemble d'entraînement faiblement annoté contenant 1578 exemples (2224 évènements), qui sera désigné comme le sous-ensemble d'entraînement « faible ». Chaque annotation a été vérifiée par des annotateurs humains.
- ($\mathcal{E}_{n_i}$) Un second sous-ensemble d'entraînement non-annoté, contenant 14412 enregistrements où occurrent les mêmes classes que le premier sous-ensemble.
- ($\mathcal{E}_{n_o}$) Un dernier jeu d'entraînement non-annoté, de 39999 enregistrements, ne contenant pas forcément que les classes d'évènements cibles.

Nous avons utilisé le sous-ensemble d'entraînement $\mathcal{E}_l$ pour entraîner les modèles, et dans le cas du modèle WGRU uniquement nous avons ajouté le sous-ensemble $\mathcal{E}_{n_i}$ à l'aide des pseudos étiquettes générées automatiquement. Les participants ont été classés en fonction des résultats de localisation sur le sous-ensemble $\mathcal{V}$, en termes de F-score [1].

1. Les résultats sont disponibles en ligne : `http://dcase.community/challenge2018/`

2.4.2 Paramètres acoustiques

Chaque enregistrement audio est échantillonné à 22 kHz avant de calculer le log-mel spectrogramme (64 bandes mel) avec une fenêtre et un pas de 2048 et 512 échantillons respectivement. Ainsi, pour une séquence de 10 secondes, une matrice de 64×431 est extraite et, est utilisée en entrée des réseaux de neurones.

Différentes méthodes de normalisation ont été testées comme étape de pré-traitement, telles que la *suppression de la moyenne globale*, ou encore la *normalisation de la moyenne et de la variance*, mais aucun gain n'a été observé par rapport à l'utilisation du spectrogramme log-mel brut.

2.4.3 Modèles

Pour notre participation au défi, nous avons utilisé deux CRNN pour l'approche WGRU et un couple CNN / CRNN pour l'approche MMM.

Système WGRU

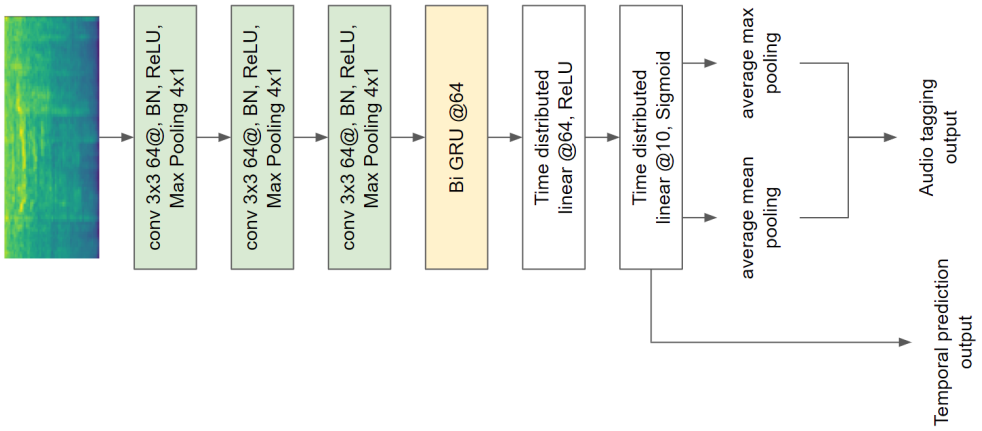

Figure 2.8 – Architecture CRNN avec deux sorties, une pour l'*audio tagging* et une seconde pour les prédictions temporelles

Les deux CRNN ont la même architecture, décrite dans la figure 2.8. Il est composé de trois blocs convolutifs, suivis d'une couche récurrente GRU / WGRU bidirectionnelle. Il possède deux sorties, une pour la classification et la seconde pour la localisation. Chaque bloc de convolution est lui-même composé d'une convolution avec un kernel 3×3 et 64 filtres, une fonction d'activation ReLU et pour finir un *Max Pooling* avec un kernel 4×1. Le choix de ce kernel n'est pas anodin car, avec un log-mel spectrogramme de taille 64×431 cela permet d'obtenir un simple vecteur de $1x431$ à donner en entrée à la couche récurrente.

`task-large-scale-weakly-labeled-semi-supervised-sound-event-detection-results`

Il s'agit de la même architecture que celle utilisée par le modèle de référence (Baseline) fourni par les organisateurs du défi [2].

Système MMM

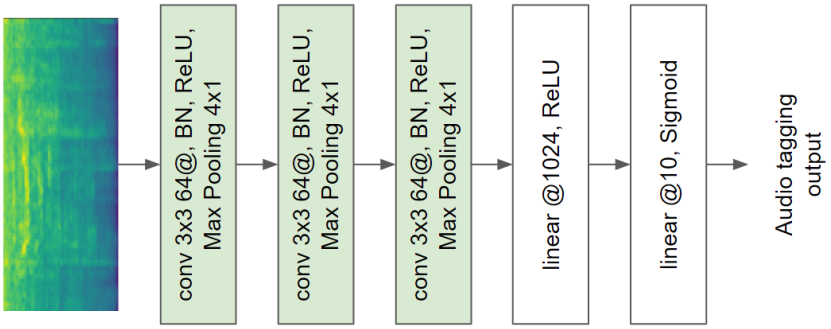

Figure 2.9 – Architecture du CNN du système MMM, utilisé pour la classification

L'approche MMM, utilise elle aussi deux réseaux. Le premier est un CNN présenté dans la figure 2.9. Il est composé des même trois blocs convolutifs que 2.8, suivis de deux couches Fully Connected Network (FC) de 1024 et 10 unités. Il sera utilisé pour effectuer la classification des évènements sonores.

Le second réseau est identique à 2.8 mais sans la sortie de classification. Il sera utilisé uniquement effectuer la prédiction temporelle.

2.4.4 Entraînement

Pour l'entraînement de notre modèle, nous avons utilisé l'optimiseur Adam [Kingma and Ba, 2017] avec un taux d'entraînement de $1e^{-3}$ qui est divisé par deux après 30 et 60 époques. L'entraînement se fait sur 100 époques.

Pour le système WGRU, un premier modèle est entraîné pour la classification en utilisant $\mathcal{E}_l$. Ce dernier est alors utilisé pour générer des pseudo-étiquettes pour le sous-ensemble $\mathcal{E}_{n_i}$. Une fois ces labels artificiels créés, le modèle est de nouveau entraîné avec ces deux sous-ensembles et leur utilisation permet d'obtenir des résultats de classification légèrement meilleurs. Les prédictions temporelles sont alors récupérées à partir de ce dernier modèle.

Pour l'approche utilisant le MMM, nous avons utilisé deux modèles : un CNN pour la tâche de classification avec une fonction de perte classique d'entropie croisée binaire, et un CRNN pour la tâche de localisation avec la fonction de coût 2.2.

2. https://github.com/DCASE-REPO/dcase2018_baseline/tree/master/task4/

2.4.5 Prédictions temporelles binaires

Comme les prédictions temporelles sont faites au niveau de chaque trame acoustique, une binarisation en « segments » est nécessaire d'appliquer un traitement supplémentaire pour faire ce découpage. Pour ce défi, les dix seuils pour les dix classes ont été sélectionnés automatiquement en effectuant une recherche rapide basée sur un algorithme génétique inspiré du recuit simulé ou (*simulated annealing*) [Kirkpatrick et al., 1983]. Cette méthode permet d'atteindre des valeurs de seuil optimales avec un temps d'exécution faible. La segmentation de ces courbes peut être faite de plusieurs façons différentes, c'est pourquoi le chapitre suivant (Chapitre 3) lui est consacré.

2.4.6 Résultats sur le jeu de test

Les résultats sur le jeu de test sont présentés dans le tableau 2.1, en termes de F-scores calculés par évènement, avec une tolérance de 200 ms sur les temps de début (*onsets*) et de 200 ms ou bien de 20 % de la durée de l'évènement pour les temps de fin (*offsets*).

Approche	Baseline	WGRU	MMM	[JiaKai]	[Liu et al.]	[Kothinti et al.]
F-score (%)	14,06	16,77	24,58	25,9	51,6	30,1
Alarm bell	3,9	17,6	28,3	na	40,4	34,9
Blender	15,4	11,6	10,1	na	48,7	20,3
Cat	0,0	0,0	48,9	na	61,6	31,2
Dishes	0,0	0,0	0,0	na	25,7	17,8
Dog	0,0	4,8	18,6	na	53,4	48,1
shaver, toothbrush	32,4	33,3	28,6	na	58,2	22,6
Frying	31,0	29,5	26,7	na	51,9	10,5
Running water	11,4	7,1	10,3	na	54,4	33,3
Speech	0,0	19,4	22,3	na	48,1	36,2
Vacuum cleaner	46,5	40,0	52,2	na	73,7	45,5

Table 2.1 – F-score Mesures globales et par classe sur le sous-ensemble de test $\mathcal{T}$.

Sur le jeu de test, WGRU surpasse légèrement le système de référence avec un F-score de 16,77 %, soit 2,7 points de mieux que le système de référence, qui a un F-score de 14,06 %. MMM fait mieux que la baseline de 10,52 points avec un F-score de 24,58 %. Certaines classes ne sont pas détectées du tout, *Dishes* et *Cat* pour WGRU, *Dishes* pour MMM, tandis que pour la Baseline ce sont les classes *Dishes*, *Dog*, *Cat*, et *Speech*.

L'utilisation d'un réseau WGRU permet donc d'effectuer des prédictions temporelles à partir d'annotations faibles pour certaines classes. Cependant, les performances du WGRU sont bien moins satisfaisantes comparées à celles de MMM et des approches proposées par les meilleurs participants tels que [JiaKai, 2018] ou [Liu et al., 2018]. Ces derniers utilisent des CRNN, ainsi que des approches d'apprentissage semi-supervisé telles que le

Mean-Teacher (MT) pour tirer parti de l'ensemble non-supervisé $\mathcal{E}_{n_i}$. Nous décrirons et utiliserons l'algorithme MT dans la deuxième partie du manuscrit, dans le chapitre 5 - 5.1.1.

2.4.7 Détails sur le choix du poids ω

Approche	Poids ω	F-score (%)
Baseline	_	14,06
WGRU	1,00	6,68
	0,50	4,69
	0,30	8,24
	0,20	11,35
Combinaison de deux poids	1,00 et 0,20	16,77

Table 2.2 – Impact du poids utilisé avec WGRU

Le tableau 2.2 montre les F-scores de WGRU en utilisant différentes valeurs fr ω. Comme on peut le constater, la performance du WGRU est moins bonne que la baseline pour les quatre valeurs testées. ω a cependant un impact significatif sur les résultats. Par exemple, la valeur $\omega = 0,20$ donne le meilleur F-score de 11,35 %. Les valeurs de pondération plus petites se sont révélées moins efficaces, ce qui semble montrer que le fait de conserver certaines informations de l'état précédent de la cellule cachée est important.

Les meilleurs résultats, également présentés dans le tableau 2.1, ont été obtenus en combinant deux prédictions, celles avec avec un poids de $1,00$ et avec un poids $0,20$ en fonction des catégories de sons. La combinaison des prédictions des WGRU est subjective et a été réalisée après observation de ces résultats sur le jeu de données de test. Les classes ont été divisées en deux catégories : les sons stationnaires (*Blender*, *Electric shaver/toothbrush*, *Running water*, *Vacuum cleaner*) et courts (*Alarm bell/ring*, *Cat*, *Dog*, *Speech*). Les prédictions du WGRU non-pondérées ($\omega = 1,00$) sont conservées pour les sons stationnaires et celles du WGRU pondérées à $0,20$ pour les sons courts.

2.4.8 Résultats sur le jeu d'évaluation

Les résultats sur le jeu d'évaluation sont présentés dans le tableau 2.3, en termes de F-scores calculés par évènement, avec une tolérance de 200 ms sur les temps de début (*onsets*) et de 200 ms ou bien de 20 % de la durée de l'évènement pour les temps de fin (*offsets*).

Le score de la baseline est de 10,8 % sur le jeu d'évaluation, soit une baisse de presque 4 points par rapport au F-score sur le test. Cela semble indiquer que le jeu d'évaluation était plus difficile que le jeu de test. MMM voit également le F-score chuter de 24,6 % sur le test

	Baseline	WGRU	MMM	[JiaKai]	[Liu et al.]	[Kothinti et al.]
F-score (%)	10,8	8,4	16,6	32,4	29,9	22,4
Alarm bell	4,8	2,5	23,8	49,9	46,0	36,7
Blender	12,7	5,9	5,1	38,2	27,1	22,0
Cat	2,9	0,5	25,3	3,6	20,3	20,5
Dishes	0,4	0,3	0,7	3,2	13,0	12,8
Dog	2,4	2,8	4,1	18,1	26,5	26,5
shaver, toothbrush	20,0	17,7	6,5	48,7	37,6	24,3
Frying	24,5	20,9	18,3	35,4	10,9	0,0
Running water	10,1	8,6	15,0	31,2	23,9	9,6
Speech	0,1	4,0	22,3	46,8	43,1	34,3
Vacuum cleaner	30,2	21,6	44,9	48,3	50,0	37,0

Table 2.3 – F-scores globaux et par classe sur le sous-ensemble d'évaluation $\mathcal{T}$.

à 16,6 %, soit 8 points de baisse. Ce résultat classe ce système à la 12ème place sur les 17 systèmes participants.

En ce qui concerne WGRU, les deux valeurs de ω ont été choisies sur le jeu de test. On constate que sur le jeu d'évaluation, les résultats sont bien moins bons que sur le test, en-dessous de la baseline, ce qui montre le manque de pouvoir de généralisation du modèle WGRU lui-même et des valeurs choisies pour ω.

2.5 Discussion

Dans ce chapitre, j'ai décrit les premières expériences de ma thèse, en détection d'évènements sonores avec étiquettes faible, qui ont fait l'objet d'une participation à la tâche 4 du défi DCASE en 2018. En utilisant le jeu de données fourni, nous avions un objectif double : d'abord classer les évènements sonores dans les enregistrements audio, et ensuite les localiser aussi précisément que possible dans le temps. La spécificité était que seules les étiquettes « faibles » étaient fournies pour entraîner les modèles.

Pour relever ce défi, nous avons essayé de modifier le comportement de cellules récurrentes appelées WGRU avec l'introduction d'un mécanisme d'oubli. L'influence des états cachés de la couche récurrente est diminuée avec l'ajout d'un poids ω pendant l'inférence uniquement. Cela permet d'éviter de prédire le même score tout au long d'un enregistrement. Ce poids peut, et doit *a priori* être adapté à la durée des évènements à catégoriser. En fonction des résultats obtenus sur le jeu de test du challenge, nous avons choisi d'utiliser deux valeurs différentes en fonction de la durée attendue des catégories de sons. Il est ensuite possible de combiner les résultats de plusieurs inférences pour améliorer les performances finales. Cette approche a donné un F-score global légèrement supérieur au système de référence avec un F-score de 16,77 %. Il s'agit cependant d'un gain très modeste, et qui ne s'est pas confirmé sur le jeu d'évaluation du challenge, avec un F-score de 8,4 %.

Les résultats sur le jeu d'évaluation ont donc été décevants. J'ai passé par la suite un certain temps à analyser les valeurs des états cachés de modèles CRNN entraîné avec des tags, mais sans parvenir à tirer d'amélioration de la méthode. Une hypothèse qui expliquerait ces résultats est le fait qu'en introduisant une pondération à l'inférence, nous créons de fait un mismatch, une inconsistance, structurels dans le fonctionnement du réseau. En effet, le réseau est entraîné d'une certaine façon et testé d'une autre avec le poids ω. Enfin une autre limite de cette approche porte sur le choix de ω qu'il faudrait faire pour chaque classe, avec les inférences multiples nécessaires. Sur un dataset tel que celui de DCASE, cela est tout à fait possible, car seulement dix classes sont à distinguer. Cependant sur des datasets plus conséquents tels qu'Audioset [Gemmeke et al., 2017] et les 527 classes, cela s'avère beaucoup plus fastidieux à mettre en oeuvre.

Dans le cadre du même défi, mon directeur de thèse a mise en oeuvre une approche appelée MMM, dans laquelle un CRNN est entraîné à prédire des tags, en utilisant une fonction de coût composite comparant les étiquettes faibles et les statistiques des prédictions trame à trame (minimum, moyenne et maximum). Cette méthode a obtenu un F-score de 16,6 % sur le jeu d'évaluation, le plaçant à la 12ème place sur 17 équipes. À la suite du challenge, nous avons remarqué que que la classe *Dishes* n'était jamais détectée, et que les exemples d'entraînement de cette classe comportaient presque toujours des évènements de la classe *Frying* et de ce fait le système ne parvient pas à apprendre à les distinguer. Nous avons proposé une solution à ce problème en ajoutant une pénalité de similarité cosinus entre les prédictions temporelles de classes co-occurrentes est ajoutée à la fonction de perte [Pellegrini and Cances, 2019].

Chapitre 3

Post-traitement pour la localisation d'évènements sonores

Ce chapitre explore différentes approches de post-traitement pour améliorer les résultats bruts fournis par les systèmes de détection d'évènements sonores. Je compare plusieurs algorithmes de lissage, de segmentation, et d'optimisation de seuils pour obtenir la meilleure segmentation des prédictions temporelles d'un modèle.

Les expériences ont été menées sur les données du challenge DCASE tâche 4 2018, les mêmes que celles du chapitre précédent. Nous avons utilisé le modèle MMM qui avait donné les meilleurs résultats, et, pour des raisons de reproductibilité des résultats, nous avons aussi testé le système baseline des organisateurs. Pour se concentrer sur l'impact du post-traitement des prédictions temporelles uniquement, nous avons considéré une configuration dite « Oracle », où nous savons quelles catégories d'évènements arrivent dans les enregistrements. Seules les prédictions de ces évènements sont retenues pour le post-traitement et le calcul des F-scores. Par la suite, nous reportons aussi les résultats sans oracle, en retenant les prédictions temporelles des évènements prédits par le modèle automatiquement.

Comme nous le verrons, la sélection des paramètres associés aux algorithmes peut être faite de deux façons : une première dite statistique, et une seconde dite paramétrique nécessitant une étape d'optimisation supplémentaire. Nous comparons plusieurs algorithmes de segmentation connus tel que le seuillage absolu ou par hystérésis, et différentes méthodes pour calculer ou optimiser les paramètres des algorithmes de segmentation. Pour finir, nous montrons qu'il s'agit d'une étape importante qui peut apporter un gain significatif sur la segmentation des évènements sonores. En effet, notre modèle MMM qui avait obtenu un F-score de 16,6% sur l'évaluation, voit son score augmenter à 32,0%

Pour rappel, les systèmes de détection d'évènements sonores produisent des probabilités temporelles qui nécessitent d'être seuillées pour pouvoir extraire des segments représentant les évènements sonores présents dans l'enregistrement audio traité. Ce post-traitement est souvent accompagné d'une étape de lissage et de recherche des meilleures valeurs de seuil. Cependant, cette partie reste dans l'ensemble peu décrite dans la littérature pour laisser plus de place, entre autres, à la description des modèles ou des algorithmes d'apprentissage. Pourtant, il s'agit d'une tâche explorée dans de nombreux différents domaines tels que l'indexation multimédia [Ballan et al., 2009], la reconnaissance de contexte [Barchiesi et al., 2015], ou encore pour la surveillance et la sécurité [Harma et al., 2005].

Les RNN sont performants lorsqu'ils sont entraînés avec des annotations fortes grâce à leur capacité à se souvenir de leur état passé [Parascandolo et al., 2016b]. Il est alors possible de directement utiliser la sortie du système telle quelle, sans utiliser de post-traitement. Cependant, lorsqu'on entraîne un RNN avec des annotations faibles, les réseaux récurrents ne sont pas capables de fournir des prédictions propres, et directement utilisables pour produire une segmentation correcte. (tel que nous l'avons vu dans le chapitre précédant 2.1.2.) Dans ce cas, le post-traitement devient important et on peut récupérer quelques mécanismes dans la littérature.

Dans les travaux de [Koutini et al., 2018], les CRNN ont été utilisés pour faire des prédictions d'étiquettes pseudo-fortes et, ils utilisent des filtres médians et gaussiens pour supprimer les segments trop petits ou fusionner les segments séparés par de trop courtes pauses. Ces filtres sont mentionnés, mais pas leurs paramètres. [Cakir et al., 2015] proposent une approche similaire en appliquant un filtre médian sur les prédictions binarisées pour regrouper les segments séparés par de trop courtes pauses. Les seules informations disponibles concernant l'étape de seuillage sont l'utilisation d'un seuil absolu testé sur 8 valeurs (entre 0,2 et 0,9 avec un pas de 0,1). C'est aussi le cas des travaux de [Harb and Pernkopf, 2018] qui utilisent aussi un filtre médian et un seuil dont la valeur n'est pas décrite, comme pour les travaux de [Hou and Li, 2018] ou encore [Liu et al., 2018].

Dans les travaux de [Guo et al., 2018] le système de prédiction temporel est couplé à un système de classification global qui permet alors de ne garder que les prédictions correspondantes aux classes présentes dans l'enregistrement. Ils appellent ce mécanisme la « correction ». Ils ne mentionnent pas comment le seuil est appliqué, ni la valeur de ce dernier.

[Xia et al., 2017] ont abordé la question de la sélection du seuil dans un contexte de détection polyphonique et fournissent les détails de leur étape de post-traitement. Ils proposent deux approches pour estimer les seuils. Dans la première, pour éviter que le seuil soit trop haut ou trop bas, ils utilisent l'information du contour du signal pour déterminer la position du seuil $T_t = \alpha \times \max(y_{c,t})$. Le rapport α pour la trame t est calculé de manière globale en utilisant l'intégralité du jeu de données. Il s'agit du rapport entre le nombre de trames non vides (avec au moins un évènement) et le nombre de trames totales. $y_{c,t}$

correspond aux prédictions du système pour la classe c à la trame t. Ils utilisent aussi une autre approche pour estimer la valeur du seuil pour chaque trame audio en utilisant un Long short-term memory (LSTM).

Dans certain cas plus extrême, aucune information n'est fournie concernant une éventuelle utilisation de post-traitement ni même sur comment la segmentation est réalisée. À notre connaissance, il n'existe pas d'analyse systématique de l'impact du post-traitement dans la tâche de détection d'évènements sonores. Nous supposons que de nombreux travaux de recherche pourraient bénéficier d'une présentation claire des différentes approches existantes ainsi que de leur évaluation détaillée.

3.1 Le post-traitement des probabilités temporelles

Comme il a été mentionné brièvement au début du chapitre, le post-traitement est une étape primordiale pour l'utilisation des prédictions temporelles émises par les SED. Elle s'applique en trois phases : une première de lissage, une seconde de seuillage et une dernière d'optimisation.

3.1.1 Les algorithmes de lissage

Le lissage des probabilités est effectué grâce à une moyenne mobile (ou dite glissante). Elle est souvent utilisée pour analyser des séries temporelles en supprimant le bruit. Cela permet de faire ressortir les tendances à plus long terme et d'éviter la création de plusieurs petits segments d'évènements qui auraient pu n'être qu'un. Il existe plusieurs façons de calculer cette moyenne mobile : la moyenne mobile arithmétique, la moyenne mobile pondérée ou la moyenne mobile exponentielle pour n'en citer que quelques-unes. La taille de la fenêtre de lissage peut être fortement influencée par la nature de l'évènement audio et doit, par conséquent, être adaptée pour chacune des classes. Pour des évènements tels que « claquement de main » ou des bruits de vaisselle par exemple, cette fenêtre sera beaucoup plus petite pour permettre la détection de ces évènements courts. Pour des bruits uniformes beaucoup plus longs comme celui d'un aspirateur, la fenêtre devra être plus grande.

3.1.2 Trois algorithmes de segmentation

Cette section présente les différentes approches de segmentation qui peuvent être utilisées pour segmenter une prédiction temporelle.

Seuillage par valeur absolue

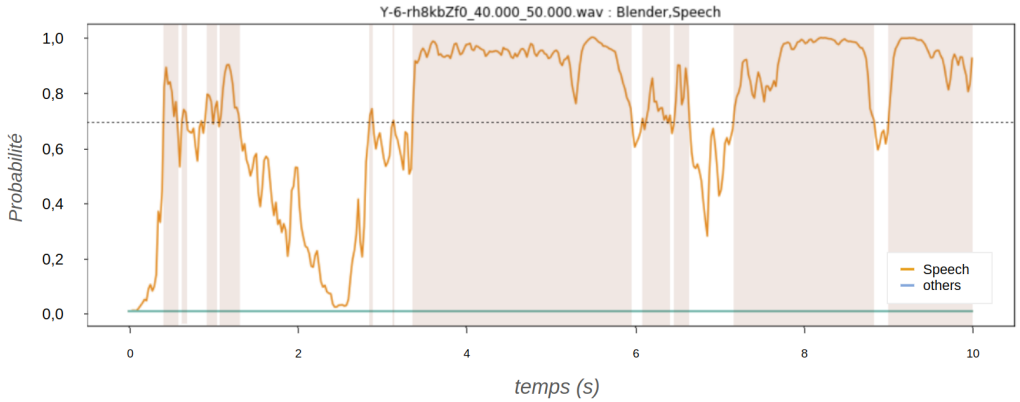

Figure 3.1 – La prédiction temporelle de la classe « Speech » est en orange, la valeur de seuil est représentée par la ligne en pointillée noire. Les autres classes ne sont pas détectés. Les segments détectés correspondent aux rectangles pleins.

Le seuillage absolu consiste à appliquer directement un seuil unique et arbitraire aux prédictions temporelles. Cette approche naïve donne néanmoins des résultats exploitables qui peuvent se rapprocher des meilleurs résultats que nous ayons obtenus dans certains cas. La valeur du seuil peut être arbitraire, calculée ou obtenue grâce à un algorithme d'optimisation. Le petit nombre de paramètres dont il dépend le rend par ailleurs très rapide à optimiser.

La figure 3.1 permet de visualiser les résultats obtenus après l'utilisation d'un seuil absolu sur les prédictions temporelles d'un système SED sur un fichier contenant des évènements de la classe Speech. On peut observer un grand nombre de segments créés, dont certains sont très courts.

Seuillage par hystérésis

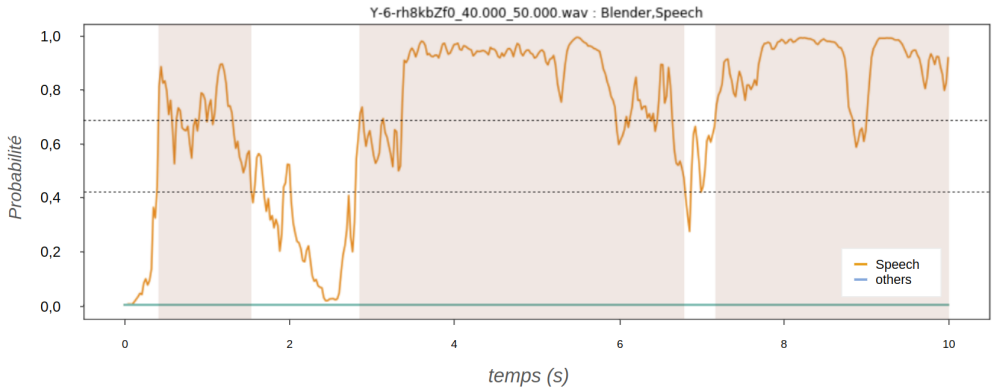

Figure 3.2 – Mêmes conventions que la figure 3.1, mais cette fois deux seuils sont utilisés.

Le seuillage par hystérésis est utilisé dans de nombreux domaines, tels que la mécanique, l'imagerie, l'électronique, etc. Elle utilise deux seuils, le premier est utilisé pour déterminer

le début d'un segment et l'autre (plus petit) sa fin. Cet algorithme est utilisé lorsque les probabilités sont instables et changent à un rythme élevé. Il devrait donc diminuer le nombre d'évènements détectés par l'algorithme et ainsi, réduire les pourcentages d'insertion et de suppression, ce qui donne un taux d'erreur plus petit que l'approche du seuil absolu. La figure 3.2 permet de visualiser les résultats de segmentation obtenus après l'utilisation d'un seuil par hystérésis sur le même exemple que la figure 3.1. Comparé au seuillage absolu, on peut voir que seulement trois segments ont été générés, les petits segments dus au « bruit » dans les prédictions ont disparu.

Seuillage par dérivée

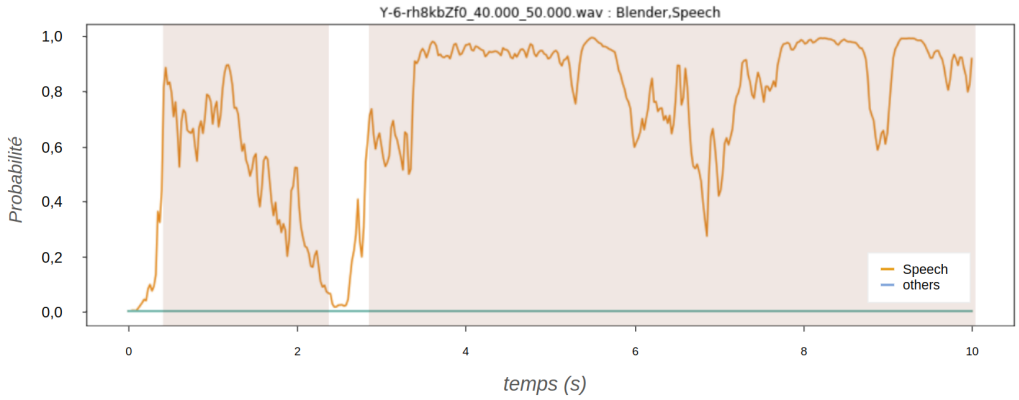

Figure 3.3 – Contrairement au seuillage par hystérésis, le premier segment est plus long et se finit lorsque les probabilités se stabilisent pendant une petite période, ici 100 ms, à une valeur proche de zéro.

Cette méthode est différente des deux précédentes, car elle n'utilise pas directement la valeur de la probabilité pour déterminer les extrémités du segment. Elle utilise plutôt la vitesse à laquelle les probabilités évoluent. Concrètement, cela signifie qu'une probabilité qui augmente rapidement déterminera le début d'un segment et une probabilité qui diminue rapidement, suivie d'un plateau, la fin de ce segment. Cela permet de détecter des fins de segments même si les prédictions restent élevées. On peut observer les résultats de cette segmentation avec la figure 3.3. Visuellement, cela se concrétise par la fusion de deux segments rapprochés.

3.1.3 Sélection des paramètres via une approche statistique

Chacun de ces algorithmes possède un ensemble de paramètres à définir et à ajuster afin d'obtenir la meilleure segmentation possible. Leurs valeurs peuvent être déterminées, soit à partir de statistiques extraites des prédictions du système sur l'ensemble des données de développement, soit grâce à des algorithmes de recherche et d'optimisation.

Les méthodes statistiques sont directement basées sur un certain nombre d'informations extraites des prédictions temporelles produites par le système. Des mesures telles que la

moyenne, la médiane ou encore l'écart type peuvent être calculées à partir de ces prédictions sur l'ensemble du jeu de données ou du fichier en cours de traitement, on parlera alors de statistiques globales ou locales. Les avantages principaux de ces approches sont la vitesse à laquelle ces informations peuvent être calculées et la simplicité de leur déploiement dans n'importe quel système.

Pour les formules ci-dessous, $\mathcal{S}$ représente le jeu de données, $\mathcal{C}$ l'ensemble des catégories à classer, $p(x_t)$ la prédiction à l'instant t produite par le système, N le nombre d'exemples disponibles, et Th le ou les seuils calculés.

Pour un seuillage absolu

La moyenne des prédictions issues du système est utilisée pour déterminer la valeur du seuil absolu (voir Eq 3.1). Il est aussi possible (et conseillé) d'utiliser un seuil pour chaque classe, dans ce cas, c'est l'équation 3.2 qui est utilisée.

$$\forall x \in \mathcal{S}, \mathrm{Th} = \frac{1}{N} \sum p(x) \tag{3.1}$$

$$\forall x \in \mathcal{S}, \forall c \in \mathcal{C}, \mathrm{Th}_c = \frac{1}{N} \sum p_c(x) \tag{3.2}$$

Pour un seuillage par hystérésis

La moyenne des prédictions issues du système est calculée de la même façon que pour un seuillage absolu. C'est l'écart type σ qui sera utilisé pour déterminer le seuil haut Th_h et bas Th_b de la façon suivante :

$$\begin{aligned} \mathrm{Th}_b &= \mathrm{Th} - \sigma \\ \mathrm{Th}_h &= \mathrm{Th} + \sigma \end{aligned} \tag{3.3}$$

L'écart type sera lui aussi calculé soit sur l'ensemble du jeu de données - dans ce cas-là, c'est la moyenne des écarts types de chaque classe et chaque exemple qui est utilisé - soit sur l'exemple courant et alors, c'est l'écart type de la classe qui sera utilisé.

Pour un seuillage par dérivée

Il n'est pas possible de déterminer les seuils à utiliser directement depuis les prédictions du système. Pour les déterminer, il est alors nécessaire d'utiliser une approche par optimisation.

3.1.4 Sélection des paramètres à l'aide d'un algorithme d'optimisation

Nous avons vu ci-dessus comment il était possible de calculer les différentes valeurs de seuil pour la segmentation par seuillage absolu et hystérésis. Cependant, il est toujours nécessaire de fixer manuellement des paramètres pour le lissage (type, taille de fenêtre, et pas) et pour tous ceux de l'approche par dérivée. Il est aussi possible que ces seuils ne soient pas optimaux. Néanmoins, la recherche de la meilleure combinaison de paramètres est un travail coûteux en temps, et il n'est souvent pas possible de faire une recherche exhaustive de toutes les combinaisons possibles. En effet, en fonction du nombre de paramètres à régler, la croissance de l'espace de recherche est exponentielle et le temps d'exécution dépasse souvent des durées raisonnables. Par conséquent, nous avons implémenté une recherche dichotomique et un algorithme génétique qui sont tous deux décrits ci-dessous.

La recherche par dichotomie

Pour chaque paramètre à optimiser, l'utilisateur fournit un intervalle de recherche initial. L'algorithme essaie alors chaque combinaison existante avec une résolution grossière, puis choisit celle qui donne le meilleur résultat. À partir de ce dernier, un nouvel intervalle plus petit est calculé et le processus complet est répété. La précision est ainsi plus fine à chaque répétition et l'espace de recherche est petit à petit réduit. L'algorithme s'arrête lorsque le nombre d'étapes indiqué par l'utilisateur est atteint. La figure 3.4 permet de visualiser le fonctionnement de ce processus et l'image 3.5 de visualiser la convergence de l'algorithme vers une valeur optimale de 29 % de F-score après seulement 1727 itérations. On peut aussi observer la distribution des seuils qui se concentrent entre 0,2 et 0,3.

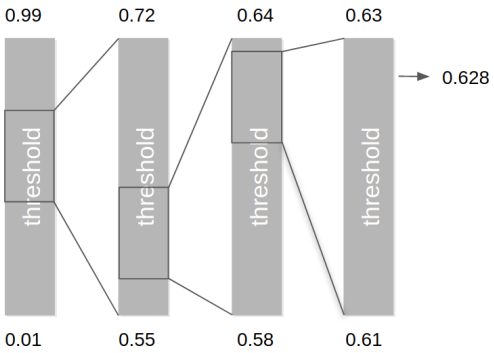

Figure 3.4 – Une optimisation par dichotomie en quatre étapes est représentée ici. Chaque rectangle gris plein représente l'espace de recherche de l'étape en cours. À chaque tour, l'intervalle de recherche diminue et la précision du résultat augmente.

L'algorithme de recherche dichotomique, comparé à une approche exhaustive de toutes les combinaisons possibles, réduit considérablement le temps nécessaire pour atteindre une

solution quasi optimale avec une excellente précision. Cependant, le temps d'exécution dépend toujours du nombre de paramètres à régler et du nombre d'itérations pour chaque étape. Le cumul total de combinaisons augmente de façon exponentielle.

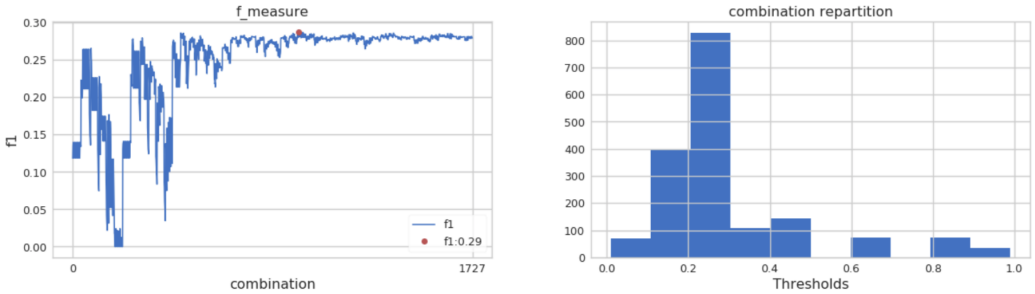

Figure 3.5 – Sur la figure de gauche, on peut visualiser le score du système (ici le F-score). Sur la figure de droite, on peut voir la distribution des valeurs de seuil.

La recherche grâce à un algorithme génétique

Le but des algorithmes génétiques est d'obtenir une solution approchée à un problème d'optimisation, lorsqu'il n'existe pas de méthode exacte pour le résoudre en un temps raisonnable. Le fonctionnement est basé sur la notion de sélection naturelle et s'applique à une population de solutions potentielles au problème donné. La solution est approchée par « bonds » successifs, et seulement le meilleur résultat est gardé à chaque étape. Cette approche peut être visualisée dans la figure 3.6.

Comparé à une approche par dichotomie, l'algorithme génétique permet de garder le contrôle sur le temps d'exécution, car le nombre de « bonds » est défini par l'utilisateur. La convergence de l'algorithme est alors directement dépendante de sa durée d'exécution, permettant d'effectuer des tests rapides lors de la phase de prototypage et de fixer le temps alloué pour l'optimisation lors de l'évaluation. C'est un algorithme intéressant lorsque les ressources disponibles sont limitées par exemple.

L'heuristique utilisée pour réaliser l'optimisation est une translation aléatoire δ des seuils utilisés. Ce δ est tiré d'une distribution normale centrée sur la valeur du dernier meilleur seuil. La figure 3.7 représente l'ensemble des valeurs de δ possible ainsi que la probabilité de tirer la dite valeur en fonction du seuil de la population actuelle.

3.2 Évaluer l'impact du post-traitement

Pour évaluer l'impact du post-traitement dans le cadre d'une tâche de détection d'évènements sonores, nous réutilisons les données de la tâche 4 du DCASE 2018 ainsi que notre meilleure soumission, le modèle MMM. Comme l'objectif est d'analyser les gains que l'on peut obtenir grâce au post-traitement seul, nous considérons comme situation idéale la

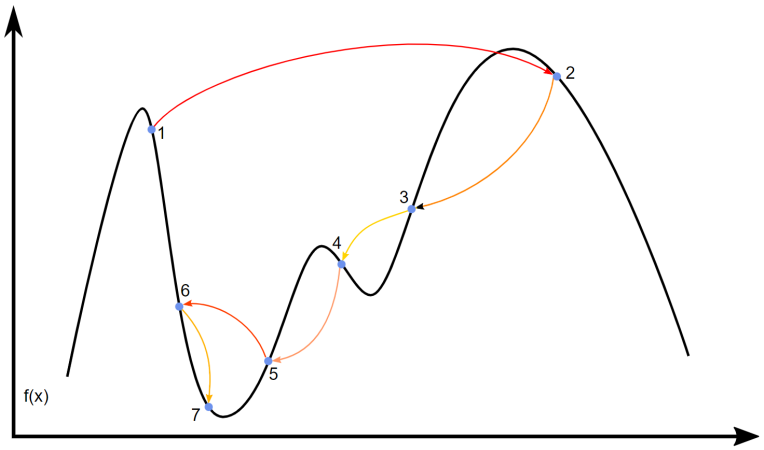

Figure 3.6 – Cette image représente dans un espace à deux dimensions la suite de bonds permettant d'atteindre un minimum global de la fonction $f(x)$ à optimiser. L'ensemble des résultats possible de cette fonction est représenté par la courbe noire.

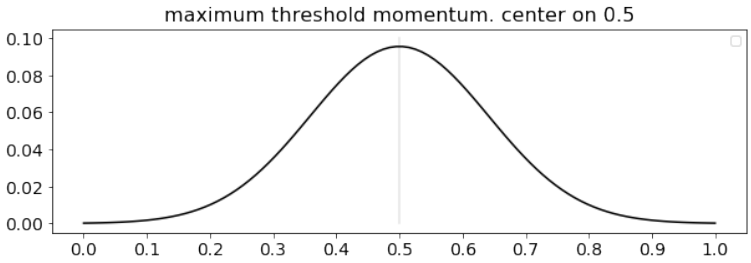

Figure 3.7 – Le δ maximal du seuil centré sur une valeur de 0,5. À chaque itération, le δ aléatoire est tiré d'une distribution normale, centrée sur la valeur du dernier meilleur seuil de la classe ciblée au cours du processus d'optimisation.

connaissance a priori de toutes les classes présentes dans chaque enregistrement audio. Nous appelons cette configuration « Oracle ».

Cette section présente le jeu de données utilisé pour réaliser l'expérimentation, les modèles d'apprentissages profonds employés pour la classification et la segmentation, comment ces derniers sont entraînés et finalement le protocole expérimental employé.

3.2.1 Jeu de données utilisé

Les enregistrements audio proviennent de la tâche 4 du défi DCASE 2018 [Serizel et al., 2018]. Ce jeu de données est déjà détaillé dans le Chapitre 2, Section 2.4.1. Pour résumer, chaque enregistrement de DCASE dure 10 secondes et peut inclure un ou plusieurs évènements d'un ensemble de catégories sonores se produisant dans des environnements domestiques. Ainsi on retrouve les catégories de bruits suivantes : *Speech*, *Dog*, *Cat*, *Alarme/sonnerie*, *Vaisselle*, *Friture*, *Mixeur*, *Eau courante*, *Aspirateur*, et *Rasoir électrique/brosse à dents*. Ces enregistrements sont divisés en trois ensembles, un pour l'entraînement composé de 1578 exemples faiblement annotés, un pour le test composé de 279 exemples fortement annotés et un ensemble d'évaluation, utilisé par les organisateurs

du défi, composé de 880 exemples fortement annotés.

Nos systèmes seront donc entraînés en utilisant le sous-ensemble « faible ». Nous avons converti ces enregistrements en un signal mono, échantillonné à 22 kHz, puis convertit en 64 x 431 log-mel spectrogramme. Les seuils seront optimisés en utilisant l'ensemble de tests et les performances finales seront évaluées sur l'ensemble de validation.

3.2.2 Architecture des modèles d'apprentissage profond

Pour observer l'impact des algorithmes de segmentation, nous avons utilisé deux approches. Les détails de ces implémentations sont fournis dans le chapitre précédant, section 2.4.3 ainsi que l'architecture du modèle de référence et du MMM.

— La première est similaire au système de référence fourni par les organisateurs du challenge [Serizel et al., 2018] (que l'on nommera Baseline). Elle utilise un seul CRNN pour effectuer à la fois la classification des évènements audio (Audio Tagging (AT)) et leur segmentation.

— La seconde méthode est basée sur la fonction de perte MMM, décrite dans le chapitre 2, Section 2.3) tel que proposé dans [Salamon et al., 2017; Cances et al., 2018; Pellegrini and Cances, 2019]. Cela consiste en deux réseaux distincts : un CNN pour la classification, entraîné avec une entropie croisée standard, et un CRNN pour la segmentation, entraîné avec la fonction de coût MMM.

3.2.3 Protocole expérimental

Le post-traitement a lieu après la phase d'apprentissage du modèle, lorsque des seuils sont appliqués aux prédictions temporelles lissées pour obtenir le début et la fin des évènements sonores. Il est effectué dans l'ordre suivant :

1. Les courbes représentant la prédiction du modèle pour chaque trame sont lissées à l'aide de l'algorithme de moyenne mobile. Le lissage n'est utilisé que lorsque les paramètres du post-traitement sont optimisés grâce à un des algorithmes présentés dans la section 3.1.4.
2. Les prédictions temporelles sont segmentées en utilisant l'un des algorithmes de segmentation décrit dans la section 3.1.2.
3. Les segments séparés par un écart plus petit que la marge de tolérance imposée par le défi sont fusionnés. De la même manière, les segments plus petits que cette marge sont supprimés.

Dans le système fourni par les organisateurs du challenge, le post-traitement consiste en un simple seuillage absolu dont le seuil est sélectionné en testant toutes les valeurs comprises entre 0 et 1 avec un pas de 0,1. Aucun lissage n'a été mentionné alors nous avons décidé de

le rajouter en utilisant une moyenne mobile arithmétique (voir 3.1.1) et nous testons toutes les tailles de fenêtre entre 5 et 21 avec un pas de 2. Cette recherche par grille grossière (*coarse grid search*) représente un total de 64 combinaisons possibles.

Nous testerons un total de huit configurations en plus du *coarse grid search*. Les seuils utilisés par chaque algorithme de segmentation pourront être identique pour toute les classes, (classe indépendant CI), ou spécifique à chacune (classe dépendant CD). Nous testerons le seuillage absolu (A), le seuillage par hystérésis (H) et le seuillage par dérivé (D).

Lorsque les paramètres de ces algorithmes sont déterminés à l'aide d'un des algorithmes d'optimisation présentés plus haut, on parlera alors d'approche paramétrique. Lorsqu'ils sont définis par des statistiques extraites sur les prédictions du modèle de l'ensemble du jeu de données, on parlera alors d'approche statistique. Pour ces derniers, la valeur des seuils est déterminée par la moyenne des probabilités en sortie du modèle pour chaque classe et chaque fichier. De la même façon, ces seuils peuvent être indépendants ou non des classes.

Avant de réaliser les expériences, il est nécessaire de résoudre deux problématiques : (1) Comment gérer les erreurs potentielles commises par le système de classification, qui aura forcément un impact sur la segmentation et (2) Comment évaluer le système de segmentation et sélectionner les meilleurs paramètres.

1. Pour éliminer le biais induit par une classification défectueuse des évènements audio, nous avons utilisé les classes des annotations fortes de l'ensemble de test et d'évaluation comme si elles étaient les sorties d'un classificateur parfait. Cela nous permet de ne retenir que les classes pertinentes sur lesquelles les évènements doivent être localisés. Nous appellerons ce mode de fonctionnement : « Oracle ».
2. Nous avons utilisé la métrique évènementielle définie dans [Mesaros et al., 2016]. Plus précisément le score macro-F1, alias F-score, avec les paramètres de précision du défi : une marge de 200 ms sur les débuts de segment et une marge de 20% sur la longueur de ce dernier. La formule du F-score est rappelée ci-dessous :

$$\text{macro-F1} = \frac{1}{C}\sum_{c=0}^{C}\frac{2P^c \times R^c}{P^c + R^c} \tag{3.4}$$

C représente le nombre de classes, P^c est la précision du système et R^c le rappel pour les exemples de la classe c.

3.3 Résultats

Les résultats sont présentés dans le tableau 3.1. De manière générale, ils montrent une grande disparité de valeurs. Le F-score varie de 17,9 % à 23,4 % avec la baseline, et de 25,8 % à 43,9 % avec le modèle MMM. Nous observons donc un impact significatif des algorithmes de post-traitement sur les résultats finaux. Les meilleurs scores sont obtenus en

	Méthode de post-traitement	Baseline Test F-score (%)	Baseline Test ER	Baseline Eval F-score (%)	Baseline Eval ER	MMM Test F-score (%)	MMM Test ER	MMM Eval F-score (%)	MMM Eval ER	TE
	Coarse grid search	20,9	1,2	19,4	1,3	18,2	1,8	15,3	1,8	1
Stat,	Classe Indépendant (CI)	19,9	1,3	17,9	1,5	29,8	2,0	25,8	2,5	0
Stat,	Classe Dépendant (CD)	19,6	1,3	**18,7**	1,4	32,5	1,8	**29,9**	2,4	0
Param,	CI Absolu (CIA)	25,0	1,1	22,8	1,2	44,2	1,1	37,1	1,4	1
Param,	CI Hystérésis (CIH)	25,0	1,1	22,6	1,2	46,4	1,0	40,7	1,2	3
Param,	CI Dérivé (CID)	24,3	1,2	21,0	1,2	43,6	1,2	35,5	1,5	115
Param,	CD Absolu (CDA)	**26,5**	1,1	22,3	1,3	**53,2**	0,9	**43,9**	1,2	10
Param,	CD Hystérésis (CDH)	**26,5**	1,1	23,0	1,2	**53,1**	0,8	42,9	1,1	29
Param,	CD Dérivé (CDD)	26,2	1,1	**23,4**	1,2	52,4	0,9	41,0	1,2	1155

Table 3.1 – F-scores et taux d'erreurs (ER) pour la Baseline et le MMM sur les ensembles de test et d'évaluation « Oracle » (les classes des évènements présents sont supposés connus pour évaluer uniquement le seuillage des prédictions temporelles). La dernière colonne montre le temps d'exécution (TE) relatif de chaque méthode par rapport à la CIA.

utilisant les algorithmes d'optimisation dichotomique, avec un seuil distinct pour chaque classe. Sur l'ensemble d'évaluation, CDD donne un F-score final de 23,4 % avec la baseline, et CDA un F-score final de 43,9 % pour le modèle MMM.

Si l'on regarde de plus près les méthodes basées sur les statistiques, CD donne de meilleures performances avec un F-score final de 18,7 % et 29,9 % respectivement sur la Baseline et MMM. Dans les deux cas, il a donné de meilleurs résultats sur le sous-ensemble d'évaluation que CI. La variante dépendante de la classe semble plus appropriée que l'indépendante, même si elle a donné un résultat légèrement moins bon sur l'ensemble de test (0,3 différence absolue). En effet, si l'on regarde de près la transition entre les ensembles de test et d'évaluation, la différence n'est que de -4 % pour la variante dépendante de la classe et de -10% relatifs pour la variante indépendante de la classe, ce qui rend la première plus robuste.

En fin de compte, les méthodes paramétriques présentent les meilleurs résultats en F-score et en taux d'erreur (ER). Elles sont plus performantes que le seuil choisi manuellement et que les approches statistiques. La Baseline atteint sur Eval un F-score final de 23,4 % avec CDD. Cela représente une amélioration de 4 points (20,6 % relatif). Pour MMM, la méthode CDA donne le meilleur F-score final avec une amélioration de 28,6 points (187,0 % relatif). La meilleure valeur ER sur Eval est de 1,1 % obtenue avec CDH. A chaque fois, les meilleurs scores sont obtenus en utilisant les variantes dépendantes de la classe.

L'observation sur la meilleure robustesse des méthodes dépendantes est aussi valide avec les algorithmes paramétriques. La différence entre les scores sur le jeu de test et d'évaluation n'est que de -8 % pour la classe dépendante, et de -18 % pour la classe indépendante, ce qui indique que la méthode dépendante de la classe, non seulement plus performante, est aussi plus robuste.

Si les méthodes statistiques ont déjà permis d'améliorer le F-score final, les approches paramétriques permettent d'aller encore plus loin, spécifiquement les variantes dépendantes de la classe. Les F-scores maximaux des méthodes statistiques sont de 18,7 % et 29,9 % pour Baseline et MMM, respectivement, et les F-score maximaux des méthodes paramétriques sont de 23,4 % et 43,9 %.

Le lissage

Il est possible d'argumenter sur l'intérêt du lissage, en particulier lors de l'utilisation d'un seuil d'hystérésis, car l'effet pourrait être redondant. Cependant, si nous examinons de plus près la taille des fenêtres obtenues après l'étape d'optimisation, nous pouvons constater une grande variété de tailles allant de 9 (*Dishes*) à 27 (*Vacuum cleaner*) trames. Ceci peut être observé pour les variantes dépendantes de la classe de chaque méthode paramétrique et souligne l'importance de lisser la prédiction produite par le système avant d'appliquer tout algorithme de segmentation.

Temps de calcul

En ce qui concerne le temps de calcul, la meilleure méthode n'est pas nécessairement la plus longue et le gain, s'il y en a un, n'est pas linéaire. En utilisant le CDD à la place du CDA, la baseline n'a bénéficié que de 0,7 point supplémentaire pour un temps de calcul plus de 100 fois plus long, alors que le MMM montre même une perte de 0,8 point. Cependant, l'optimisation de seuil spécifique pour chaque classe apporte systématiquement un gain de performance non négligeable comparé à l'optimisation d'un seuil global et cela même, si le processus dure dix fois plus longtemps.

Répartition de l'optimisation

Utiliser une méthode paramétrique (et donc l'optimisation correspondant) apportera un gain de performance uniforme avec la variante globale de l'algorithme (même seuil pour chaque classe). Cette amélioration n'est plus régulière lorsque la variante dépendante est utilisée. Certaines classes ne tirent aucun avantage de l'optimisation. C'est le cas de la classe *dishes* par exemple.

Performance sans utiliser d'Oracle sur les tags

Enfin, nous avons appliqué ces méthodes sur notre modèle sans utiliser l'*Oracle* mais la classification réelle fournie par nos modèles. Les résultats sont montrés dans le tableau 3.2, où nous avons également reporté les scores des trois meilleurs systèmes lors du challenge.

Méthodes	F-score Test (%)	F-score (%) Eval
[JiaKai, 2018]	25,9	32,4
MMM optimisé	-	**32,0**
[Liu et al,, 2018]	51,6	29,9
[Kothinti et al, 2018]	30,1	22,4
MMM non-optimisé	24,6	**16,6**
Baseline optimisé	-	**14,1**
Baseline [Serizel et al., 2018]	14,1	10,8

Table 3.2 – Classement par F-score décroissant sur le jeu d'évaluation, après avoir appliqué le post-traitement sur notre meilleur système (MMM optimisé) et le système de référence (Baseline optimisé) fourni par le défi.

Après optimisation, le F-score de la Baseline (notre version) passe de 12,6 % à 14,1 % (CDH), et pour MMM, de 21,1 % à 32,0 % (CDH). Les participants classés premiers [JiaKai, 2018] et deuxième [Liu et al., 2018] ont obtenu un F-score de 32,4 % et 29,9 %, respectivement, ce qui placerait notre système MMM au deuxième rang si nous avions utilisé le seuillage optimisé présenté ici lors du challenge.

3.4 Participation à la tâche 4 de DCASE 2019

Nous avons participé à nouveau à la tâche 4 de DCASE en 2019, avec plusieurs soumissions. Mon directeur de thèse a entraîné un CRNN de même architecture que celui de l'édition 2018, avec une fonction objectif avec deux termes : l'un pour la classification, l'autre pour la segmentation lorsque des annotations fortes sont disponibles (données synthétiques). Au lieu d'utiliser deux modèles comme en 2018, cette fois-ci le même modèle fait les deux tâches. Le tableau 3.3 montre les gains obtenus sans optimisation et avec trois méthodes différentes. En utilisant l'algorithme par seuillage hystérésis dépendant de la classe (CDH), nous obtenons notre meilleur résultat (F-score de 39,7 % sur l'ensemble d'évaluation) et avons atteint le quatrième rang sur 19 équipes [Cances et al., 2019].

	F-score
Sans optimisation	15.9%
CI Absolute	29,1 %
CD Absolute	36,9 %
CD Hystérésis	**39,9 %**

Table 3.3 – F-score obtenus sans et avec optimisation sur l'ensemble de test DCASE 2019 tâche 4.

3.5 Discussion

Pour la tâche de segmentation d'évènements sonores, le post-traitement des prédictions est souvent négligé, et nous n'avons pas trouvé dans la littérature d'analyse systématique des différentes méthodes et de son impact. Pour cette raison, suite à notre participation au challenge en 2018, nous avons travaillé sur le sujet et proposé des solutions de lissage et de seuillage. Trois algorithmes de segmentation à appliquer directement sur les prédictions temporelles produites par un système ont été proposés. Ces méthodes utilisent des paramètres qui peuvent être définis soit manuellement, soit via un calcul statistique, soit via un algorithme d'optimisation. Chacun de ces paramètres peut aussi être déterminé de manière globale, c'est-à-dire que les mêmes valeurs sont utilisées pour toutes les catégories, ou de manière dépendante des classes avec des valeurs spécifiques pour chaque type d'évènement.

Les méthodes présentées montrent, au vu des résultats, le grand impact que le post-traitement peut avoir sur la performance finale. Les méthodes basées sur les statistiques ne nécessitent pas d'optimisation, ce qui les rend appropriées pour un aperçu rapide des résultats qui peuvent être obtenus. Elles sont indépendantes du modèle, faciles à mettre en œuvre, rapides à calculer et peuvent produire de meilleurs résultats qu'une recherche sur grille grossière des paramètres de lissage et de seuillage.

Les méthodes paramétriques sont néanmoins meilleures. Notre meilleur modèle montre une amélioration de 28,6 points (187,0 % relatif) en utilisant un seuil absolu dépendant de la classe. Les méthodes dépendantes de la classe ne donnent pas seulement de meilleurs résultats, mais aussi une plus grande robustesse lors du passage de l'ensemble de test à l'ensemble d'évaluation. La méthode par hystérésys nous a permis d'atteindre un bon classement lors de l'édition 2019 de DCASE.

La recherche systématique des meilleurs paramètres pour une méthode donnée n'est pas nécessairement quelque chose de trivial. De plus, le post-traitement ne se limite pas seulement à la segmentation des prédictions produites par le système, mais passe aussi par des étapes de lissage avant et après cette segmentation. De la même façon qu'il existe plusieurs méthodes pour découper des prédictions, il existe plusieurs façons de les lisser.

L'ensemble de ce travail a permis de produire une boite à outils, nommé AESEG pour « Audio Event SEGmentation » [1], qui fournit un moyen rapide d'établir une chaîne de post-traitement accomplissant les différentes étapes nécessaires et permettant d'utiliser les différents algorithmes de segmentation et d'optimisation décrit dans ce chapitre.

Il serait intéressant de regarder l'impact du post-traitement sur d'autres jeux de données, avec des tailles et des classes différentes pour confirmer l'efficacité des méthodes présentées ici. Il existe d'autres options pour effectuer l'optimisation de paramètres qui pourraient être mentionnées : par exemple une approche descente de gradient à l'aide de gradients approchés

1. https ://github.com/aeseg/aeseg

pour contourner le problème de la non-dérivabilité des fonctions de seuillage [Pellegrini and Masquelier, 2021].

Deuxième partie

Classification d'évènements sonores semi-supervisée

Chapitre 4

Co-entraînement semi-supervisé

Dans ce chapitre, j'étudie en détail une méthode d'apprentissage semi-supervisé qui fut testée avec succès pour une tâche de classification d'images. Nommée Deep Co-Training (DCT)[Qiao et al., 2018], cette approche a prouvé son efficacité sur une tâche de classification sur les jeux de données d'images les plus populaires tels que ImageNet, Canadian Institute For Advanced Research - 10 (CIFAR-10) ou encore The Street View House Numbers (SVHN) [Netzer et al., 2011]. Avant d'effectuer les adaptations requises pour son application à la classification d'évènements sonores, un travail de reproduction des travaux de [Qiao et al., 2018] a été nécessaire dans un premier temps. Nous testerons notre implémentation sur CIFAR-10. C'est seulement après nous être assurés que nous travaillons à partir d'une implémentation correcte, que l'on a pu analyser, modifier et tester le DCT sur le jeu de données audio UrbanSound 8K Dataset (UBS8K).

Après son adaptation, et au vu des résultats modestes obtenus, nous testons deux modifications avec pour objectifs d'améliorer les performances du DCT et de réduire sa complexité. La première consiste à augmenter les données pour artificiellement augmenter le ratio supervisé / non-supervisé des mini-lots, et ainsi améliorer les performances du système. La seconde remplace un des mécanismes principaux du DCT, à savoir la génération d'exemples adversaires, par ces mêmes augmentations. Les avantages sont doubles, car elles sont généralement plus rapides à calculer que les exemples adversaires qui ne sont alors plus nécessaires, et cela permet de diminuer la complexité de l'algorithme.

4.1 Le Deep Co-Training (DCT)

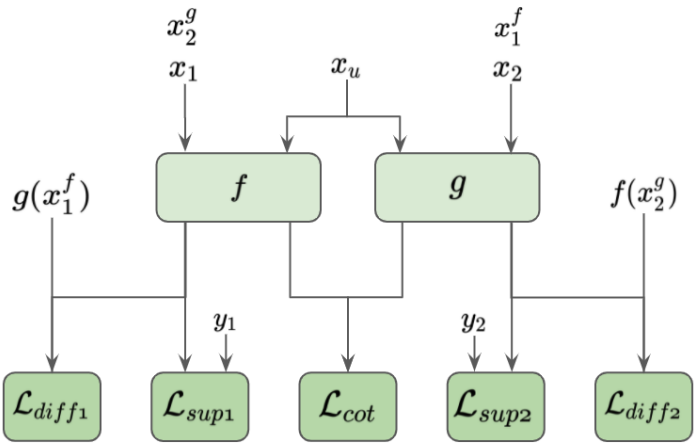

Figure 4.1 – Schéma DCT. Chaque modèle f et g est entraîné sur ses propres échantillons étiquetés x_i, les échantillons non étiquetés x_u et les exemples adversaires générés par l'autre modèle. Le modèle f fait des prédictions sur x_1 et x_2^g, et g sur x_2 et x_1^f.

Le DCT a été proposé par Qiao et al. [2018]. Il est basé sur le Co-Training (CT), le cadre générique bien connu du Semi Supervised Learning (SSL) initié par Blum and Mitchell [1998]. L'idée principale du Co-Training est basée sur l'hypothèse que deux vues indépendantes sur un ensemble de données d'entraînement sont disponibles pour entraîner deux modèles séparément. Idéalement, les deux points de vue sont conditionnellement indépendants compte tenu de la classe. Les deux modèles sont ensuite utilisés pour faire des prédictions sur le sous-ensemble de données non étiquetées. Les prédictions les plus fiables sont sélectionnées et ajoutées au sous-ensemble étiqueté. Ce processus est itératif.

DCT est une adaptation du CT pour l'apprentissage profond. Au lieu de s'appuyer sur des vues différentes des données, le DCT utilise des exemples adversaires pour garantir l'indépendance de la « vue » présentée aux modèles. La deuxième différence est que l'ensemble des données non étiquetées est utilisé pendant l'entraînement. Chaque lot est composé d'une partie supervisée et d'une autre non supervisée. Ainsi, les exemples non annotés sont directement utilisés, et l'aspect itératif de l'algorithme est supprimé.

4.1.1 Fonctions d'entraînement

Soient $\mathcal{S}$ et $\mathcal{U}$ les sous-ensembles de données étiquetées et non étiquetées, respectivement, et soit f et g les deux réseaux neuronaux censés collaborer. La fonction de coût DCT est composée de trois termes, comme le montre l'équation 4.1. Ils correspondent à des fonctions de coût estimées soit sur $\mathcal{S}$, soit sur $\mathcal{U}$, soit sur les deux. Pendant l'apprentissage, chaque mini-lot est composé d'échantillons étiquetés et non étiquetés dans une proportion fixe.

De plus, les exemples étiquetés donnés à chacun des deux modèles sont échantillonnés différemment, on parlera alors de x_1 et x_2.

$$\mathcal{L} = \mathcal{L}_{\text{sup}} + \lambda_{\text{cot}}\mathcal{L}_{\text{cot}} + \lambda_{\text{diff}}\mathcal{L}_{\text{diff}} \tag{4.1}$$

Le premier terme, $\mathcal{L}_{\text{sup}}$, donné dans l'équation 4.2, correspond à la fonction de coût de classification supervisée pour les deux modèles f et g, estimée sur des exemples x_1 et x_2 échantillonnés à partir de $\mathcal{S}$. Dans notre cas, nous utilisons l'entropie croisée (CE), la fonction de coût standard utilisée dans les tâches de classification avec des classes mutuellement exclusives.

$$\begin{aligned} f(x_1) &= P(y|x_1;\theta_f) \\ g(x_2) &= P(y|x_2;\theta_f) \\ \mathcal{L}_{\text{sup}} &= \text{CE}\left(f(x_1), y_1\right) + \text{CE}\left(g(x_2), y_2\right) \qquad (4.2) \\ & \qquad (4.3) \end{aligned}$$

Le mécanisme de maintien de la consistance est utilisé, car les deux classifieurs sont censés fournir des prédictions cohérentes et similaires sur les données étiquetées x_s et non étiquetées x_u. Pour encourager ce comportement, la divergence de Jensen-Shannon (JS) entre les deux ensembles de prédiction est minimisée sur les exemples x_u échantillonnés dans le sous-ensemble non étiqueté $\mathcal{U}$ uniquement. En effet, il n'est pas nécessaire de minimiser cette divergence également sur $\mathcal{S}$ puisque $\mathcal{L}_{\text{sup}}$ encourage déjà les deux modèles à avoir des prédictions similaires sur $\mathcal{S}$. L'équation 4.4 donne l'expression analytique de JS, avec $\mathcal{H}$ représentant l'entropie.

$$\begin{aligned} \mathcal{H}(x) &= -x \cdot log(x) \\ \mathcal{L}_{\text{cot}} &= \mathcal{H}\Big(\frac{1}{2}\big(f(x_u) + g(x_u)\big)\Big) - \frac{1}{2}\Big(\mathcal{H}\big(f(x_u)\big) + \mathcal{H}\big(g(x_u)\big)\Big) \end{aligned} \tag{4.4}$$

Pour que le DCT fonctionne, les deux modèles doivent être complémentaires : sur un sous-ensemble différent de $\mathcal{S} \cup \mathcal{U}$, les exemples incorrectement prédits par un modèle doivent être correctement classés par l'autre modèle [Krogel and Scheffer, 2004]. Cela peut être réalisé en apprentissage profond en générant des exemples adversaires avec un premier modèle et en entraînant un second modèle à être résistant à ces exemples adversaires. Pour ce faire, le terme $\mathcal{L}_{\text{diff}}$ (Eq. 4.5) est la somme des pertes d'entropie croisée entre les prédictions $f(x_1)$ et $g(x_1')$, où x_1 est échantillonné dans $S \cup U$ et x_1^f est l'exemple adversaire à x_1 généré avec le modèle f. Le second terme est symétrique pour le modèle g.

$$\mathcal{L}_{\text{diff}} = \text{CE}\left(f(x_1), g(x_1^f)\right) + \text{CE}\left(g(x_2), f(x_2^g)\right) \tag{4.5}$$

4.1.2 Détails sur la phase d'entraînement

L'entraînement du DCT utilise les trois fonctions de coûts détaillées ci-dessus, un mécanisme d'échauffement pour celles utilisant les fichiers non supervisés et la génération d'exemples adversaires pour chaque modèle. Ces deux mécanismes sont détaillés dans la section 4.1.2.

— Chaque mini-lot d'entraînement est composé d'échantillons annotés et non annotés dans les mêmes proportions que le jeu de données. Les données supervisées sont fournies dans un ordre différent pour chaque modèle, car cela permet d'augmenter la différence entre eux. Les données non-annotées sont quant à elles identiques pour les deux modèles.

— Les exemples adversaires x_1^f et x_2^g sont générés pour chaque modèle en utilisant la Fast Gradient Signed Method (FGSM) [Goodfellow et al., 2015] à chaque itération pendant l'entraînement. Renouveler ainsi les exemples adversaires empêches les modèles de devenir robustes à ces exemples adversaires et garantit leur efficacités..

— Les trois fonctions de coût sont calculées individuellement et linéairement combinées dans 4.1. Deux hyperparamètres, λ_{cot} et λ_{diff}, sont introduits par cette combinaison. Leur objectif est de pondérer l'influence de ces fonctions pendant l'entraînement. Leur valeur n'est pas fixe et évoluera jusqu'à une valeur maximale $\lambda_{\text{cot,max}}$ et $\lambda_{\text{diff,max}}$ pendant la phase d'échauffement.

Échauffement (*warmup*)

Si nous commençons à entraîner le système avec les trois fonctions de coût en même temps, le modèle ne commencera jamais à converger et la prédiction restera plutôt aléatoire. Le mécanisme d'échauffement est donc nécessaire pour atténuer l'impact des différentes fonctions de coût et faciliter la convergence à un stade précoce de l'apprentissage. Les deux composantes non-supervisées sont pondérées par les ratios lambda λ qui sont fixés à 0 au début de l'apprentissage et augmenteront lentement jusqu'à leur valeur maximale après un nombre fixe d'itérations. La croissance de ces ratios est définie par l'équation 4.6 :

$$\lambda = \lambda_{\text{max}} \times \left(1 - e^{-5\times(1-(t/\text{wl}))}\right)^2) \tag{4.6}$$

Génération d'exemples adversaires

En considérant x comme étant l'enregistrement audio original, ici le mot *Backward* est prédit. Nous avons la matrice ∇_x, obtenue grâce à la propagation du gradient d'erreur depuis la sortie du modèle jusqu'à la couche d'entrée. Cette matrice est alors utilisée pour perturber l'exemple original très légèrement grâce au ratio $\mathcal{E}$. Le résultat de cette combinaison linéaire devrait être alors quasiment identique à l'oeil nul, mais très différent pour le modèle qui est lui, beaucoup plus sensible. Le modèle est alors trompé par l'exemple adversaire et ne prédit pas la bonne classe, ici le mot *No*. La figure 4.2 illustre un exemple adversaire généré à partir de l'enregistrement du mot *backward*.

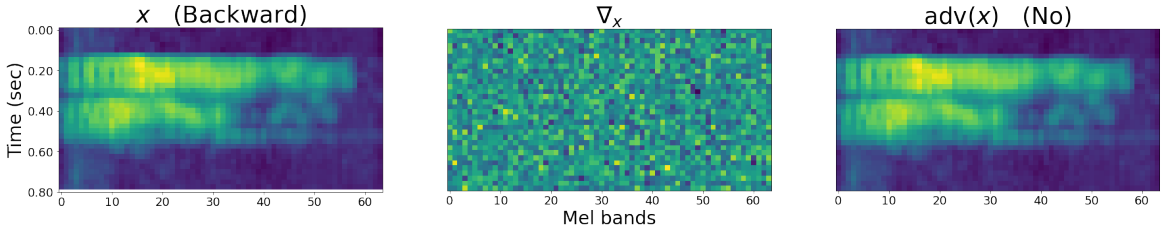

Figure 4.2 – Illustration d'un exemple adversaire, le mot *Backward* devient *No* «

Pour la génération d'exemple adversaires, nous utilisons la FGSM [Goodfellow et al., 2015], comme dans les travaux de [Qiao et al., 2018]. Elle est définie ci-dessous : Soit x un exemple provenant de $\mathcal{S} \cup \mathcal{U}$ et adv(x) son exemple adversaire, ∇_x le gradient d'erreur obtenu grâce à la fonction de coût du système θ, $\mathcal{E}$ un coefficient multiplicateur très faible et, $P(y \mid x; \theta) = f(x)$ la prédiction y fournie par le modèle pour x.

$$\text{adv}(x) = x + \mathcal{E} \times \text{sign}\Big(\nabla_x f(x)\Big) \tag{4.7}$$

Hyperparamètres

Chaque mécanisme du DCT apporte son lot d'hyperparamètres qui peuvent nécessiter un ajustement pendant une phase de recherche exhaustive (ou *grid search*).

— L'échauffement introduit $\lambda_{\text{cot,max}}$, et $\lambda_{\text{diff,max}}$ qui sont utilisés pour pondérer les effets de la fonction de coût à laquelle ils sont couplés. L'échauffement dure pendant un temps défini wl qui est lui aussi nécessaire d'ajuster.

— La FGSM introduit $\mathcal{E}$ qui permet de régler la force de la génération adversaire.

— Le taux d'apprentissage lr évolue en suivant un schéma spécifique définie par la formule 4.8. Sa valeur de départ peut influencer les résultats finaux de manière significative.

$$\text{lr} = \frac{1}{2}\left(1 + \cos\left((t - 1)\frac{\pi}{e}\right)\right) \tag{4.8}$$

4.2 Le DCT pour la classification d'images

Avant de commencer à modifier l'algorithme pour une tâche d'audio tagging, il est nécessaire de s'assurer que notre implémentation est correcte. Pour cela, nous l'avons testé sur un des jeux de données d'images utilisé par [Qiao et al., 2018] : CIFAR-10.

Cette section fournit la description du jeu de données CIFAR-10, l'architecture du modèle, les valeurs des hyperparamètres pour l'entraînement, et finalement les résultats obtenus.

4.2.1 CIFAR-10

CIFAR-10 est un jeu de données pour la classification d'images de 32x32 pixels qui sont divisées en 10 catégories. Composé de 50 000 images d'entraînement et 10 000 de test, il est très utile pour rapidement tester des systèmes. Il est souvent employé comme référence pour expérimenter de nouvelles approches. Pour l'entraînement nous avons appliqué une combinaison d'augmentations correspondant à une symétrie horizontale et une translation d'au plus 2 pixels.

Nous avons artificiellement modifié CIFAR-10 pour le rendre utilisable dans un contexte semi-supervisé. Pour cela, les données d'entraînement sont aléatoirement séparées en deux sous-ensembles. Un premier supervisé, $\mathcal{S}$ contenant 4000 enregistrements de l'ensemble d'entraînement, et un second non-supervisé, $\mathcal{U}$ contenant les 46 000 restants.

4.2.2 Modèle

Le modèle utilisé dans les travaux de [Laine and Aila, 2017] et [Qiao et al., 2018] est décrit dans la figure 4.3.

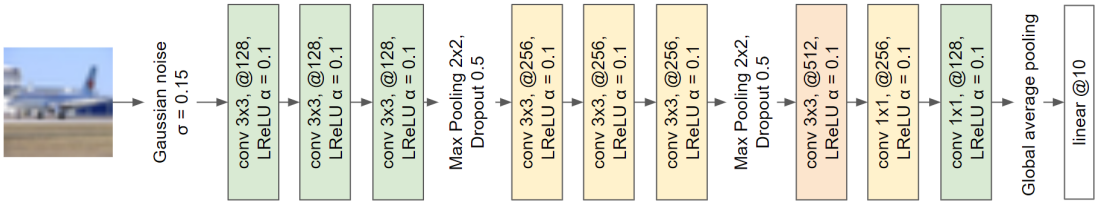

Figure 4.3 – CNN du DCT sur CIFAR-10

Il s'agit d'un modèle relativement gros avec 3,12 millions de paramètres entraînables. Basé sur une architecture de Convolutional Neural Network (CNN), il est composé de trois groupes de 3 couches convolutives utilisant une fonction d'activation Leaky Rectified Linear Activation (LReLU). Après le premier et le second groupe de convolution, on retrouve un *Max pooling* 2x2 suivi d'un *dropout* à 50 %. Dans le premier groupe, chaque couche convolutive possède 128 filtres, dans le second 256 filtres et dans le troisième, 512, 256,

et 128 filtres. Toutes les couches convolutives ont un champ réceptif 3x3, sauf les deux dernières qui sont à 1x1. La couche finale est une Fully Connected Network (FC) permettant de faire la classification.

4.2.3 Entraînement

Pour nous assurer que nos résultats sont similaires avec ceux de [Qiao et al., 2018], nous utilisons les mêmes paramètres d'entraînement qu'eux.

Les hyperparamètres $\lambda_{\text{cot,max}}$ et $\lambda_{\text{diff,max}}$ sont fixés à 10 et 0,5 respectivement. Pour le FGSM, $\mathcal{E}$ est fixée à $0,02$. Pour l'entraînement, nous utilisons l'optimiseur Stochastic Gradient Descent (SGD) avec un *momentum* de 0,9 et une décroissances des poids de 0,0001 (*weight decay*). Le système est entraîné sur 600 époques e avec des mini-lots bs de taille 100. L'échauffement se fait sur les 80 premières époques et le taux d'apprentissage lr initialisé à 0,05 diminue tout au long des 600 époques de l'entraînement en suivant la fonction 4.8.

4.2.4 Résultats sur CIFAR-10

Le tableau 4.1 montre les résultats du DCT comparé à d'autres approches semi-supervisées telles que Mean-Teacher [Tarvainen and Valpola, 2018] (décrit en détail dans le chapitre 5.1.1) ou le Π modèle [Laine and Aila, 2017].

	1000 labels	2000 labels	4000 labels	50 000
Supervisé	53,57 ± 1,21	66,06 ± 0,73	64,44 ± 1,59	92,67 ± 0,04
Pseudo labelling	-	-	83,91 ± 0,28	-
Γ model	-	-	79,60 ± 0,47	-
Π model	72,64 ± 1,20	81,98 ± 0,60	86,80 ± 0,27	93,94 ± 0,11
Temporal Ensembling	-	-	87,84 ± 0,24	-
Mean Teacher	78,45 ± 1,48	84,27 ± 0,31	87,69 ± 0,28	94,06 ± 0,15
Deep Co-Training	-	-	90,97 ± 0,18	

Table 4.1 – Précision et écart-type de différents systèmes semi-supervisés pour CIFAR-10 en utilisant 1000, 2000, 4000 et 50 000 labels. Comparaison avec d'autres résultats extraits de la littérature : [Lee, 2013; Qiao et al., 2018; Rasmus et al., 2015]

Comparé aux méthodes SSL plus anciennes, le DCT présente les meilleures performances avec une précision de 90,97 %. C'est mieux de 3 points que le Mean-Teacher (MT) et seulement 1,7 point de moins qu'un entraînement supervisé. Le DCT se rapproche de l'état de l'art supervisé en n'utilisant que 4000 enregistrements annotés (12.5 % des labels).

4.3 Application à l'audio

Une fois que nous avons pu reproduire les résultats de [Qiao et al., 2018] sur CIFAR-10, nous avons appliqué le DCT sur le jeu de données audio populaire, UrbanSound8K (UBS8K). Cette section fournit la description de ce dernier, l'architecture du modèle, la valeur des hyperparamètres pour l'entraînement, et finalement les résultats obtenus.

4.3.1 UrbanSound8k

UBS8K [Salamon et al., 2014] est composé de 8742 enregistrements audio d'une durée comprise entre 1 et 4 secondes, séparés en 10 catégories plus ou moins équilibrées. Les catégories *car horn* et *gun shot* ont seulement 429 et 374 exemples respectivement, tandis que toutes les autres en ont 1000. UBS8K est découpé en 10 dossiers qui doivent être utilisés pour faire une validation croisée. Cette procédure permet une meilleure évaluation des systèmes et aussi la comparaison avec d'autres recherches. Les enregistrements qui ne durent pas quatre secondes sont complétés avec du silence pour homogénéiser les durées. La fréquence d'échantillonnage étant très variable, tous les enregistrements sont échantillonnés à 22 kHz et encodés en 16 bits avant d'être convertis en 64×173 log-mel spectrogramme.

Comme pour CIFAR-10, nous avons aléatoirement séparé UBS8K en deux ensembles. Le premier, supervisé $\mathcal{S}$, contient 10 % des enregistrements, tandis que le second, non supervisé $\mathcal{U}$ contient les 90 % restant.

4.3.2 Modèle

Le modèle est basé sur la même architecture que celle proposée dans [Salamon and Bello, 2017]. Il s'agit d'un petit modèle d'environ 143 000 paramètres, permettant de réaliser les expériences rapidement tout en atteignant l'état de l'art pour un entraînement entièrement supervisé.

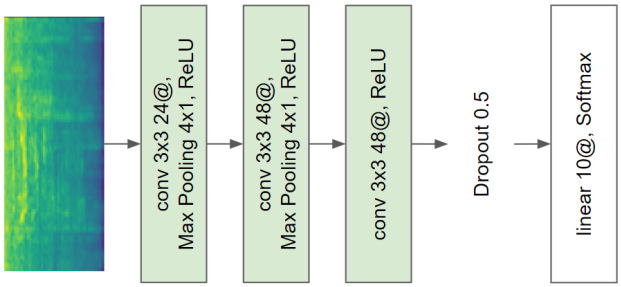

Figure 4.4 – CNN pour DCT sur UBS8K

Son architecture, présentée dans la figure 4.4, se compose de trois couches convolutives de 24, 48 et 48 filtres et un champ réceptif 3x3, directement suivies par un *max pooling*

4x1 (sauf pour la dernière couche) et une activation Rectified Linear Unit (ReLU). Pour finir, un *dropout* à 50 %, suivi d'une couche FC et une activation *Softmax* se charge de la classification.

4.3.3 Entraînement

Les hyperparamètres $\lambda_{\text{cot,max}}$ et $\lambda_{\text{diff,max}}$ sont fixés à 1 et 0,5 respectivement. Pour le FGSM, $\mathcal{E}$ est fixée à $0,02$. Pour l'entraînement, nous utilisons l'optimiseur SGD avec un *momentum* de 0,9 et une décroissances des poids de 0,0001 (*weight decay*). Le système est entraîné sur 300 époques e avec des mini-lots bs de taille 64. L'échauffement se fait sur les 160 premières époques et le taux d'apprentissage lr initialisé à 0,05 diminue tout au long des 300 époques d'entraînement en suivant la fonction 4.8

4.3.4 Résultats sur UrbanSound8k

Dans le tableau ci-dessous, nous comparons une approche entièrement supervisée au DCT sur le jeu de données UBS8K. Plus de comparaisons avec d'autres méthodes de SSL sont disponibles dans le chapitre 5. L'approche nommée *Augmented Deep Co-Training* est décrite dans la section 4.5.2.

	10 % labels	50 % labels	100 % labels
[Salamon and Bello, 2017]	-	-	73 / 79
Supervisé	47,3 ± 4,1	67,1 ± 3,9	75,6 ± 4,8
Deep Co-Training	55,4 ± 4,6	71,5 ± 2,7	-
Augmented Deep Co-Training	59,7 ± 5,1	-	-

Table 4.2 – Précision et écart-type de DCT pour UBS8K en utilisant 10 %, 50 %, et 100 % des labels. Comparaison avec un entraînement supervisé classique

Le modèle CNN que nous employons permet d'atteindre l'état de l'art pour un entraînement supervisé qui utilise 100 % des annotations. Avec un score de 75.6 %, il se situe entre les 73 % (entraînement sans augmentation) et 79 % (entraînement avec augmentation) obtenus dans les travaux de Salamon and Bello [2017].

Les résultats sont plus mitigés sur UBS8K mais tout de même prometteurs. DCT, avec seulement 10 % des données annotées, atteint une précision de 55,4 % contre 47,3 % pour un entraînement supervisé. Cela représente tout de même un gain de 17 %. Lorsqu'on augmente la quantité d'exemples annotés à 50 %, l'écart avec un entraînement 100 % supervisé est plus faible de seulement 4,1 points.

4.4 Analyse et hypothèses

Les résultats ont montré que les gains apportés par le DCT sur UBS8K n'étaient pas aussi importants que sur CIFAR-10. Cette différence de performance peut-elle être uniquement expliquée par une différence au niveau des données ? On peut alors se poser les questions suivantes :

— Dans un contexte semi-supervisé, seulement 87 fichiers sont annotés pour UBS8K contre 400 pour CIFAR-10. Les deux jeux de données étant séparés en 10 catégories, cela représente moins de 10 enregistrements audio supervisés pour UBS8K et 40 pour CIFAR-10. Est-ce que cette différence dans la quantité de fichiers annotés disponible peut influencer à ce point les résultats du DCT ? On cherchera alors à virtuellement augmenter le nombre d'exemples supervisés dans la section 4.5.

— Est-ce que la génération d'exemples adversaires fonctionne aussi bien sur des enregistrements audio que sur des images ? Peut-on remplacer ces exemples adversaires par quelque chose de plus simple à mettre en place et peut-être de manière plus performante ? Cette étude est décrite dans la section 4.6.

4.5 Équilibrer les minis lot d'entraînement

L'apprentissage avec des enregistrements non étiquetés est possible grâce à la présence d'un nombre minimum d'exemples étiquetés. Certaines approches comme le pseudo-labeling vont entraîner un modèle sur les données annotées pour ensuite prédire les labels des exemples non-annotés. Ces derniers seront ensuite utilisés pour un nouvel entraînement supervisé. D'autres approches, dont le DCT fait partie, utilise à la fois les exemples annotés et non annotés pendant l'entraînement. Il est alors nécessaire d'avoir un minimum d'exemples annotés dans chaque mini-lot pendant l'apprentissage.

Les deux expériences permettent d'observer les performances du DCT lorsqu'on modifie l'équilibre supervisé / non-supervisé dans les minis lot d'entraînement, et cela sans augmenter le nombre d'exemples supervisés utilisés. Dans un premier temps, les exemples supervisés seront dupliqués, et dans un second temps on utilisera des augmentations pour artificiellement modifier les exemples dupliqués et réduire le sur-entraînement.

Attention, au vu du grand nombre de modèles qui ont dû être entraînés, les expériences ont été réalisées en utilisant un sous-ensemble de UBS8K. Ce dernier, composé de seulement 10 % des enregistrements totaux permet de réaliser les expériences beaucoup plus rapidement tout en permettant de faire des interprétations sur l'impact de chacune des configurations testées ci-dessous.

4.5.1 Simple duplication

Nous allons virtuellement augmenter le nombre d'exemples supervisés disponibles simplement en les dupliquant. Ce faisant, le ratio d'exemple supervisé / non-supervisé **par mini-lot** changera. La figure 4.5 montre le résultat de cette expérience avec des ratios d'exemples étiquetés par minibatch de 10, 15, 20, 30, 40, 50 et 75 %.

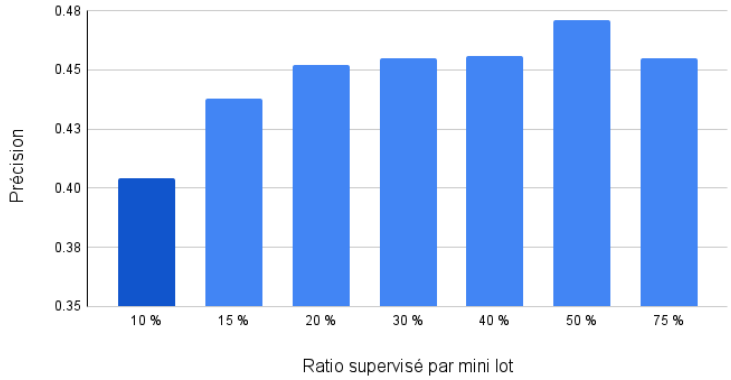

Figure 4.5 – Évolution de la précision au fur et à mesure que le ratio d'exemples étiquetés par minibatch augmente. Expérience réalisée sur le jeu de données sous-échantillonné (10 %)

L'amélioration des performances est significative et atteint un plateau à environ 45 % de précision (voir Figure 4.5). Un modèle entraîné avec 50 % d'exemples supervisés par mini-lot est jusqu'à 6,7 points plus efficaces que lorsque la distribution des échantillons étiquetés et non étiquetés par mini-lot est différente du ratio par défaut de 10 %. D'autre part, les exemples supervisés sont dupliqués cinq fois, et le sur-apprentissage est inévitable. Pour palier a ce problème, nous utilisons l'augmentation de données.

4.5.2 L'augmentation de données contre le sur-apprentissage

On peut voir que le simple fait de dupliquer les exemples supervisés dans le but d'équilibrer les mini-lots permet déjà d'augmenter les résultats de plus de 6 points. Cependant, on peut tout à fait se demander si cela n'entraîne pas un problème de sur-apprentissage. Nous pouvons mitiger ce problème en utilisant des augmentations pour artificiellement créer de nouveaux exemples inédits. Les différentes augmentations que nous utilisons sont décrites ci-dessous :

1. **Pitch shift** : Consiste à augmenter ou à diminuer la fréquence du signal audio tout en gardant la durée inchangée. Le décalage est fait en demi-tons et est choisi aléatoirement dans un intervalle qui dépend de la variante faible ou forte. Cette augmentation est illustrée par l'image 4.6.

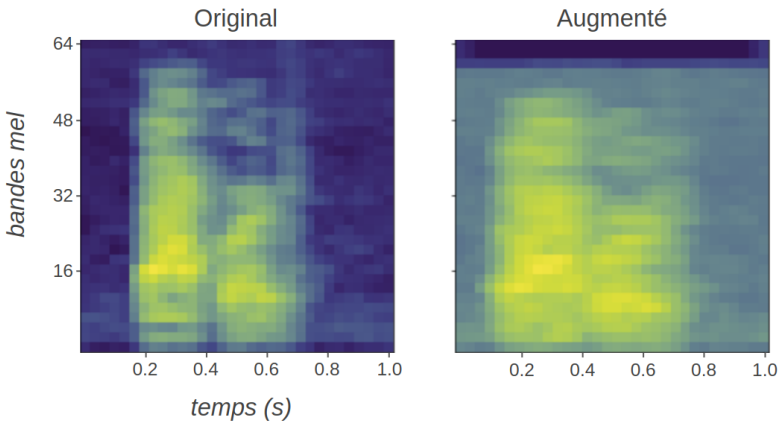

Figure 4.6 – Impact de l'augmentation *Pitch Shift* sur un enregistrement audio issu de Google Speech Command Dataset (GSC). À droite, l'enregistrement original (*Backward*), à gauche le même enregistrement augmenté

2. **Noise** : Cette augmentation ajoute un signal de bruit supplémentaire au signal audio d'origine. Le bruit est échantillonné à partir d'une distribution normale dont les paramètres sont définis de telle sorte que le Signal to Noise Ration (SNR) visé soit obtenu. Cette augmentation est illustrée par l'image 4.7 et le calcul du SNR est défini comme suit :

$$\begin{aligned} \mathcal{P}_{\text{signal,dB}} &= 10\log_{10}(\mathcal{P}_{\text{signal}}) \\ \mathcal{P}_{\text{noise,dB}} &= 10\log_{10}(\mathcal{P}_{\text{noise}}) \\ \text{SNR} &= \frac{\mathcal{P}_{\text{signal,dB}}}{\mathcal{P}_{\text{noise,dB}}} \end{aligned} \tag{4.9}$$

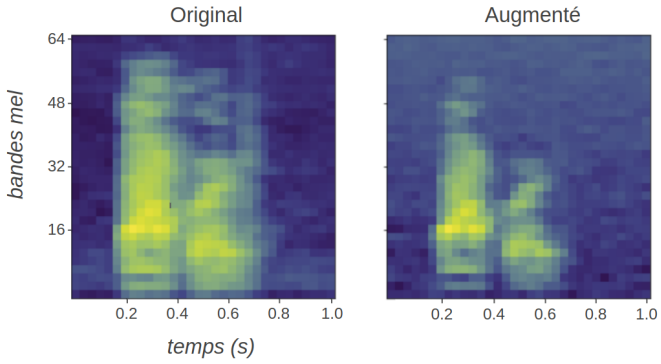

Figure 4.7 – Impact de l'augmentation « Noise » sur un enregistrement audio issu de GSC. À droite, l'enregistrement audio original (*Backward*), à gauche l'enregistrement augmenté avec un bruit et un SNR de 20dB

3. **SpecAugment [Park et al., 2019]** : Il s'agit d'une méthode simple d'augmentation des données qui est appliquée directement sur le log-spectrogramme du modèle. Elle consiste à masquer des bandes de canaux de fréquence et / ou des bandes d'intervalles de temps, en utilisant la valeur minimale du spectrogramme (-80 dB). Le nombre de

bandes et leur largeur sont choisis aléatoirement dans une plage de valeurs définie par l'utilisateur. Cette augmentation est illustrée par l'image 4.8.

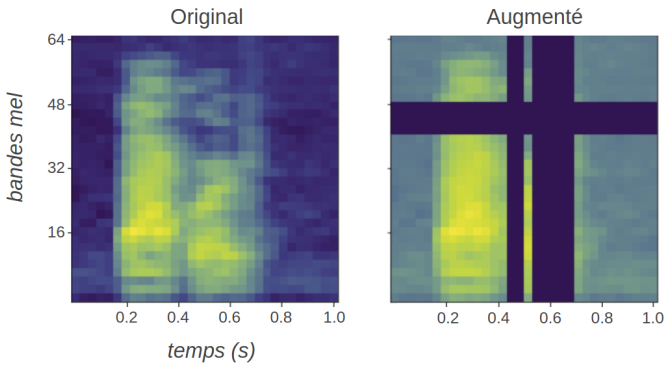

Figure 4.8 – Impact de l'augmentation *SpecAugment* sur un enregistrement audio issu de GSC. À droite, l'enregistrement original (*Backward*), à gauche l'enregistrement augmenté

4. **SpecAugment Stretch** : La séquence audio est découpée en trois bandes verticales qui seront étirées ou compressées de telle sorte à ce que la taille finale soit identique à la taille d'origine. Ce processus est aussi effectué horizontalement. Cette augmentation est illustrée par l'image 4.9.

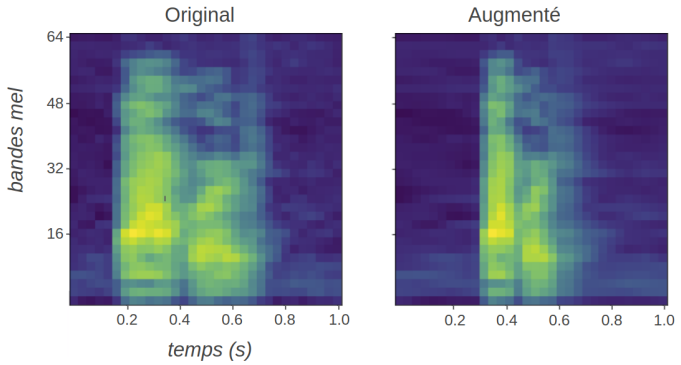

Figure 4.9 – Impact de l'augmentation *SpecAugment* variante « étirement » sur un enregistrement audio issu de GSC. À droite, le fichier original (*Backward*), à gauche le fichier augmenté

Augmentation unique avec 50 % de chance d'être appliquée

Toujours effectué sur un sous-échantillon de UBS8K, les résultats de cette expérience sont présentés dans la figure 4.10.

L'augmentation du nombre d'exemples étiquetés par mini-lots et l'application de l'augmentation sur les exemples dupliqués améliorent les performances du système. Le meilleur score est observé lorsque 40% du mini-lot est supervisé avec une chance sur deux d'appliquer un *Pitch Shift* sur les échantillons étiquetés.

Lorsque les exemples étiquetés sont dupliqués quatre fois, mais que l'augmentation a une chance sur deux d'être appliquée, alors statistiquement, les données originales sont présentées deux fois au système, ce qui, malgré l'utilisation d'augmentation, continue à encourager le

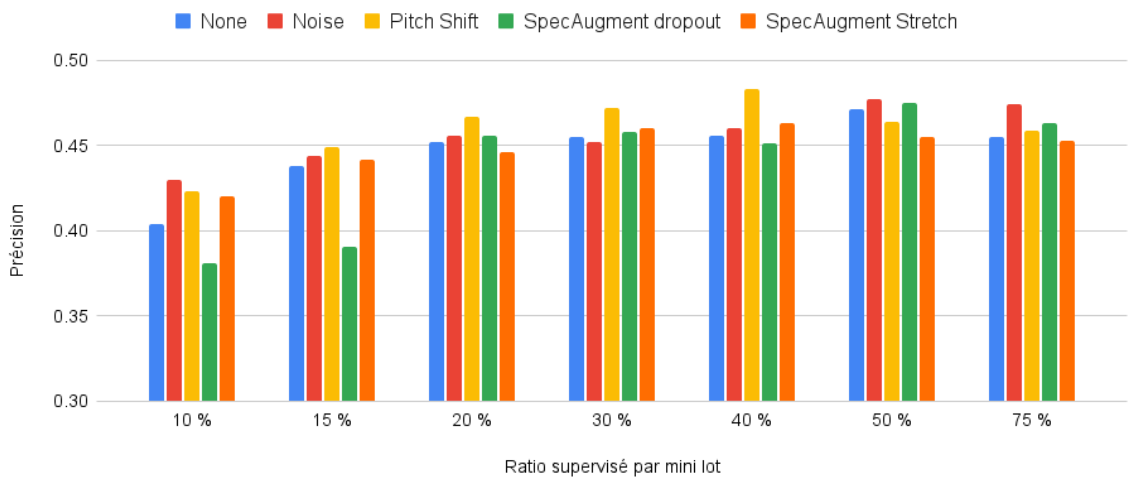

Figure 4.10 – Évolution de la précision lorsque nous combinons l'augmentation du nombre d'exemples étiquetés dans le mini-lot avec certaines augmentations. L'expérience est réalisée sur le jeu de données sous-échantillonné (10%).

sur-entraînement. Ce phénomène est exacerbé lorsque le pourcentage d'échantillons étiquetés dans chaque mini-lot augmente.

Augmenter la chance d'appliquer l'augmentation

Une façon d'atténuer ce comportement est d'augmenter la chance d'appliquer l'augmentation au fur et à mesure que le pourcentage d'échantillons étiquetés augmente dans le minibatch. Le résultat de cette expérience est présenté dans la Figure 4.11. Elle est réalisée en utilisant l'ensemble complet de données.

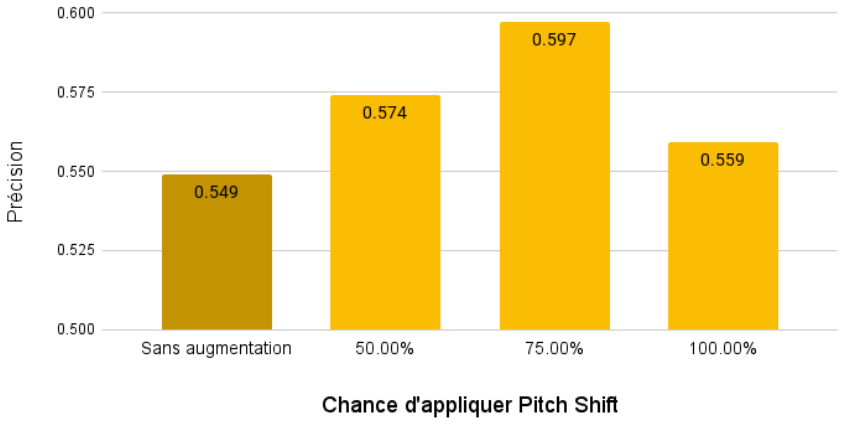

Figure 4.11 – Évolution du score de précision lorsque la probabilité d'appliquer l'augmentation *Pitch Shift* augmente. Ces résultats proviennent de la meilleure configuration et sont calculés sur le jeu de données complet.

Augmenter artificiellement le nombre d'échantillons supervisés pour augmenter l'équilibre supervisé / non-supervisé à 40 % dans les minis lot d'entraînements, tout en appliquant une

augmentation (ici *Pitch-Shift*) avec 75 % de chance d'être appliquée permet d'augmenter le score de DCT sur UBS8K de 4,8 points, soit 7,8 %.

4.5.3 Conclusion

Grâce à ces expériences réalisées sur UBS8K, nous avons pu évaluer l'impact de l'équilibre supervisé/non-supervisé dans les minis lot d'entraînement. En utilisant seulement 10 % des données d'entraînement étiquetées et le reste des données non étiquetées, DCT a obtenu un score de précision de 54,9 %. Grâce à la duplication des exemples supervisés dans les mini lots, et à l'utilisation de *Pitch Shift*, le DCT « augmenté » atteint une précision de 59,7 %. Cela représente une amélioration de 12,4 points (soit 26 %) par rapport à la version supervisée.

4.6 Améliorer la génération d'exemples adversaires

Pour comprendre la différence de performances entre des tâches de classification d'images ou d'audio tagging, nous avons observé l'efficacité de la génération d'exemples adversaires. En effet, la fonction de coût $\mathcal{L}_{\text{diff}}$ (Eq. 4.5) est décrite comme primordiale par [Qiao et al., 2018] pour éviter l'écroulement des modèles sur eux-mêmes. L'écroulement décrit le phénomène pendant lequel l'apprentissage ne fonctionne plus et le modèle recommence à faire des prédictions aléatoires. J'ai pu observer ce phénomène plusieurs fois sur CIFAR-10 ainsi que sur UBS8K. Si l'hyperparamètre λ_{diff}, qui régule l'influence de $\mathcal{L}_{\text{diff}}$ n'est pas correctement défini, alors les modèles s'écroulent. C'est un équilibre difficile à trouver et qui nécessite une recherche exhaustive.

4.6.1 Ratio d'exemples adversaires

L'adaptation de la fonction de coût $\mathcal{L}_{\text{diff}}$ pour l'*audio tagging* est la moins facile à réaliser. Il est probable que la différence de performance observée entre CIFAR-10 et UBS8K, soit majoritairement due à son influence. $\mathcal{L}_{\text{diff}}$ utilise les exemples adversaires et nous concentrerons notre analyse sur ces derniers. Pour cela, nous utiliserons une métrique, nommée « ratio » nous permettant de suivre l'efficacité de la génération d'exemples adversaires tout au long de l'entraînement. Le ratio représente le nombre d'exemples réellement adversaires, c'est-à-dire capables de tromper le modèle, par rapport au nombre total d'enregistrements. Il est calculé à chaque itération.

Pendant un entraînement sur CIFAR-10 et sur UBS8K, on observe l'évolution du ratio et de la précision sur le sous-ensemble $\mathcal{U}$. En effet, $\mathcal{L}_{\text{diff}}$ est calculée grâce aux labels des fichiers non-annotés (Pour rappel, l'absence d'annotation est artificielle, les labels existent, mais on ne les utilise pas pendant l'entraînement).

Sur l'image 4.12 on peut observer que, pour CIFAR-10, le ratio démarre vers 20 %, forme une petite parabole pendant les 50 premières époques, culminant aux alentours de 40

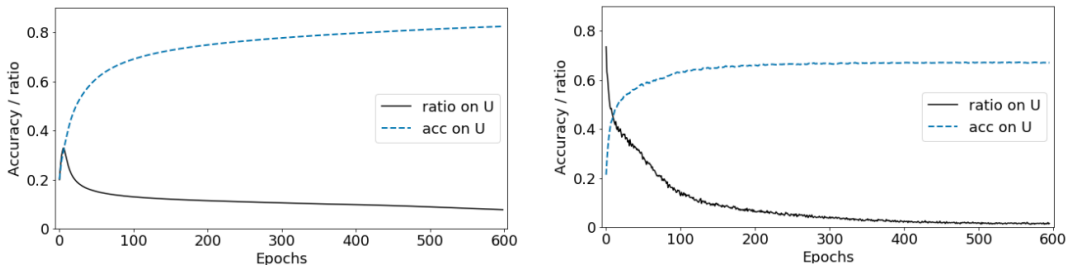

Figure 4.12 – Observation de l'évolution du ratio sur CIFAR-10 (gauche) et sur UBS8K (droite)

% avant de décroître pour atteindre un minimum de 10 % au bout de 600 époques. On peut aussi observer que plus ce dernier est bas, moins la précision semble augmenter.

pour UBS8K, le ratio démarre très haut, avec presque 80 % des exemples vraiment adversaires et décroît rapidement pour finalement atteindre un ratio très proche de 0 à partir de 300 époques. On peut observer que lorsque ce ratio est presque nul, la précision n'augmente plus.

Ces observations nous permettent d'émettre l'hypothèse que l'efficacité de la génération des exemples adversaires joue un rôle important dans l'utilisation des fichiers non-supervisés. Sur UBS8K, les modèles deviennent rapidement robustes à ces adversaires, bloquant l'apprentissage non-supervisé. Le comportement de ces exemples adversaires semble différent entre UBS8K et CIFAR-10.

Pour améliorer le ratio et tenter de réduire le temps de calcul des exemples adversaires, nous avons essayé de remplacer la génération adversaire par des augmentations.

4.6.2 Remplacer les exemples adversaires par des augmentations

Le rôle des exemples adversaires dans DCT est de maintenir une différence entre les deux modèles entrainés parallèlement. Cela leur permet de faire des prédictions en se basant sur des caractéristiques différentes de l'enregistrement audio. Les exemples adversaires destinés au premier modèle sont générés par le deuxième modèle. Idéalement, il faut sélectionner une augmentation différente pour chaque modèle. Ces augmentations doivent être en mesure de remplacer les exemples adversaires, et donc tromper les modèles.

Le choix des augmentations

En plus des quatre augmentations présentées dans l'expérience précédente 4.5.2, un certain nombre d'augmentations supplémentaires ont été testées et sont brièvement décrites ci-dessous.

— *flip up / dowm* : Applique une symétrie horizontale sur le spectrogramme.

— *flip left / right* : Applique une symétrie verticale sur le spectrogramme.

— *Level* : Intensifie ou diminue la puissance du signal par une valeur aléatoire en db comprise dans un intervalle prédéfini.

— *Occlusion* : Appliquée sur le signal, l'occlusion consiste à mettre à 0 une partie de ce signal. La durée et la position sont choisies aléatoirement dans un intervalle prédéfini.

— *Clip* : Consiste à contraindre le signal dans un intervalle prédéfini inférieur aux extrémités. (c.a.d -1, 1.)

— *Signal Noise* : Applique sur le signal un bruit normal pour atteindre un SNR précis.

— *Spectrogram Noise* : Applique un bruit uniforme sur le spectrogramme. Les valeurs sont ensuite multiplées par un entier prédéfini.

— *Time Dropout* : Appliqué sur le spectrogramme, va aléatoirement mettre à zéro un certain pourcentage des trames temporelles.

— *Frequency Dropout* : Appliqué sur le spectrogramme, va aléatoirement mettre à zéro un certain pourcentage des bandes de fréquence.

Chacune de ces perturbations a été testée avec plusieurs combinaisons de paramètres. Pour calculer leur « intensité», et donc déterminer celles qui auront le plus fort ratio (c.-à-d. celles qui tromperont le plus le modèle), nous avons généré toutes ces augmentations et les avons comparées aux enregistrements originels du dossier 1 de la cross-validation de UBS8K. Après les avoir triées dans l'ordre croissant de ratio, on obtient l'image 4.13. Les paramètres de ces augmentations sont listés dans l'annexe A.

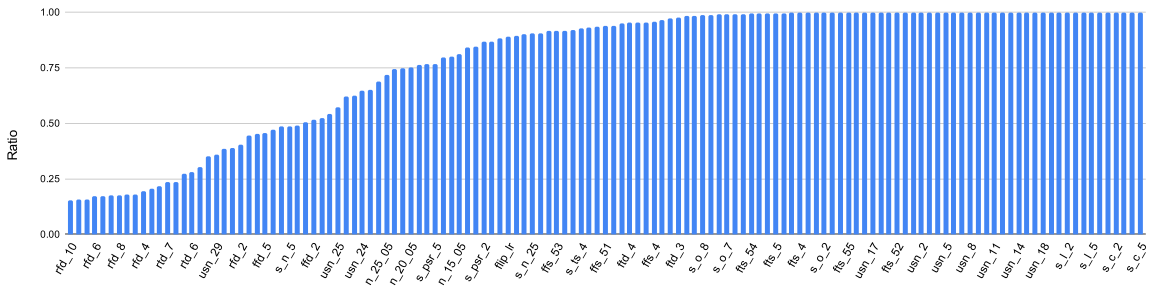

Figure 4.13 – Ratio du nombre d'exemples adversaires sur le nombre total d'exemples. L'ensemble des augmentations est testé sur le dossier 1 de UBS8K.

De l'importance d'une augmentation avec un fort ratio

Choisir des augmentations avec un fort ratio d'exemples adversaires est important pour l'entraînement du DCT. Sans ça, la fonction de coût $\mathcal{L}_{\text{diff}}$ ne joue pas correctement son rôle. Le modèle ne converge pas. La figure 4.14 montre la différence entre un entraînement utilisant une augmentation avec un faible ratio (en orange), et un utilisant une augmentation avec un fort ratio (en gris).

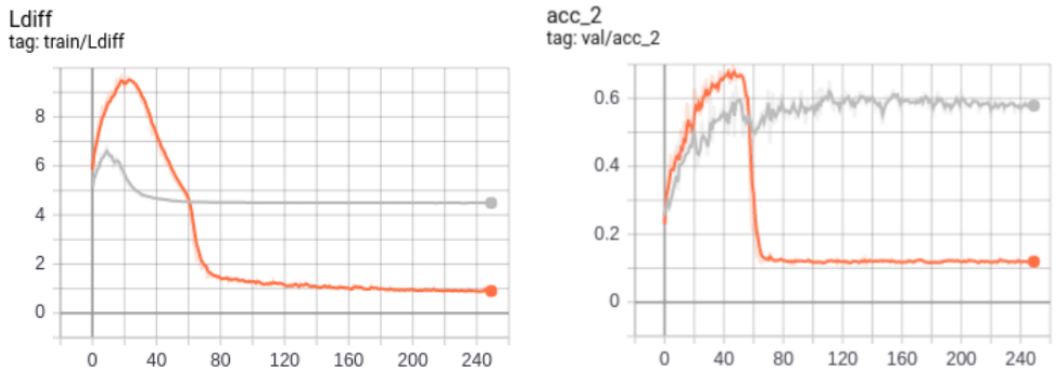

Figure 4.14 – La courbe orange représente le système entraîné avec des augmentations ayant un faible ratio, la courbe grise, celui entraîné avec des augmentations ayant un fort ratio. À gauche, le tracé de la fonction de coût $\mathcal{L}_{\text{diff}}$ et à droite, le tracé de la précision. Entraînement effectué sur 250 époques.

Avec l'augmentation qui ne trompe pas le modèle, la fonction de coût $\mathcal{L}_{\text{diff}}$ converge vers 0 (est correctement minimisé), mais avec des augmentations plus fortes, cette dernière reste stable avec une valeur élevée aux alentours de 4,5. On peut observer que lorsque la valeur de $\mathcal{L}_{\text{diff}}$ est trop faible, le modèle s'écroule et ses prédictions redeviennent aléatoires. Mais lorsqu'elle se stabilise à une valeur élevée, la fonction de coût continuent son rôle et les modèles continue de diverger tout au long de l'entraînement.

Sélection des augmentations

On a vu dans la section précédente qu'il faut correctement sélectionner les augmentations qui remplaceront les exemples adversaires. Pour ce faire, on doit se concentrer sur deux aspects. Le premier, c'est que l'augmentation doit être en mesure de tromper le modèle, cette information nous est donnée par le ratio. Le second c'est que l'augmentation doit le faire en ayant la plus faible empreinte sur le spectrogramme de l'enregistrement original. Nous avons alors besoin de calculer la similarité entre l'exemple augmenté et l'original.

Il existe de nombreuses méthodes pour calculer la similarité entre deux spectrogrammes, mais toutes peuvent donner des résultats très différents. Par exemple, un simple calcul de différence au niveau des trames, n'est pas nécessairement exploitable. En effet, un simple décalage d'une trame audio peut entraîner un score de similitude très mauvais, même si à l'oeil nu, elle est indétectable. Pour cela, on testera plusieurs métriques, comme un score de cohérence, une distance trame à trame, une Root Mean Square Error (RMSE), ou encore une distance d'histogramme.

Les meilleurs résultats ont été obtenus en utilisant la distance d'histogramme entre deux spectrogrammes décrite par la formule 4.10. Les autres méthodes sont décrites dans l'annexe B.

$$\text{similarité} = 1 - \Big(\text{hist}(x) - \text{hist}(y)\Big) \tag{4.10}$$

De plus, dans le DCT, les exemples adversaires destinés au premier modèle sont générés par le deuxième modèle. De la même manière, il faut sélectionner deux augmentations différentes pour reproduire ce fonctionnement. Pour ce faire, on combine le score de similarité de chacune des augmentations avec leur ratio d'exemples adversaires générés. Lorsqu'on affiche cette combinaison dans un plan 2D, avec pour abscisse le score de similarité calculé avec la formule 4.10 et en ordonnée le ratio, on obtient la figure 4.15.

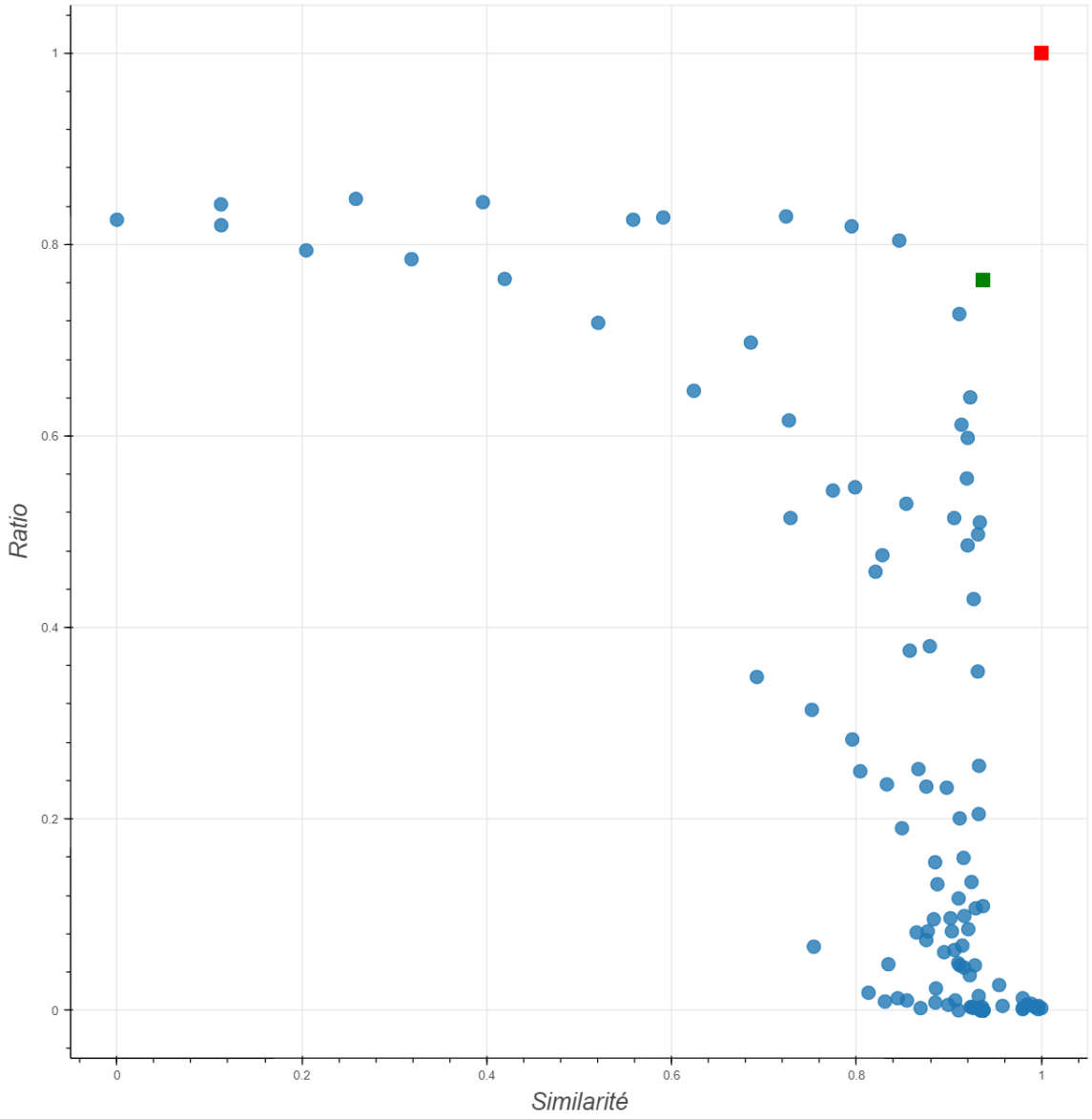

Figure 4.15 – Carte de sélection des augmentations. Le carré rouge représente l'augmentation idéale, celle qui permet d'avoir le meilleur ratio sans modifier le spectrogramme. Le carré vert est celui qui se trouve le plus proche de cet idéal. Dans ce cas, il s'agit des résultats obtenus en utilisant la distance d'histogramme et la meilleure augmentation serait alors une symétrie horizontale.

Pour chacun des calculs de similarité, les deux meilleures augmentations sont utilisées pour remplacer la génération d'exemples adversaires. Les meilleurs résultats ont été obtenus en utilisant la distance d'histogramme, et comme augmentations, une symétrie horizontale pour le premier modèle, et un *Time Dropout* à 50 % pour le deuxième. Cette configuration

a permis d'atteindre un F-score de 65.65 % sur le premier dossier de la validation croisée.

4.6.3 Conclusion

Les résultats visant à remplacer la génération d'exemples adversaires par des augmentations ne sont pas aussi bons que nous l'espérions. Les meilleurs résultats sur le premier dossier de la cross-validation sur UBS8K sont d'environ 66% de F-score, ce qui est inférieur de 4 points au score de 70% avec les exemples adversaires. Le premier dossier étant plus facile que les autres, il n'était pas pertinent de faire une validation croisée complète.

Ce que l'on peut néanmoins conclure :

— Le remplacement des exemples adversaires est possible dans une certaine mesure.

— Le choix des augmentations a un vrai impact sur le comportement des modèles lors de l'apprentissage. Une augmentation forte permet de maintenir l'efficacité de $\mathcal{L}_{\text{diff}}$, tandis qu'avec une augmentation trop faible, on peut observer le même phénomène décrit par [Qiao et al., 2018] sur l'écroulement des modèles (*collapse*).

4.7 Discussion

Le Deep Co-Training DCT est une méthode complexe à mettre en place qui utilise de nombreuses techniques d'apprentissage différentes. Les résultats sur le jeu de données CIFAR-10 nous ont poussé à tester cette approche dans un contexte audio. Pour cela nous avons utilisé un jeu de données populaire UBS8K. Les résultats, comparés avec un apprentissage supervisé sont probants, avec une augmentation de la précision finale de 8,1 points lorsque 10 % des annotations sont utilisées et 4,4 points avec 50 % des annotations.

Cependant, cette amélioration est moindre que celle que l'on pouvait espérer au vu des résultats sur CIFAR-10. Nous avons exploré deux solutions. La première consiste à augmenter le nombre d'exemples supervisés disponibles dans les mini lots en les dupliquant et en utilisant l'augmentation de données. Cette approche nous a permis d'améliorer les performances de 4,3 points supplémentaires avec 10 % des annotations.

La seconde approche s'attaque à la génération d'exemples adversaires. Les analyses des courbes d'entraînement et de la métrique ratio nous ont fait penser que la génération d'adversaires sur des enregistrements audio pouvait être moins performante que pour des images. L'idée est alors de remplacer les exemples adversaires par des augmentations « fortes ». Cependant, nos expériences n'ont pas été satisfaisantes, avec même une perte de performance par rapport à un DCT utilisant la génération d'exemples adversaires.

Dans le chapitre 5 suivant, la méthode DCT est comparée à d'autres méthodes semi-supervisées sur plusieurs datasets. Des différences importantes seront introduites : une métrique différente (taux d'erreur plutôt que F-score), un nouveau modèle (Wide-ResNet

28-2), et surtout l'optimiseur ADAM pour l'entraînement plutôt que SGD dans le présent chapitre.

Chapitre 5

Comparaison de méthodes semi-supervisées pour l'Audio Tagging

En continuation du chapitre précédant, nous comparons DCT à d'autres approches semi-supervisées qui sont à l'origine développées pour des tâches de classification d'images. Nous comparons ces méthodes à un système de référence, entièrement supervisé, sur une tâche de classification audio. Pour nous concentrer que sur ce que peuvent apporter ces algorithmes, nous utilisons une architecture de modèle classique, connue sous le nom de *Wide Resnet* [Zagoruyko and Komodakis, 2016]. Les jeux de données utilisés pour la réalisation des expériences ont été choisis pour leur diversité et popularité : UBS8K, Environmental Sound Classification Dataset (ESC-10) et GSC. Dans ce chapitre, nous étudierons les approches multi-modèle MT et DCT, ainsi que trois autres approches, mono-modèle cette fois-ci, MixMatch (MM), ReMixMatch (RMM), et FixMatch (FM) qui ont été mises en oeuvre par Etienne Labbé, un collègue doctorant travaillant aussi avec des méthodes semi-supervisées.

En gardant seulement 10 % des étiquettes fournies par le jeu de données, nous montrons que toutes ces approches permettent d'obtenir de meilleures performances par rapport à la référence supervisée. Les techniques multi-modèle (MT et DCT) sont néanmoins moins efficaces que les mono-modèle (MM, RMM et FM). Nous montrons également que deux mécanismes de régularisation comme « l'augmentation » et le « Mixup » jouent des rôles importants dans ces approches.

Lorsqu'on regarde de plus près l'évolution des approches pour des tâches de classification semi-supervisée, on peut distinguer une rupture en 2019 avec l'apparition des méthodes dites holistiques. En effet, les approches telles que les *ladder*/Γ/Π modèle, le *temporal ensembling*, MT ou DCT sont des approches basées essentiellement sur le mécanisme de maintien de consistance en utilisant au moins deux modèles. Avec les approches holistiques comme MM, RMM et FM, on tend vers une simplification des architectures, mais une multiplication des mécanismes de SSL. On retrouve ainsi le maintien de la consistance, la minimisation d'entropie et la régularisation, ce dernier mécanisme est d'ailleurs le plus représenté, car les approches holistiques utilisent extensivement l'augmentation de données. En plus d'être plus simples à mettre en place, les approches holistiques sont plus performantes que les anciennes méthodes et ont permis de repousser l'état de l'art, frôlant même les performances jusqu'alors envisageables seulement avec les méthodes entièrement supervisées.

Dans ce chapitre, nous commencerons par détailler toutes les méthodes de SSL utilisée par la suite à l'exception de DCT déjà décrite dans le chapitre précédant. Nous présenterons les augmentations de données utilisées pour l'entraînement de ces méthodes, et plus particulièrement l'augmentation Mixup qui sera le principal sujet d'une expérience supplémentaire. Ensuite, nous parlerons du modèle, des jeux de données et les hyperparamètres utilisés avant de présenter les résultats de nos comparaisons et expériences.

5.1 Algorithmes d'apprentissage profond semi-supervisé

Cette section fournit une description détaillée de chacune des méthodes que nous avons décidé de tester. Nous les avons choisies pour leurs performances et leur nouveauté. Les deux approches MT [Tarvainen and Valpola, 2018] et DCT [Qiao et al., 2018] utilisent deux modèles et le principe de maintien de la consistance décrit dans l'état de l'art (Chapitre 1 Section 1.3.1). Les autres méthodes, MM, RMM, et FM, utilisent un seul modèle et combinent les trois mécanismes de SSL.

Nous fournissons une figure pour illustrer chacune des méthodes. Dans un deuxième temps nous ajouterons Mixup aux méthodes qui ne l'utilisent pas déjà. Nous avons inclus l'opération de mixup dans une boîte de couleur bleu dans les figures.

5.1.1 Mean-Teacher (MT)

Nous pouvons trouver de nombreuses applications de MT [Tarvainen and Valpola, 2018] pour l'audio dans les défis de la tâche 4 du Detection and Classification of Acoustic Scenes and Events (DCASE).

MT utilise deux réseaux neuronaux : un « étudiant » f et un « professeur » g, qui partagent la même architecture. Les poids ω du modèle de l'élève sont mis à jour à l'aide

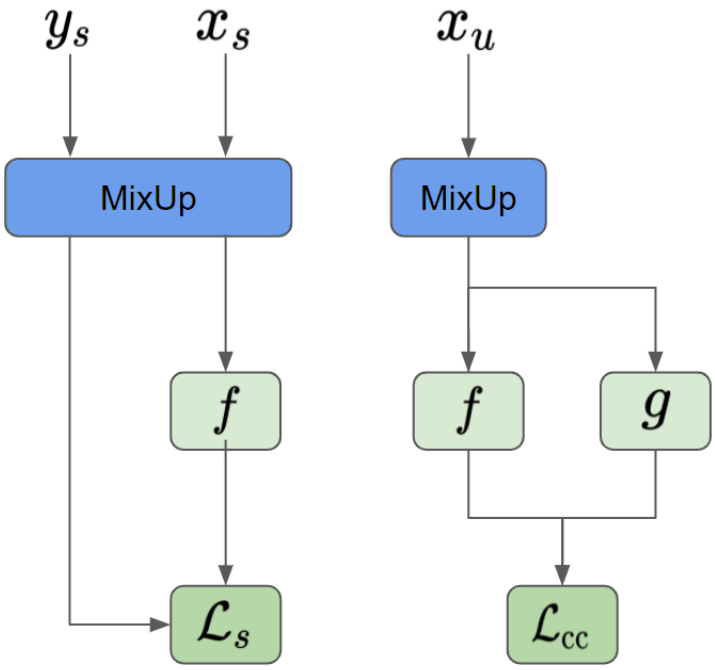

Figure 5.1 – Schéma du Mean Teacher. Les deux modèles reçoivent en entrée des enregistrements sonores étiquetés x_s ou non x_u. Un coût supervisé $\mathcal{L}_s$ est calculé entre la vérité terrain et les prédictions de l'étudiant, tandis qu'un coût de cohérence $\mathcal{L}_{\text{cc}}$ est calculé entre les prédictions de l'étudiant et celles de l'enseignant. MixUp sera ajouté dans un second temps.

de l'algorithme standard de descente de gradient, tandis que les poids W du modèle du professeur sont calculés grâce à la moyenne mobile exponentielle (EMA) des poids de l'élève à chaque itération de mini-batch t, en utilisant la formule ci dessous 5.1. Ils sont calculés comme la combinaison convexe de ses poids à t - 1 et des poids de l'élève, avec une constante de lissage α_{ema}. La répartition des données annotées x_s et non-annotées x_u est décrit par la figure 5.1.

$$W_t = \alpha_{\text{ema}} \cdot W_{t\text{-}1} + (1 - \alpha_{\text{ema}}) \cdot \omega_t \tag{5.1}$$

Deux fonctions de coût sont appliquées, une pour le sous-ensemble de données étiquetées x_s, une autre aux sous-ensembles de données non-étiquetées u_x. Il s'agit pour la première de l'entropie croisée (CE) habituelle qui est utilisée entre les prédictions du modèle étudiant $f(x_s)$ et la vérité terrain y_s.

$$\mathcal{L}_{\text{sup}} = \text{CE}(f(x_s), y_s) \tag{5.2}$$

Pour la seconde, la cohérence est calculée à partir de la prédiction de l'élève $f(x_u)$ et de la prédiction de l'enseignant $g(x'_u)$, où x_u est un échantillon du sous-ensemble non-étiqueté, et x'_u le même échantillon mais légèrement perturbé par un bruit gaussien et un rapport signal/bruit de 15 dB (voir sec 5.2.1). Dans notre cas, nous utilisons l'erreur quadratique

moyenne (Mean Square Error (MSE)).

$$\mathcal{L}_{\text{cc}} = \text{MSE}\Big(f(x_u), g(x'_u)\Big) \tag{5.3}$$

La fonction de coût finale est la somme de la composante supervisée et du coût de cohérence pondéré par un facteur λ_{cc} qui contrôle son influence. λ_{cc} évolue selon un schéma précis qui sera décrit dans la section 5.3.3, et sera utilisé pour la phase d'échauffement.

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{sup}} + \lambda_{\text{cc}} \cdot \mathcal{L}_{\text{cc}} \tag{5.4}$$

5.1.2 Deep Co-Training (DCT)

Tous les détails de cette méthode ont été donnés dans le chapitre 4. On peut la résumer comme suit : DCT est une méthode d'apprentissage basé sur deux modèles complémentaires, utilise trois fonctions de coût différentes qui sont combinées pour donner la fonction d'entraînement finale $\mathcal{L}$. La fonction de coût $\mathcal{L}_{\text{sup}}$ (Eq. 4.2) permet d'utiliser les données supervisées, $\mathcal{L}_{\text{cot}}$ (Eq. 4.4) les données non-supervisées, et finalement $\mathcal{L}_{\text{diff}}$ (Eq. 4.5) permet de forcer les modèles à prendre leurs décisions à partir de caractéristiques différentes.

5.1.3 MixMatch

MixMatch [Berthelot et al., 2019] (MM) utilise la minimisation de l'entropie et la régularisation standard (l'étiquetage artificiel [Lee, 2013], Mixup et l'augmentation des données faibles) pour exploiter les données non-étiquetées et fournir de meilleures capacités de généralisation. Contrairement à MT et DCT, cette approche n'utilise qu'un seul modèle. Les différentes étapes sont illustrées dans la Fig. 5.2 et détaillées dans les paragraphes suivants.

Pendant la phase d'apprentissage, chaque mini-batch est composé d'échantillons étiquetés x_s et non-étiquetés x_u dans des proportions équivalentes. La première étape consiste à appliquer une augmentation à la partie étiquetée du mini-batch et k augmentations à la partie non-étiquetée. Dans la deuxième étape, des pseudo-étiquettes $\hat{y}_u$ sont générées pour les échantillons non-étiquetés en utilisant la prédiction du modèle moyennée sur ces k variantes, comme indiqué dans l'équation 5.5, où $x'_{u,i}$ désigne la i-ième variante d'un exemple augmenté non-étiqueté, et $f(x)$ la prédiction du modèle.

$$\hat{y}_u = \frac{1}{k}\sum_{i=1}^{k} f(x'_{u,i}) \tag{5.5}$$

Pour encourager le modèle à produire des prédictions fiables, une étape de post-traitement est nécessaire pour diminuer l'entropie de la sortie. Pour ce faire, la probabilité la plus élevée est augmentée et les autres diminuées. Ce processus est appelé « aiguisage » (ou *sharpening*

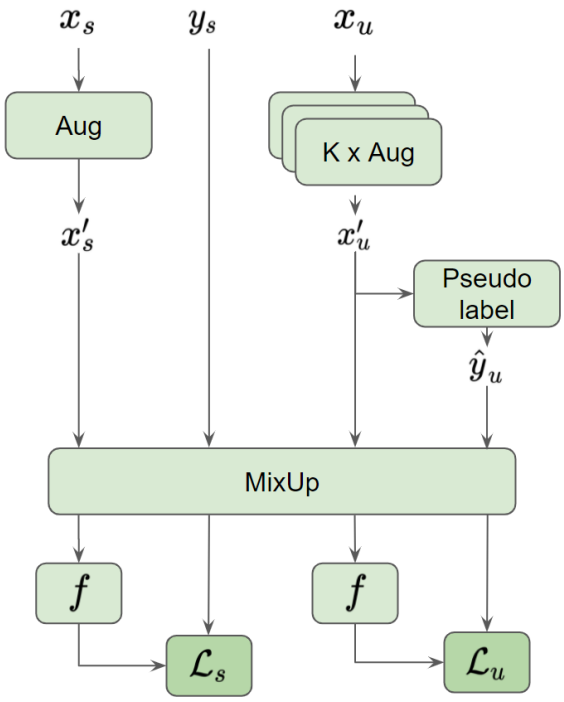

Figure 5.2 – Schéma de MM. K augmentations sont appliquées aux données non-étiquetées x_u, et la moyenne des prédictions du modèle est utilisée comme pseudo-étiquettes $\hat{y}_u$. Les données non-étiquetées, étiquetées et augmentées sont mélangées et utilisées pour calculer les fonctions de coût supervisées et non-supervisées.

en anglais) par les auteurs de la méthode, et il est défini par la fonction $\mathrm{Sharpen}$ comme suit :

$$\mathrm{Sharpen}(p, \mathcal{T})_i := p_i^{1/T} \Big/ \sum_{j=1}^{|p|} p_j^{1/\mathcal{T}} \tag{5.6}$$

La fonction $\mathrm{Sharpen}$ est appliquée aux pseudo-étiquettes $p = \hat{y}_u$. Le paramètre $\mathcal{T}$, appelé Température, contrôle la force de cette fonction. Plus $\mathcal{T}$ s'approche de zéro, plus l'entropie des prédictions baisse.

Enfin, tous les échantillons augmentés, étiquetés ou non, sont concaténés et mélangés dans un ensemble W puis utilisés comme pool d'échantillons d'entraînement par la fonction Mixup. Le mélange asymétrique est appliqué séparément sur les parties étiquetées et non-étiquetées du mini-lot, comme formulé ici :

$$x'^{\mathrm{mix}}_s = \mathrm{Mixup}(x_s, W_{1..|x_s|}) \tag{5.7}$$
$$x'^{\mathrm{mix}}_u = \mathrm{Mixup}(x_u, W_{|x_s|..|W|}) \tag{5.8}$$

L'ensemble W et les étiquettes correspondantes sont mélangés dans le même ordre. Chaque échantillon étiqueté est ensuite perturbé par un deuxième échantillon étiqueté ou non-étiqueté. Le mélange des deux est effectué de manière à ce que l'échantillon étiqueté

original reste le composant principal de l'échantillon résultant. L'opération a été détaillée dans la section 5.2.2. La même procédure est appliquée aux échantillons non-étiquetés en utilisant les échantillons restants de W.

La fonction de coût originale de MixMatch est composée de l'entropie croisée (CE) standard pour le coût supervisé $\mathcal{L}_s$, et d'une norme $L2$ pour le coût non-supervisé $\mathcal{L}_u$. Nous avons remplacé la norme $L2$ par une entropie croisée dans toutes nos expériences, comme proposé dans l'article sur le ReMixMatch (RMM) Berthelot et al. [2020]. En effet, il semble que la CE soit plus performante que la norme $L2$ dans nos expériences.

$$\begin{aligned} \mathcal{L}_s &= \frac{1}{B_s} \sum_{(x'^{\mathsf{mix}}_s, y^{\mathsf{mix}}_s)} \mathsf{CE}(f(x'^{\mathsf{mix}}_s), y^{\mathsf{mix}}_s) \\ \mathcal{L}_u &= \frac{1}{k \cdot B_u} \sum_{(x'^{\mathsf{mix}}_u, \hat{y}^{\mathsf{mix}}_u)} \mathsf{CE}(f(x'^{\mathsf{mix}}_u), \hat{y}^{\mathsf{mix}}_u) \end{aligned} \tag{5.9}$$

Le coût total est la somme des deux composantes, avec un hyperparamètre λ_u :

$$\mathcal{L} = \mathcal{L}_s + \lambda_u \cdot \mathcal{L}_u \tag{5.10}$$

5.1.4 ReMixMatch (RMM)

ReMixMatch [Berthelot et al., 2020] (RMM) est une extension de MixMatch qui introduit la notion d'augmentations faibles et fortes, un mécanisme d'alignement de la distribution ainsi que deux fonctions de coût supplémentaires : $\mathcal{L}_{u_1}$ qui est calculé à partir des pseudo-labels $\hat{y}_u$, et $\mathcal{L}_r$ qui est un coût auto-supervisé.

À chaque étape, nous appliquons une augmentation faible et k augmentations fortes sur le lot non-étiqueté x_u. Le lot faiblement augmenté y_u sera utilisé pour créer les pseudo-labels $\hat{y}_u$ à partir des prédictions du modèle $f(x'_u) = P\Big(y|x'_u;\theta\Big)$:

$$\hat{y}_u = f(x'_u)$$

La fonction d'alignement de la distribution modifie les pseudo-étiquettes pour obtenir une distribution proche de celle de la partie étiquetée. Pour appliquer ce processus, nous calculons la distribution étiquetée P_s avec les vrais labels et la distribution non-étiquetée P_u est calculée comme étant la moyenne des prédictions des 128 derniers lots prédits, tandis que pour P_s c'est directement la vérité terrain qui est utilisée. Finalement, nous pouvons normaliser nos labels artificiels $\hat{y}_u$ de la façon suivante :

$$\hat{y}_u = \hat{y}_u \cdot \frac{P_s}{P_u} \tag{5.11}$$

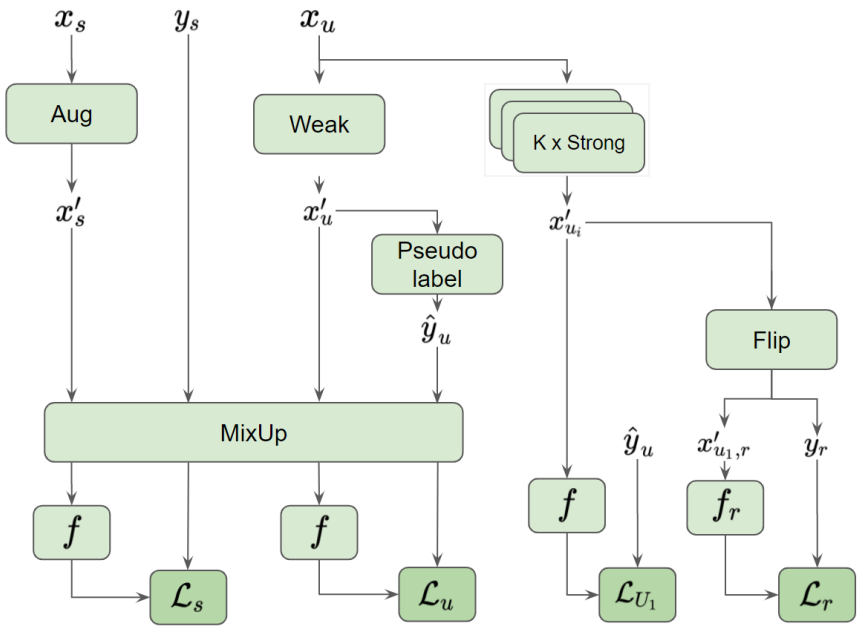

Figure 5.3 – Schéma RMM. Une augmentation faible et K augmentations fortes sont appliquées aux données non-étiquetées x_u. Les données non-étiquetées faiblement augmentées x'_u sont utilisées pour créer des pseudo-étiquettes $\hat{y}_u$. Le premier lot de données non-étiquetées fortement augmentées $x'_{u_i}, i = 1$ est utilisé pour la fonction de coût non-supervisée $\mathcal{L}_{u_1}$ (en utilisant les pseudo-étiquettes $\hat{y}_u$) et dans la fonction de coût auto-supervisée $\mathcal{L}_r$. Les étiquettes créées par l'algorithme d'auto-apprentissage sont nommées y_s. Dans notre cas, il s'agit de prédire une symétrie horizontale et / ou verticale.

De plus, nous appliquons l'équation $\mathrm{Sharpen}$ 5.6 sur les pseudo-labels $\hat{y}_u$ comme dans MixMatch. L'étiquette $\hat{y}_u$ sera utilisée comme cible pour les lots faibles et forts. Comme dans MixMatch, nous concaténons les lots étiquetés et non-étiquetés en un ensemble W pour le Mixup, et les fonctions de coût supervisées et non-supervisées $\mathcal{L}_s$ et $\mathcal{L}_u$ restent les mêmes.

ReMixMatch introduit également une fonction de coût forte pour augmenter la stabilité et la précision du système. Cette composante, calculée à partir de la première version fortement augmentée de x_u, est nommée $\mathcal{L}_{u_1}$.

$$\mathcal{L}_{u_1} = \frac{1}{B_u} \sum_{\left(x'_{u_1}, \hat{y}_u\right)} \mathrm{CE}\left(P\left(y|x'_u;\theta\right), \hat{y}_u\right) \tag{5.12}$$

La dernière fonction de coût est une composante auto-supervisée qui doit prédire quelle transformation est appliquée au lot x'_{u_1}. Elle aide le modèle à éviter de s'effondrer pendant l'apprentissage. Dans le ReMixMatch original, la transformation utilisée était une rotation de 0, 90, 180 ou 270 degrés. Ici, nous avons choisi de remplacer cette composante par une stratégie de symétrie aléatoire appliquée sur le log-mel spectrogramme de l'enregistrement audio. Les quatre transformations que nous avons utilisées sont : l'identité (i), la symétrie verticale (v), la symétrie horizontale (h) et la combinaison des deux (hv). Le modèle essaie de prédire quelle symétrie est appliquée grâce une sortie dédiée dans le modèle. Comme les autres composants, le composant auto-supervisé utilise l'entropie croisée standard, et y_r est

échantillonné à partir de {i, v, h, hv}. :

$$\mathcal{L}_r = \frac{1}{B_u} \sum_{(x'_{u_1}, \hat{y}_u)} \mathrm{CE}\left(P(y|x'_{u_1,r};\theta), y_r\right) \tag{5.13}$$

Enfin, nous calculons le coût final comme étant la somme des différentes composantes pondérées par leur coefficient λ respectif.

$$\mathcal{L} = \mathcal{L}_s + \lambda_u \cdot \mathcal{L}_u + \lambda_{u_1} \cdot \mathcal{L}_{u_1} + \lambda_r \cdot \mathcal{L}_r \tag{5.14}$$

5.1.5 FixMatch

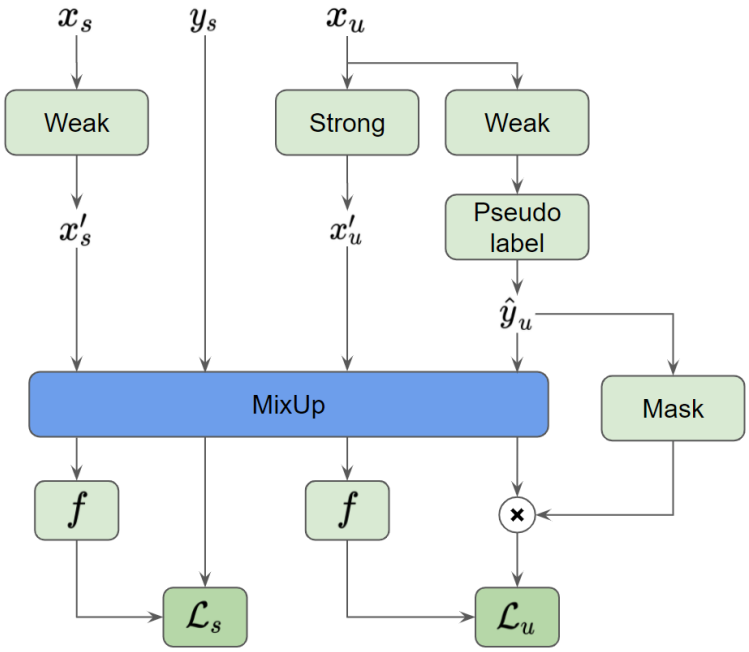

Figure 5.4 – Schéma de FM. Les données étiquetées sont faiblement augmentées pour obtenir x'_s. Celles qui ne sont pas étiquetées sont faiblement et fortement augmentées pour obtenir x'_u. La variante faiblement augmentée est utilisée pour créer des pseudo-étiquettes $\hat{y}_u$. Ces dernières seront filtrées pour garder seulement celles qui dépassent un certain seuil avant de les utiliser dans la fonction de coût non-supervisée $\mathcal{L}_u$.

FixMatch [Sohn et al., 2020] (FM) propose une simplification de MM et RMM. Elle utilise également un seul modèle, supprime le Mixup, et remplace la fonction Sharpen par des pseudo-étiquettes binaires. FM utilise à la fois des augmentations faibles (Weak) et fortes (Strong). Ces dernières peuvent induire en erreur les prédictions du modèle en perturbant trop les données d'apprentissages. La figure 5.4 montre le schéma de fonctionnement de FM.

La fonction de coût supervisée utilise l'entropie croisée standard appliquée aux données faiblement augmentées :

$$\mathcal{L}_s = \mathsf{CE}\Big(f\Big(P(y|x'_s;\theta)\Big), y_s\Big) \tag{5.15}$$

Pour la fonction de coût non-supervisée $\mathcal{L}_u$, les prédictions du système $\hat{y}_u$ - obtenus à partir des données non-étiquetées faiblement augmentées - sont filtrées par un seuil τ pour ne garder que celles prédites avec une confiance élevée. Cela permet au modèle de mieux généraliser et d'utiliser les données non-étiquetées et ainsi améliorer la performance du système. Le seuillage permet d'éviter les erreurs d'apprentissages qui peuvent émerger surtout au début de l'entraînement lorsque le modèle n'est pas encore capable de différencier les différentes classes. Les formules ci-dessous formalisent la création des pseudo-labels, le masque de seuillage ainsi que la fonction de coût non-supervisée.

$$\mathcal{L}_u = \mathrm{Mask} \cdot \mathrm{CE}\left(P\Big(y|\,\mathrm{Strong}(x_u);\theta\Big), \hat{y}_u\right) \tag{5.16}$$

$$\hat{y}_u = f\left(P\Big(y|\,\mathrm{Weak}(x_u);\theta\Big)\right) \tag{5.17}$$

$$\mathrm{Mask} = \mathbb{1}\Big(\mathsf{max}(\hat{y}_u) > \tau\Big) \tag{5.18}$$

Comme pour MM et RMM la fonction de coût finale est calculée comme étant la somme des différentes composantes, pondérées par leur coefficient λ respectif.

$$\mathcal{L} = \mathcal{L}_s + \lambda_u \cdot \mathcal{L}_u \tag{5.19}$$

5.2 L'augmentation de données au coeur des méthodes holistiques

L'une des techniques de régularisation la plus utilisée en apprentissage profond, et qui a fait ses preuves en apprentissage supervisé, est l'augmentation de données. On applique une, ou plusieurs transformations sur l'enregistrement audio avant de l'envoyer au modèle. Cette modification n'est pas censée affecter la sémantique de la classe, c'est-à-dire que pour un exemple x ou sa version modifiée $\mathrm{Augment}(x)$, le système doit prédire la même classe. Par exemple, dans le cas d'une tâche de classification d'images, il est courant de déformer / étirer / déplacer une image sans altérer son étiquette [Cireșan et al., 2010; Simard et al., 2003; Cubuk et al., 2019]. Elle permet de tirer parti des exemples non-étiquetés qui sont fournis, lors de l'apprentissage, au système de classification. Dans cette section, nous décrivons les différentes augmentations que nous avons utilisées pour l'audio, et nous regardons plus en détail le mécanisme de régularisation (qui se comporte comme une augmentation) qu'est Mixup.

5.2.1 Augmentations

Pour l'apprentissage supervisé, MM, RMM et FM, nous avons sélectionné trois augmentations différentes : *Occlusion*, *StretchPadCrop*, et *CutOutSpec*. Pendant l'entraînement, seulement une de ces augmentations est appliquée de manière aléatoire à chaque échantillon.

1. **Occlusion** : Appliquée au signal audio brut, l'occlusion consiste à mettre à zéro une section du signal. La taille du segment est choisie de manière aléatoire et peut aller de quelques millisecondes à plusieurs secondes. Cet intervalle est défini par l'utilisateur et dépend de la longueur des exemples du jeu de données. La position du segment est également choisie de manière aléatoire.

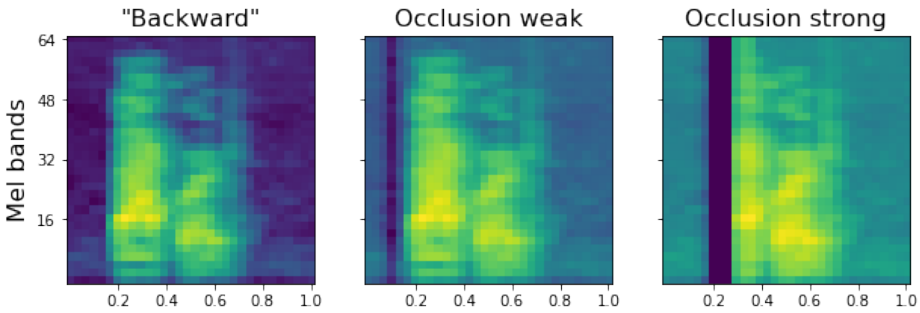

Figure 5.5 – Impact de l'augmentation « Occlusion » sur un enregistrement audio issu de GSC. De gauche à droite : original (le mot *Backward*), la variante faible de l'augmentation puis l'augmentation forte.

2. **Time Stretch** : également appliquée au signal audio brut, le *Time Strech* consiste à accélérer ou ralentir le signal. La façon dont le signal est modifié est choisie de manière aléatoire dans un intervalle prédéfini par l'utilisateur. L'exemple augmenté qui en résulte est soit plus court, soit plus long. Il est alors nécessaire de le compléter ou de le couper pour maintenir constante la durée des échantillons.

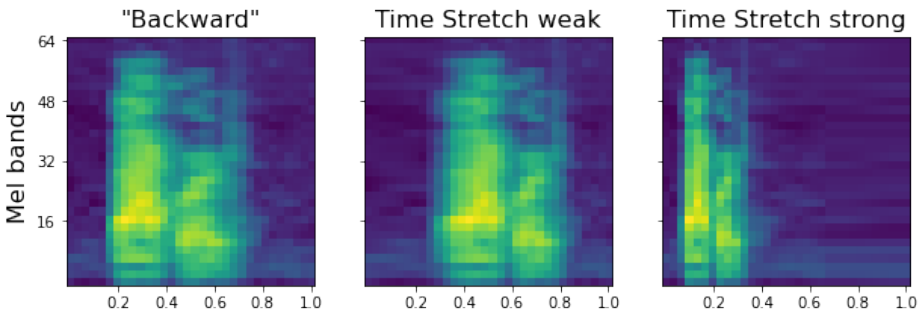

Figure 5.6 – Exemple de l'augmentation « Time Stretch » sur un enregistrement audio issu de Google Speech Command, de gauche à droite : L'enregistrement original (le mot **Backward**), faible augmentation (un ralentissement faible), forte augmentation (une accélération forte).

3. **CutOutSpec** : appliqué aux spectrogrammes log-Mel, CutOutSpec fixe les valeurs dans une zone rectangulaire aléatoire avec la valeur -80 dB, qui correspond au niveau d'énergie du silence dans nos spectrogrammes. La longueur et la largeur des sections

supprimées sont choisies aléatoirement dans un intervalle prédéfini et dépendent de la taille du spectrogramme.

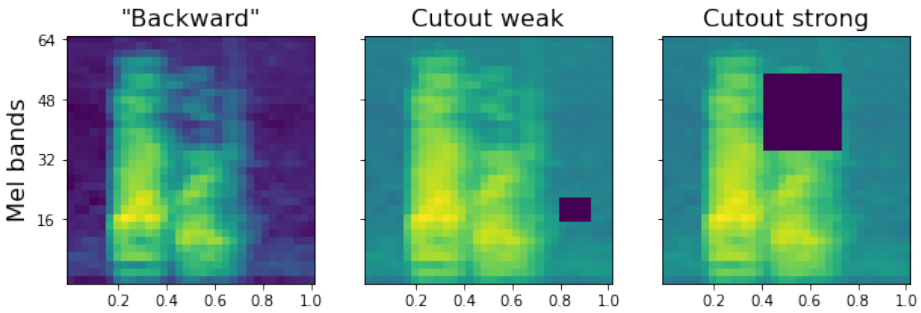

Figure 5.7 – Impact de l'augmentation « Cutout » sur un enregistrement audio issu de GSC. De gauche à droite : original (le mot « Backward »), la variante faible de l'augmentation puis l'augmentation forte.

Les augmentations dites « Faibles » et « Fortes »

Nous avons systématiquement deux versions de chaque augmentation que nous nommerons « Faible » et « Forte » . La différence principale entre ces deux variantes est la force de l'augmentation, ainsi que l'importance du changement apporté. Cela est particulièrement bien illustré avec l'Occlusion (Figure 5.5) ou encore avec SpecAugment(Figure 4.8). Lors de l'entraînement, les augmentations fortes seront systématiquement appliquées tandis que les augmentations faibles auront 50 % de chance d'être appliquées.

Pour les augmentations faibles, il s'agit d'ajouter virtuellement des fichiers annotés au jeu de données. Pour cela, elles ne doivent pas affecter la sémantique des classes. Cela signifie que la prédiction du système ne devrait pas changer, que ce soit pour l'échantillon original ou sa variante augmentée. Pour les augmentations fortes, elles sont utilisées lorsqu'on cherche à se rapprocher des limites du modèle pour lui présenter des versions extrêmes de certains enregistrements. Ces augmentations auront plus tendance à tromper le système et seront principalement utilisées de manière conjointe avec le mécanisme de minimisation d'entropie.

Il existe des altérations qui sont naturellement incompatibles dans un cadre semi-supervisé, car la probabilité qu'elles puissent tromper le système est trop élevée, même pour leur variante faible. Par exemple, dans le cas d'un enregistrement audio, les augmentations modifiant les fréquences, telles que *Pitch Shift*, sont à éviter, car elles peuvent modifier l'étiquette. C'est le cas pour les enregistrements contenant des évènements longs tels qu'un aspirateur ou un ventilateur qui sont facilement confondus si on change leurs fréquences.

La sélection des augmentations

Afin de sélectionner les augmentations les plus efficaces, nous avons procédé comme suit : Pour chaque augmentation, nous avons entraîné plusieurs modèles, en modifiant légèrement les paramètres d'augmentation jusqu'à l'obtention d'un résultat satisfaisant. Pour limiter le

temps de calcul, seul un petit ensemble de valeurs a été testé. Une augmentation et ses paramètres étaient alors sélectionnés si elle permettait d'atteindre un meilleur score que notre modèle de référence, non augmenté.

Parmi les augmentations décrites ci-dessus, nous avons sélectionné trois augmentations dont les détails sont fournis dans le tableau ci-dessous **??**

Nom	Paramètres	Faible	Forte
Occlusion	max_size	[0,25, 0,25]	[0,75, 0,75]
Time Stretch	rate	[0,50, 1,50]	[0,25, 1,75]
Cutout	scale	[0,10, 0,50]	[0,50, 1,00]

Table 5.1 – Liste des paramètres d'augmentation utilisés dans la formation supervisée augmentée, MixMatch et FixMatch.

5.2.2 Mixup

Mixup [Carratino et al., 2020] est une technique d'augmentation de données qui crée de nouveaux exemples sous forme de combinaisons convexes d'échantillons d'entraînement (et des étiquettes correspondantes). Cette technique simple a empiriquement montré qu'elle améliore la précision de nombreux modèles dans différents contextes et applications. La figure 5.8 donne un exemple du mélange de deux enregistrements audios. L'avantage principal de cette augmentation, contrairement à celles que j'ai présentées plus haut, c'est qu'elle est agnostique aux données et donc beaucoup plus facile à utiliser. Cependant, il faut faire attention lorsqu'on l'utilise lors d'un apprentissage semi-supervisé, car on veut éviter que l'exemple non-annoté prenne le pas sur l'exemple annoté, il faudra alors utiliser la variante « non-symétrique » de la formule (5.2.2).

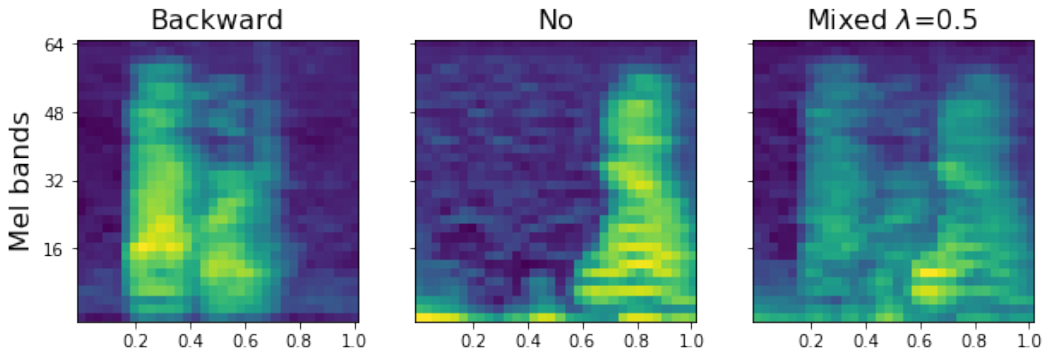

Figure 5.8 – Exemple d'un mélange de deux enregistrements audios provenant du jeu de données GSC grâce à l'algorithme de MixUp. La valeur de lambda est $\lambda = 0,50$

Formule symétrique

Si x_1 et x_2 sont deux échantillons d'entrée différents (des spectrogrammes dans notre cas) et y_1, y_2 leurs étiquettes respectives codées sous la forme d'un vecteur *one hot*, alors, Mixup mélange x_1 avec x_2 et fait de même avec leur étiquette grâce à une simple combinaison convexe :

$$\mathrm{Mixup}(x_1, x_2) \quad = \quad \lambda x_1 + (1 - \lambda) x_2 \tag{5.20}$$

- Où λ est un scalaire échantillonné à partir d'une distribution Bêta symétrique à chaque génération de mini-lots :
$$\lambda \sim \mathrm{Beta}(\alpha, \alpha)$$
- Où α est un hyperparamètre à valeur réelle à ajuster (toujours inférieur à 1,0 dans notre cas).

Formule asymétrique

Dans l'algorithme MM original, une version « asymétrique » du mélange est utilisée, dans laquelle la valeur maximale entre λ et $1 - \lambda$ est récupérée :

$$\lambda = \max(\lambda, 1 - \lambda) \tag{5.21}$$

Cela permet à l'exemple mélangé résultant d'être plus proche de l'un des deux échantillons d'origine (celui qui a le coefficient λ). Ceci est utile lorsque la méthode mélange des échantillons avec, et sans annotation.

5.3 Détails sur les expériences

Dans la section 5.2, nous avons décrit les différents algorithmes SSL que nous avons testés, ainsi que les méthodes d'augmentation de données choisies pour la phase d'entraînement. Cette section fournit les détails concernant l'architecture du modèle utilisé, les jeux de données utilisés et les paramètres d'entraînement.

5.3.1 Modèle utilisé pour les expérimentations

Nous avons choisi d'utiliser le modèle *Wide Resnet* 28-2 pour réaliser l'ensemble de nos entraînements et cela, quel que soit la méthode d'apprentissage ou le jeu de données. Ce

modèle est très efficace, a démontré qu'il pouvait atteindre l'état de l'art supervisé pour la classification d'évènements sonores sur les trois jeux de données choisis.

L'avantage de ce modèle est que, grâce à sa taille relativement faible de 1,4 million de paramètres entraînables, il est possible d'itérer et d'expérimenter rapidement car le temps total d'entraînement est faible. C'est un avantage certain, puisque certaines méthodes d'apprentissages semi-supervisées peuvent être beaucoup plus longues comparées à un entraînement classique (voir Section 5.4.4).

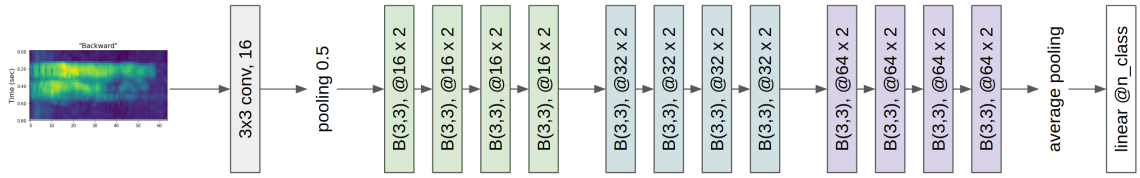

Figure 5.9 – L'Architecture détaillée du modèle *Wide Resnet* 28-2. On peut discerner trois ensembles de quatre BasicBlock$(3, 3)$ chacun.

Sa structure, décrite dans la figure 5.9, consiste en une couche convolutive initiale, suivie de trois groupes de blocs résiduels et finalement, une moyenne globale suivie d'une couche linéaire agissant comme un classificateur. La première convolution contient 16 filtres, et les trois BasicBlock$(3, 3)$ (Voir Chapitre 1 Section 1.1.2) ont un nombre de filtres de base successivement égal à 16, 32, et 64, doublé par le coefficient multiplicateur $k = 2$.

L'architecture du *Wide Resnet* est déjà implémentée dans la bibliothèque d'apprentissage profond PyTorch [Paszke et al., 2019] et semble être très appropriée comme modèle de référence pour réaliser les expériences.

5.3.2 Jeux de données et pré-traitement

Les trois jeux de données utilisés sont très courants pour des tâches d'audio tagging mono-label. Nous les avons choisi, car ils nous permettent d'avoir plus de variété et de configuration possibles pour la réalisation des expériences. Nous avons utilisé en entrée de nos systèmes, des log-mel spectrogrammes calculés avec une fenêtre de 2048 échantillons et un pas de 512. Les spécificités telles que la fréquence d'échantillonnage, la taille du spectrogramme obtenue est différente pour chaque jeu de données, et sera précisée ci-dessous.

ESC-10

ESC-10 [Piczak, 2015] est une sélection de 400 enregistrements de cinq secondes d'évènements audio séparés en dix catégories. Le nombre d'exemples présents pour chaque classe est équilibré avec 40 représentants par classe. Il est séparé en 5 dossiers qui doivent être utilisés pour faire une validation croisée et ainsi correctement évaluer les performances de nos systèmes. Les enregistrements audio fournis sont mono-label (c'est-à-dire que seulement

un évènement est présent), échantillonnés en 44,1 kHz et encodés en 16 bits avant de les convertir en 64×431 log-mel spectrogrammes.

UBS8K

UBS8K [Salamon et al., 2014] est composé de 8742 enregistrements sonores d'une durée comprise entre 1 et 4 secondes, séparés en dix catégories plus ou moins équilibrées. La catégorie *car horn* et *gun shot* ont seulement 429 et 374 exemples respectivement tandis que toutes les autres en ont 1000. UBS8K est découpé en 10 dossiers qui doivent être utilisés pour faire une validation croisée. Cette procédure permet une meilleure évaluation des systèmes et aussi la comparaison avec d'autres recherches. Les enregistrements qui ne durent pas quatre secondes sont complétés avec du silence pour homogénéiser les durées. La fréquence d'échantillonnage étant très variable, nous les avons tous reéchantillonnés à 22 kHz et encodés en 16 bits avant de les convertir en 64×173 log-mel spectrogramme.

Google Speech Command GSC

GSC [Warden, 3209] est composé exclusivement de 35 mots parlés et est conçu pour évaluer les systèmes de détection de commandes (ou mots-clés). Il est divisé en trois sous-ensembles contenant : 85 511 exemples pour l'entraînement, 10 102 exemples pour la validation et 4890 exemples pour l'évaluation. Les enregistrements qui ne durent pas une seconde sont complétés avec du silence et ils sont tous échantillonnés à 16 kHz avant de les convertir en 64×32 log-mel spectrogramme.

5.3.3 Entraînement

Les hyperparamètres de l'apprentissage ont été déterminés à l'aide d'une recherche par grille. Pour chaque paramètre, étant donné un minimum, un maximum et une taille de pas, la performance est évaluée sur un ensemble de validation. Pour un apprentissage supervisé, le nombre de paramètres à optimiser est déjà important, les méthodes semi-supervisées rajoutent encore plus de paramètres (voir section 5.1) et donc, la recherche prend encore plus de temps.

Nous avons appliqué cette méthode pour sélectionner nos hyperparamètres, mais comme nous avons sept algorithmes différents et trois ensembles de données, nous n'avons pas pu réaliser les 21 recherches par grille nécessaires. Au lieu de cela, nous l'avons fait une fois pour chaque algorithme sur le jeu de données GSC, et nous avons conservé les paramètres sélectionnés pour les deux autres. Ces paramètres sont détaillés dans la table 5.2.

Méthodes	bs	lr	wl	e	α
Supervised	256	0,001	-	300	-
mixup	256	0,001	-	300	0,4
MT	64	0,001	50	200	-
DCT	64	0,0005	160	300	-
MM	256	0,001	-	300	0,75
RMM	256	0,001	-	300	-
FM	256	0,001	-	300	-
MT+mixup	64	0,001	50	200	0,40
DCT+mixup	64	0,0005	160	300	0,40
MM-mixup	256	0,001	-	300	-
RMM-mixup	-	-	-	-	-
FM+mixup	256	0,001	-	300	0,75

Table 5.2 – Paramètres d'apprentissage utilisés sur les jeux de données. bs : taille du lot, lr : taux d'apprentissage, wl : durée de l'échauffement en nombre d'époques, e : nombre d'époques, α : paramètre de mélange Beta.

Optimiseur et taux d'apprentisssage

Nous avons utilisé l'optimiseur ADAM [Kingma and Ba, 2017] pour l'entraînement de nos modèles. Le taux d'apprentissage lr est maintenu constant pendant la phase d'entraînement pour les méthodes supervisées, MM, RMM et FM alors que pour MT et DCT, il est pondéré par une règle de cosinus décroissant définie comme suit :

$$\mathrm{lr} = \frac{1}{2}\Big(1,0 + \cos\Big((t\text{ - }1)\frac{\pi}{e}\Big)\Big) \tag{5.22}$$

Échauffement (*Warmup*)

Toutes les approches SSL, introduisent un ou plusieurs termes subsidiaires dans la fonction de coût. Pour atténuer leur impact au début de l'apprentissage, ces termes sont pondérés par un coefficient lambda λ, qui augmente progressivement jusqu'à sa valeur maximale λ_{max} pendant une durée dite d'échauffement wl (le *warmup* en anglais). Cette stratégie est linéaire pour MM, RMM et FM, sinon, elle est définie par l'équation 5.23 pour MT et DCT.

$$\lambda = \lambda_{\mathsf{max}} \times \left(1 - e^{-5\times(1-(t/\mathrm{wl}))^2}\right) \tag{5.23}$$

- Dans MT, la valeur maximale de λ_{cc} vaut 1 et α_{ema} est fixé à 0,999.
- Pour DCT, les valeurs maximales de λ_{cot} et λ_{diff} sont respectivement de 1 et 0,5.
- Pour MM, la valeur maximale de λ_u est de 1.
- Pour RMM, la valeur maximale de λ_u vaut 1,5, λ_{u_1} et λ_r 0,5.

- Pour FM, λ_u vaut 1.

Autres paramètres

Les méthodes MM, RMM et FM utilisent le mécanisme du maintien de la consistance et de la minimisation de l'entropie grâce aux augmentations appliquées sur les données d'entrée du système. Pour MM, deux augmentations ($k = 2$) sont utilisées, le paramètre de température $\mathcal{T}$, utilisé dans la fonction $\mathrm{Sharpen}$ (Eq. 5.6) est fixé à 0,5. Pour FM, le seuil τ est fixé à 0,8 sur les jeux de données ESC-10 et GSC, et à 0,95 pour UBS8K.

Lors de l'entraînement, les mel-spectrogrammes sont envoyés au modèle par paquets dont la taille est définie par le paramètre bs. Ces lots contiennent à la fois des exemples annotés et non annotés dont la proportion change en fonction de la méthode. Pour MM, RMM et FM le nombre d'exemples annotés est identique au nombre d'exemples non annotés. Pour MT et DCT la répartition suit celle du jeu de données. C'est-à-dire que pour une expérience à 10 % / 90 %, le lot contiendra 10 % d'échantillon annoté et 90 % de non annotés.

5.4 Résultats

Tous les résultats de cette section sont basés sur une validation croisée pour Environmental Sound Classification Dataset (ESC) (10 *folds*) et UBS8K (5 *folds*). Nous calculons à la fois le taux d'erreur moyen et l'écart-type pour l'évaluation des différentes méthodes SSL et les systèmes de référence supervisées classiques et augmentées. Nous aborderons dans un premier temps, la performance de l'apprentissage supervisé classique avec et sans augmentation (incluant Mixup), puis la performance des différentes approches SSL, pour finalement discuter de l'impact de Mixup sur la performance globale.

5.4.1 Référence supervisée

Les méthodes SSL ne sont intéressantes que si les performances que nous pouvons obtenir sont au moins assez proches de celles de l'apprentissage supervisé. Il est également important de considérer que certains des mécanismes utilisés dans SSL peuvent également être appliqués à un apprentissage supervisé. Par conséquent, nous présentons le résultat de l'apprentissage supervisé avec et sans augmentations, en utilisant les mêmes augmentations que celles employées dans les méthodes SSL. Le tableau 5.3 montre les performances de l'apprentissage supervisé pour les trois ensembles de données en utilisant 10 % et 100 % des étiquettes.

Nous avons entraîné des modèles sans aucune augmentation (supervisé), en utilisant le Mixup seul (Mixup), les augmentations faibles seules (Faible), une combinaison d'aug-

mentations faibles et de Mixup (Faible+Mixup), les augmentations fortes seules (Forte), et pour finir, une combinaison d'augmentations fortes et de Mixup (Forte+Mixup).

Dataset	ESC-10		UBS8K		GSC	
Pourcentage annoté	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
+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**

Table 5.3 – Taux d'erreur et écart type obtenus lors d'un entraînement supervisé avec différentes augmentations pour ESC-10, UBS8K and GSC.

ESC-10

— En utilisant 10 % des étiquettes, l'entraînement supervisé sans augmentation donne un taux d'erreur de 32 %. En utilisant les augmentations faibles, nous obtenons le taux d'erreur le plus bas, soit 22,67 %, ce qui représente une amélioration de 9,3 points (amélioration de 29.16 %).

— En utilisant les 100 % des étiquettes, un entraînement supervisé sans augmentation produit un taux d'erreur de 8,00 %, tandis que l'utilisation des augmentations faibles et Mixup fournit une amélioration de 3,33 points, portant le taux d'erreur à 4,67 %(amélioration de 41.62 %).

UBS8K

— En utilisant 10 % des labels, l'entraînement supervisé sans augmentation donne un taux d'erreur de 33,80 %. En utilisant les augmentations faibles combinées Mixup, nous pouvons obtenir le taux d'erreur le plus bas de 23,75 %, ce qui représente une amélioration de 10,05 points (ou 29,73 % relatifs).

— En utilisant les 100 % des étiquettes, la même combinaison d'augmentation a donné les meilleurs résultats avec un taux d'erreur de 17,96 %, soit 5,33 points de moins. Cela représente une amélioration relative de 22,88 %.

GSC

— Une fois de plus, c'est la combinaison d'augmentation faible + Mixup qui permet d'atteindre le meilleur taux d'erreur de 6,58 lors d'un entraînement utilisant 10 % des annotations. Si l'on compare le score sans augmentation de 10,01 %, cela représente une amélioration de 3,43 points ou 34,26 %.

— Cette fois, en utilisant 100 % des étiquettes, la performance de l'entraînement supervisé sans augmentation est déjà très bonne, avec un taux d'erreur de 4,94 %. L'association de l'augmentation forte et Mixup a toutefois permis une amélioration substantielle, avec une diminution de 1,96 point, pour un taux d'erreur de 2,98 % ou 39.68 %.

Dans l'ensemble, nous observons que dans un cadre supervisé, si l'utilisation d'une seule augmentation a amélioré la précision de nos systèmes, la combinaison de Mixup avec une augmentation faible ou forte est systématiquement meilleure que l'utilisation d'une de ces augmentations individuellement, sauf dans le jeu de données ESC-10. Cela peut s'expliquer en partie par le fait que les augmentations audios sont souvent difficiles à choisir et que leur impact dépend souvent du jeu de données et de la tâche à accomplir [Lu et al., 2019]. Dans cette optique, Mixup semble être bénéfique quel que soit le jeu de données utilisé.

5.4.2 Résultats SSL

Dataset	ESC-10		UBS8K		GSC	
Fraction 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
MT	28,28 ± 5,28	-	32,80 ± 4,21	-	8,51	-
DCT	25,16 ± 4,42	-	27,85 ± 4,29	-	6,22	-
MM	15,33 ± 5,58	-	**18,02 ± 4,00**	-	**3,25**	-
RMM	19,17 ± 6,52	-	26,02 ± 6,90	-	3,78	-
FM	**13,33 ± 2,89**	-	21,44 ± 4,16	-	4,44	-

Table 5.4 – Taux d'erreur et écart type obtenus pour chaque méthode semi-supervisé pour ESC-10, UBS8K and GSC.

Maintenant que nous avons établi nos systèmes de référence avec et sans augmentation, nous pouvons correctement comparer les résultats des différentes méthodes SSL testées. Le tableau 5.4 résume les performances obtenues dans un cadre entièrement supervisé sans augmentation ainsi que le système le plus performant pour les trois ensembles de données. Il fournit également le taux d'erreur moyen et l'écart-type obtenus pour les trois ensembles de données et chaque méthode SSL. Pour le GSC, seulement la moyenne est donnée car les écart-type sont très proches.

La première observation que nous pouvons faire est que, par rapport à un entraînement supervisé sans augmentation, toutes les méthodes SSL ont un meilleur taux d'erreur et quelque soit le jeu de données. Cependant, face aux entraînements supervisés augmentés, MT et DCT sont moins performants.

La deuxième observation que nous pouvons faire est que MM, RMM et FM surpassent largement MT et DCT sur les trois ensembles de données, à l'exception de la performance de RMM sur UBS8K.

ESC-10

— Le meilleur taux d'erreur est obtenu en utilisant FM sans l'ajout de Mixup. Le taux d'erreur est de 13,33 % contre 32,00 % pour un entraînement supervisé sans augmentation. Lorsqu'on compare ce résultat au meilleur entraînement supervisé obtenu qui a un taux d'erreur de 22,67 %, FM reste marginalement meilleur. La différence est de 9,34 points, soit 41,20 % d'amélioration.

— Un entraînement supervisé utilisant l'intégralité des annotations disponibles, que ce soit avec ou sans augmentation reste meilleur que n'importe quel entraînement semi-supervisé. La différence de 8,66 points reste importante malgré les bons résultats obtenus dans une configuration à 10 %.

UBS8K

— Le meilleur taux d'erreur est obtenu en utilisant MM. Le taux d'erreur est de 18,02 % contre 33,80 % pour un entraînement supervisé sans augmentation. Lorsqu'on compare ce résultat au meilleur entraînement supervisé obtenu qui a un taux d'erreur de 23,75 %, MM reste meilleur avec une différence de 5,73 points, ou 24,13 %.

— La performance de MM sur UBS8K est meilleure qu'un entraînement 100 % supervisé sans augmentation, et est très proche d'un entraînement supervisé utilisant des augmentations avec seulement une différence de 0,06 point.

GSC

— Les mêmes observations peuvent être faites sur GSC, le meilleur résultat est obtenu avec MM pour un taux d'erreur de 3,25 %. Comparé au 10,01 % ou 6,58 % obtenue avec un entraînement avec et sans augmentations, cela représente une amélioration de 6,76 et 3,33 points respectivement, soit 67,53 % et 50,61 % d'amélioration.

— Lorsqu'on compare ces résultats à ceux obtenus via un entraînement 100 % supervisé avec et sans augmentation, MM permet d'obtenir de meilleurs résultats que le premier, et est très proche du second à seulement 0,27 point.

5.4.3 L'impact de Mixup

Ajout de Mixup à MT, DCT et FM

Comme nous l'avons décrit ci-dessus, MM et RMM utilisent déjà Mixup dans leur algorithme. Afin d'en mesurer l'impact, nous avons retiré Mixup de ces deux méthodes. Les trois autres méthodes SSL explorées dans notre travail (MT, DCT, FM) n'utilisent pas Mixup dans leur version originale, mais, l'ajout de ce mécanisme n'est pas trivial et nécessite

de tester différentes configurations pour chaque algorithme SSL. Les schémas 5.1, 4.1 et 5.4 ont un bloc « Mixup » d'une couleur différente à l'endroit où le mécanisme a été rajouté. Dans tous les cas, Mixup est appliqué sur les log-Mel spectrogrammes calculés à partir des enregistrements audios.

— Pour MT, contrairement à FM, Mixup est appliqué sur les échantillons étiquetés et non-étiquetés séparément et, uniquement pour le modèle « Teacher ». C'est la formule symétrique qui est utilisée et l'augmentation « Noise », normalement appliquée aux données fournies au « Teacher », est supprimée. Mélanger les données annotées ou non entre elles résulte systématiquement à une perte de performance du système. La même chose peut être observée lorsqu'on utilise Mixup sur le modèle « Student ».

— Pour DCT, seulement les données non annotées sont modifiées avec Mixup. Elles sont communes aux deux modèles la variante symétrique est utilisée. Utiliser Mixup sur les données supervisées, conduit à de moins bons résultats.

— Après avoir rajouté Mixup,FM est très similaire à MM. C'est la variante asymétrique de la formule qui est utilisée puisqu'elle est appliquée entre des exemples annotés et non annotés. D'autres configurations, telle qu'appliquer Mixup distinctement sur l'ensemble supervisé ou non n'apporte pas d'amélioration, et peut même nuire aux performances du système.

Résultats

Dataset	ESC-10	UBS8K	GSC
MT	28,28 ± 5,28	32,80 ± 4,21	8,51
MT+Mixup	27,81 ± 2,25	32,00 ± 5,80	8,50
DCT	25,16 ± 4,42	27,85 ± 4,29	6,22
DCT+Mixup	23,75 ± 2,36	25,77 ± 4,73	5,63
MM	15,33 ± 5,58	**18,02 ± 4,00**	**3,25**
MM-Mixup	17,33 ± 3,84	20,42 ± 4,88	4,49
RMM	19,17 ± 6,52	26,02 ± 6,90	3,78
RMM-Mixup	35,33 ± 11,93	38,50 ± 5,18	4,99
FM	**13,33 ± 2,89**	21,44 ± 4,16	4,44
FM+Mixup	14,67 ± 7,21	18,27 ± 3,80	3,31

Table 5.5 – Taux d'erreur et écart type obtenus pour chaque méthode semi-supervisée pour ESC-10, UBS8K and GSC.

En retirant Mixup de MM et RMM, nous pouvons observer une perte de performance de 2, 2,4, et 1,24 point pour ESC-10, UBS8K et GSC respectivement pour MM, et 16,16, 12,48, et 1,21 point pour RMM. Ces résultats nous ont motivés à ajouter Mixup aux méthodes qui ne l'utilisaient pas déjà. Dans le tableau 5.5, nous avons indiqué les résultats obtenus en ajoutant le Mixup à MT, DCT et FM, (MT+Mixup, DCT+Mixup, FM+Mixup). Nous

donnons également les résultats obtenus en supprimant le Mixup de MM et RMM, dans les lignes nommées MM-Mixup et RMM-Mixup. En observant de plus près ce tableau, nous pouvons faire les observations suivantes :

— Pour toutes les méthodes SSL, l'ajout de Mixup a diminué le taux d'erreur d'évaluation, et ce sur tous les jeux de données. Le seul contre-exemple serait FM sur ESC où le taux d'erreur est passé de 13,33 % à 14,67 %.

— L'ajout de Mixup sur MT, DCT et FM permet d'améliorer les performances de ces méthodes sur les trois jeux de données sans pour autant modifier leur ordre. FM reste la meilleure approche pour ESC, et MM pour UBS8K et GSC.

— L'ajout du Mixup a permis d'obtenir des taux d'erreur très proches de ceux obtenues grâce à un entraînement 100 % supervisé utilisant des augmentations. On peut observer ce résultat avec FM+Mixup sur UBS8K et GSC dont la différence de taux d'erreur n'est que de 0,31 point et 0,33 point.

5.4.4 Temps d'entraînement

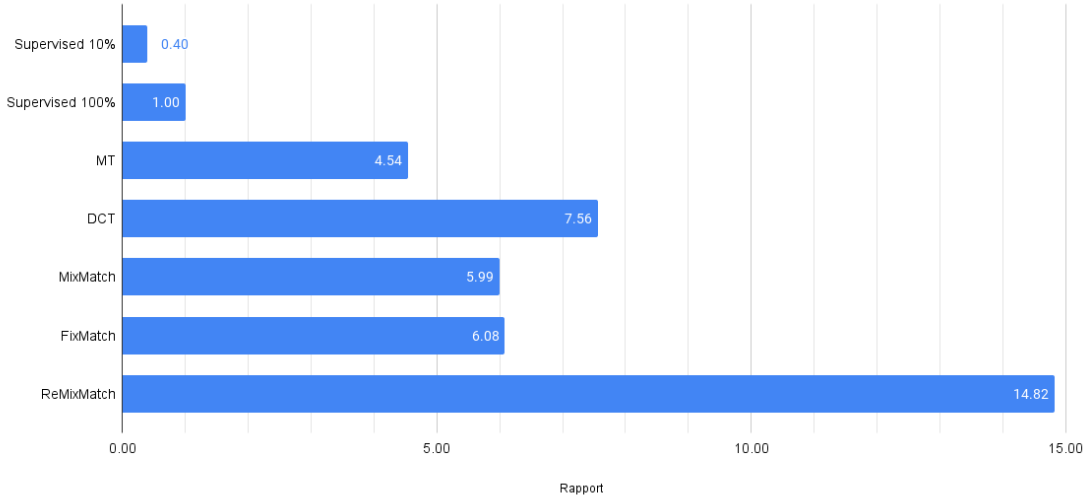

Figure 5.10 – Durées moyennes normalisées d'entraînement pour toutes les méthodes sans Mixup.

Les durées d'apprentissage normalisées pour toutes ces méthodes sont présentées dans la Fig. 5.10 et ont été calculées à l'aide de la formule suivante :

$$\text{mean} = \frac{d}{f \cdot e \cdot \text{bs}} \tag{5.24}$$

où d est la durée totale, f le nombre d'exemples dans le jeu de données, e le nombre d'époques, et bs la taille du lot utilisée dans chaque méthode. Nous utilisons le temps d'exécution supervisé de 100 % comme référence pour comparer les méthodes. Enfin, l'utilisation du Mixup a un impact négligeable sur la durée d'une époque (environ 0,5 %).

Parmi les approches SSL, la plus rapide serait MT qui est 4,5 fois plus longue que l'entraînement entièrement supervisé utilisant 100 % des données. Viennent ensuite, FM et RMM avec un facteur d'environ 6. DCT, avec sa grande complexité et l'utilisation de données adversaires, prend jusqu'à 7,5 fois plus de temps, et enfin RMM la plus longue de toute, en raison du grand nombre d'augmentations utilisées, est 15 fois plus long que l'entraînement entièrement supervisé.

5.5 Conclusion

Dans ce chapitre, plusieurs méthodes d'apprentissage semi-supervisé ont été comparées pour une tâche d'audio tagging mono-label, sur trois ensembles de données audio populaires de tailles et de contenus différents : le très petit *ESC-10* avec des évènements audio génériques, des bruits urbains avec *UrbanSound8K*, et de la parole avec *Google Speech Commands*. Pour réaliser nos expériences, nous ne gardons que 10 % des annotations pour simuler un entraînement semi-supervisé. Les résultats obtenus permettent de conclure que les performances des approches SSL holistiques (mono-modèle) sont significativement supérieures à celles de MT et DCT ainsi que celle de la référence supervisée à 10 % avec augmentation (jusqu'à 9,33 points de réduction d'erreur pour ESC avec FM).

Nous avons mesuré l'impact positif de Mixup sur MM et RMM. Son ajout aux méthodes MT, DCT et FM a apporté une amélioration substantielle dans presque tous les cas et tous les jeux de données.

On peut réduire les coûts liés à l'annotation puisque, avec seulement 10 % des labels, il est possible d'atteindre le même score qu'un entraînement supervisé augmenté utilisant 100 % des enregistrements et des annotations. Cependant, l'inconvénient de ces méthodes est leur temps d'apprentissage, car la méthode la plus rapide (mais la moins efficace) MT nécessite quatre fois plus de temps qu'un entraînement supervisé. La plus longue, RMM est même 15 fois plus lente. Si, parmi les méthodes testées dans ce travail, il y en avait une à recommander, ce serait MixMatch et éventuellement FixMatch+Mixup, qui ont obtenu les meilleurs taux d'erreur tout en gardant un temps d'entraînement raisonnable.

Dans ce travail, la distinction entre exemples annotés et non annotés est artificielle car ils font partie du même jeu de données. C'est-à-dire que ces exemples contiennent les mêmes classes que les fichiers annotés. Une perspective serait d'utiliser des exemples non-annotés hors-domaine, ce qui correspondrait à une situation plus réaliste et plus complexe.

Chapitre 6

Conclusion et perspectives

Dans ce chapitre nous résumons le travail réalisé dans le cadre de cette thèse et proposons des pistes pour continuer ce travail de recherche.

6.1 Conclusion

Dans la première partie, nous avons exploré la tâche de détection d'évènements sonores Sound Event Detection (SED) dans un cadre faiblement supervisé ainsi que les différents algorithmes de post-traitement et leur impact sur les performances des systèmes. Dans la seconde partie, nous nous sommes penchés sur l'apprentissage semi-supervisé pour la tâche de classification d'évènements sonores monophoniques et l'exploration de méthodes originellement développées pour les images. Nous nous concentrons particulièrement sur l'une d'elles, le DCT.

Le chapitre 2 définit la tâche de détection d'évènements sonores faiblement supervisée telle que présentée dans le défi DCASE, c'est-à-dire, un scénario d'un environnement domestique réaliste polyphonique. Seules des annotations faibles étaient disponibles pour effectuer des prédictions temporelles dites fortes. Dans ce chapitre, nous proposons deux approches utilisées lors de notre participation au challenge 2018. Une première intitulée Weighted Gated Recurrent Unit (WGRU) intègre un mécanisme d'oubli dans les cellules Gated Recurrent Unit (GRU) qui composent un modèle Convolutional Recurrent Neural Network (CRNN). Une seconde approche repose sur une fonction de coût Min Mean Max (MMM) qui introduit des contraintes statistiques sur les prédictions du modèle. L'approche WGRU a obtenu des résultats légèrement meilleurs que le système de référence sur l'ensemble de tests, mais n'a pas réussi à le dépasser sur l'ensemble d'évaluation. MMM a dépassé le système de référence sur les deux ensembles de tests et d'évaluation. Cependant, ces deux approches restent inférieures aux systèmes proposés par les meilleurs participants au challenge.

Le chapitre 3 propose de fournir une solution au problème du post-traitement des systèmes SED. Nous montrons que, bien qu'il soit souvent négligé, le post-traitement des prédictions temporelles permet d'augmenter les performances de manière significative. Sans faire de modifications sur l'architecture des modèles, ou de calibrage des hyperparamètres, nous comparons plusieurs algorithmes de segmentation, de lissage et d'optimisation pour permettre d'exploiter au mieux les prédictions des systèmes SED. Nous proposons aussi une boite à outils, sous la forme d'une librairie Python et open source, permettant d'appliquer ces algorithmes sur les prédictions temporelles en sortie de système SED. Nous avons appliqué ce post-traitement sur notre soumission MMM du défi 2018, et les gains apportés par le post-traitement nous auraient permis d'atteindre la seconde place. Ainsi, pour l'édition 2019, l'utilisation de ce post-traitement nous a permis d'atteindre un F-score de 39,7 % et d'atteindre la quatrième place.

Le chapitre 4 se concentre sur une méthode SSL nommée DCT avec laquelle nous réalisons nos premières tâches de classification audio de manière semi-supervisée. Nous avons décidé de nous concentrer sur DCT car il s'agissait, à ce moment-là, de la meilleure solution SSL pour la classification d'images. Nous avons alors entamé une phase exploratoire visant à modifier DCT pour l'adapter à l'audio tagging. Ainsi, après avoir reproduit le travail original de [Qiao et al., 2018] sur le jeu de données CIFAR-10, nous avons appliqué DCT sur le jeu de données UBS8K. Finalement, nous avons envisagé plusieurs axes d'amélioration possible tirant parti de l'augmentation de données. Dans une première expérience, nous équilibrons les mini lots d'entraînement en augmentant artificiellement la quantité d'échantillons supervisés par l'augmentation de données, permettant d'améliorer légèrement les performances globales du système. Dans une seconde expérience, nous essayons de remplacer la génération d'exemples adversaires par des augmentations fortes, sans succès. DCT s'est avéré difficile à implémenter, utilisant des mécanismes délicats à adapter pour des données audio. Les résultats obtenus se situent légèrement au-dessus de ceux qu'on peut atteindre avec MT, mais en deçà de nos espérances lorsqu'on les compare au gain obtenu pour une tâche de classification d'images.

Le chapitre 5 décrit et compare les méthodes SSL les plus récentes et performantes, originalement développées pour la classification d'images, lorsqu'elles sont adaptées à une tâche *d'audio tagging*. Après avoir implémenté ces approches, nous les avons appliquées sur trois jeux de données audio populaires. Ces approches peuvent être regroupées en deux catégories, une première pour les méthodes multi-modèles avec MT et DCT, et une seconde regroupant les approches dites holistiques et mono modèles nommées FM, MM et RMM. Nous avons ensuite isolé le mécanisme Mixup du MM, pour l'ajouter aux autres approches, apportant des gains importants et systématiques. Nous avons montré que toutes les méthodes SSL ont un avantage significatif dans un contexte semi-supervisé par rapport à un entraînement supervisé. Les approches holistiques permettent d'obtenir de

meilleurs résultats qu'un entraînement supervisé augmenté et approchent les résultats d'un entraînement supervisé augmenté utilisant la totalité des données annotées.

6.2 Perspectives

Apprentissage multi-tâche : sélection aléatoire d'une fonction de coût par itération

Toutes les approches SSL présentées dans cette thèse, sauf le *pseudo labelling*, nécessitent l'ajout d'au moins une nouvelle fonction de coût pour pouvoir tirer parti des enregistrements sonores non-annotés. Pour certains comme le RMM (Section 5.1.4), ou le Deep Co-Training & Teacher (DCT-T), la fonction de coût finale peut avoir jusqu'à quatre composantes différentes. Chacune de ces fonctions est très coûteuse et augmente les temps d'entraînements. En effet, les fonctions de coût calculent les gradients d'erreur pour l'ensemble du modèle indépendamment. Il faut donc les calculer autant de fois qu'il y a de composantes. Ces dernières sont pondérées par des ratios λ dont l'objectif est de diminuer leur impact lors de l'entraînement.

L'idée est alors de ne calculer qu'une de ces fonctions, sélectionnée aléatoirement, à chaque itération de l'entraînement. Les facteurs λ seront alors remplacés par une probabilité de sélection. Ainsi, plus le facteur λ est grand, plus la fonction de coût qui lui est associée à une chance d'être utilisée. Nous pourrions nommer ce mécanisme « Planificateur de coût ». Son rôle consiste à sélectionner la fonction de coût qui devra être calculée à l'itération t.

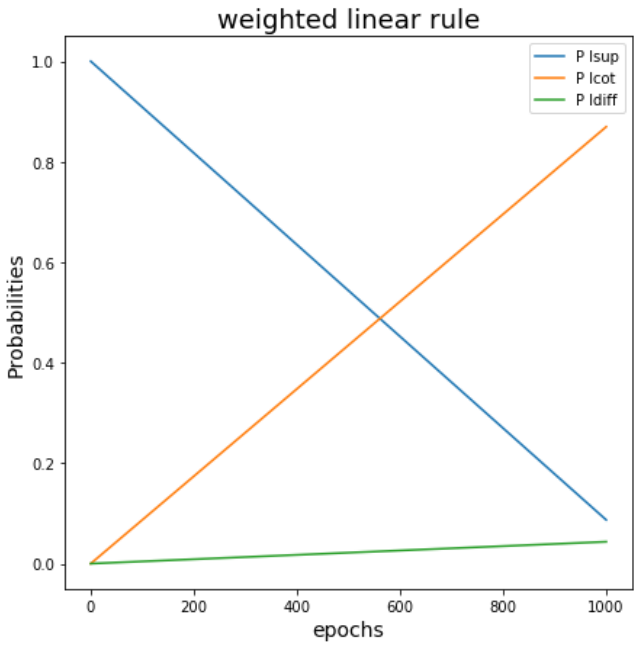

Figure 6.1 – Probabilité d'appliquer l'une des trois composantes de la fonction de coût à chaque époque.

Nous avons testé cette approche avec le DCT sur les trois jeux de données ESC-10, UBS8K et GSC (définis dans le chapitre 5, Section 5.3.2). Les résultats présentés dans le tableau 6.1 ont été obtenus après plusieurs expériences pour déterminer le meilleur planificateur. L'image 6.1 présente le planificateur *Weighted Linear Rule* qui a permis d'obtenir les meilleures performances.

	DCT (ER)	DCT-uniloss (ER)
ESC-10	17,50	16,41
UBS8K	19,49	21,54
GSC	6,16	7,98

Table 6.1 – Comparaison des taux d'erreur (ER) sur les trois jeux de données ESC-10, UBS8K, et GSC entre un entraînement DCT et sa variante *uniloss*.

Si l'Uniloss a permis d'obtenir des résultats légèrement meilleurs sur ESC-10, avec un taux d'erreur de 1,1 point inférieur, ce n'est pas le cas ni pour UBS8K ni pour GSC qui sont respectivement de 2,05 et 1,82 point supérieur. Après avoir testé de nombreux paramètres, exécuté plusieurs *grid search* et testé différents planificateurs, nous n'avons pas réussi à obtenir de meilleurs résultats et nous avons donc décidé de ne pas continuer avec ces expériences.

Combinaison de méthodes d'apprentissage semi-supervisé

Chacune des méthodes présentées dans les chapitres 4 et 5 utilisent plusieurs mécanismes d'apprentissage qui pourraient être complémentaires. C'est ce que nous a démontré l'intégration du Mixup dans les systèmes qui ne l'utilisaient pas à l'origine (voir Section 5.4.3). Nous avons eu alors l'idée de combiner un DCT avec un MT par exemple, nous appellerons cette nouvelle architecture le DCT-T :

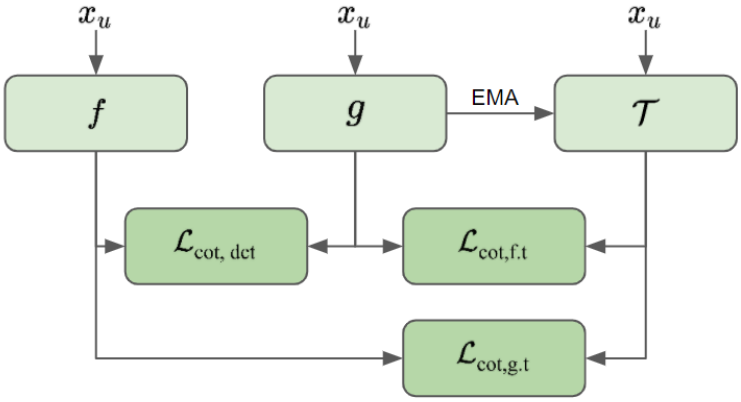

Figure 6.2 – Schéma d'une architecture de Deep Co-Training combinée avec celle d'un Mean Teacher. Seulement les parties concernées par l'utilisation des données non-annotées sont affichées pour des soucis de lisibilité.

Dans le DCT-T, dont la figure 6.2 est une représentation simplifiée, on trouve un troisième modèle, qui aura pour rôle d'être le « professeur ». Tout comme l'algorithme du MT, ses poids seront mis à jour via une Exponential Moving Average (EMA) des poids du modèle f, ou bien du modèle g (choix à définir au début de l'entraînement). Les prédictions du professeur sont alors utilisées pour un calcul de consistance supplémentaire avec f et g. La fonction de coût $\mathcal{L}_{\text{cot}}$ du DCT est alors divisée en trois composantes définies dans l'équation 6.1.

$$\begin{aligned}
\mathcal{L}_{\text{cot,f.t}} &= \text{MSE}\Big(f(x_u), T(x_u)\Big) \\
\mathcal{L}_{\text{cot,g.t}} &= \text{MSE}\Big(g(x_u), T(x_u)\Big) \\
\mathcal{L}_{\text{cot,dct}} &= \text{JS}\Big(f(x_u), g(x_u)\Big) \\
\mathcal{L} &= \mathcal{L}_{\text{sup}} + \lambda_{\text{cot}}\mathcal{L}_{\text{cot}} + \lambda_{\text{diff}}\mathcal{L}_{\text{diff}} + \lambda_{\text{t}}\Big(\mathcal{L}_{\text{cot,f.t}} + \mathcal{L}_{\text{cot,g.t}}\Big)
\end{aligned} \tag{6.1}$$

Nous avons testé cette approche sur les jeux de données ESC-10 et UBS8K, et les résultats sont présentés dans le tableau 6.2. Comparé au DCT, le résultat du DCT-T est inférieur sur ESC-10 de 2,84 points, et est statistiquement équivalent sur UBS8K. Au regard de la complexité supplémentaire apportée par l'ajout d'un modèle, l'ajout de deux fonctions de coût, et les hyperparamètres qui leur sont liés, on peut conclure que cette approche n'apporte pas de bénéfice par rapport à un DCT classique. C'est pourquoi nous avons décidé de ne pas continuer sur cette approche-là.

	Deep Co-Training (ER)	Deep Co-Training & Teacher (ER)
ESC-10	25,16 ± 4,42	28,00 ± 6,22
UBS8K	27,85 ± 4,29	27,87 ± 4,71

Table 6.2 – Taux d'erreur du DCT-T comparé au DCT classique sur les jeux de données ESC-10 et UBS8K.

Vers le semi-supervisé hors-domaine

Chacune des expériences réalisées dans cette thèse ont été faites sur des jeux de données qu'on pourrait classifier de *Benchmark*. En effet, ils sont tous entièrement annotés et nous avons simulé le contexte de semi-supervision en n'utilisant qu'une fraction des étiquettes disponible. Cela implique que les enregistrements audio « non annoté » sont assurés de contenir des évènements sonores présents aussi dans les exemples annotés.

Dans la continuation du travail réalisé dans cette thèse, il serait très intéressant d'adapter les approches SSL à des jeux de données plus représentatifs du monde réel tels que Audioset [Qiao et al., 2018] et FreeSound [Font et al., 2013]. Premièrement, le grand nombre de classes sonores représentées dans ces jeux de données permettrait de tester des configurations

semi-supervisées incluant des exemples *out of domain*. On peut alors, par exemple, exploiter ces derniers grâce à l'apprentissage autosupervisé [Liu et al., 2021]. Deuxièmement, leur nature polyphonique nécessite des modifications importantes sur les différents algorithmes SSL présentés dans les chapitres 4 et 5. Pour DCT, l'efficacité de la génération d'exemples adversaires se retrouve fortement diminuée, et les techniques telles que le *sharpen* dans MixMatch doivent être repensées pour fonctionner avec de multiples labels.

6.3 Liste de publications

Cette thèse a donné lieu à :

Deux publications en conférences internationales

— Thomas Pellegrini, et Léo Cances, « Cosine-similarity penalty to discriminate sound classes in weakly-supervised sound event detection », *International Joint Conference on Neural Networks (IJCNN)*, Budapest, Hongrie, 2019.

— Léo Cances et, Thomas Pellegrini, « Comparison of Deep Co-Training and Mean-Teacher approaches for semi-supervised audio tagging », *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Toronto, Canada, 2021.

Deux publications dans des *workshops*

— Léo Cances, Thomas Pellegrini, et Patrice Guyot, « Sound event detection from weak annotations : weighted-gru versus multi-instance-learning », *Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, Surrey, England, 2018.

— Léo Cances, Patrice Guyot, et Thomas Pellegrini, « Evaluation of post-processing algorithms for polyphonic sound event detection », *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New York, USA, 2019.

Deux rapports techniques pour nos participations au défi DCASE 2018 et 2019

— Léo Cances, Thomas Pellegrini, et Patrice Guyot, « SOUND EVENT DETECTION FROM WEAK ANNOTATIONS : WEIGHTED GRU VERSUS MULTI-INSTANCE LEARNING », *Technical report of the Detection and Classification of Acoustif Scenes and Events (DCASE)*, Surrey, England, 2018.

— Léo Cances, Thomas Pellegrini, et Patrice Guyot, « Multi task learning and post processing optimization for sound event detection », *Technical report of the Detection and Classification of Acoustic Scenes and Events (DCASE)*, New York, USA, 2019.

De plus, un article sur la comparaison des méthodes SSL pour la classification sonores a été soumis en septembre 2021 à la revue IEEE Transactions on Audio Speech and Language Processing et est en cours d'évaluation

Bibliographie

Adavanne, S. and Virtanen, T. (2017). Sound event detection using weakly labeled dataset with stacked convolutional and recurrent neural network. Technical report, DCASE2017 Challenge.

Ballan, L., Bazzica, A., Bertini, M., Del Bimbo, A., and Serra, G. (2009). Deep networks for audio event classification in soccer videos. In *Proc. ICME*, pages 474–477, Cancun. IEEE.

Barchiesi, D., Giannoulis, D., Stowell, D., and Plumbley, M. D. (2015). Acoustic scene classification : Classifying environments from the sounds they produce. *IEEE Signal Processing Magazine*, 32(3) :16–34.

Belkin, M. and Niyogi, P. (2001). Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Proceedings of the 14th International Conference on Neural Information Processing Systems : Natural and Synthetic*, NIPS'01, page 585–591, Cambridge, MA, USA. MIT Press.

Bengio, Y., Delalleau, O., and Le Roux, N. (2006). Label propagation and quadratic criterion. In *Semi-Supervised Learning*, pages 193–216. MIT Press.

Berthelot, D., Carlini, N., Cubuk, E. D., Kurakin, A., Sohn, K., Zhang, H., and Raffel, C. (2020). Remixmatch : Semi-supervised learning with distribution alignment and augmentation anchoring.

Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., and Raffel, C. (2019). Mixmatch : A holistic approach to semi-supervised learning. In *proc. NeurIPS*, pages 5049–5059, Vancouver.

Bilen, C., Ferroni, G., Tuveri, F., Azcarreta, J., and Krstulovic, S. (2020). A framework for the robust evaluation of sound event detection.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*, volume 4 of *Information science and statistics*. Springer.

Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *proc. COLT*, pages 92–100, Madison.

Cakir, E., Heittola, T., Huttunen, H., and Virtanen, T. (2015). Polyphonic sound event detection using multi label deep neural networks. In *Neural Networks (IJCNN), 2015 International Joint Conference on*, pages 1–7. IEEE.

Cakir, E., Parascandolo, G., Heittola, T., Huttunen, H., Virtanen, T., Cakir, E., Parascandolo, G., Heittola, T., Huttunen, H., and Virtanen, T. (2017). Convolutional recurrent neural networks for polyphonic sound event detection. *Proc. TASLP*, 25(6) :1291–1303.

Cakır, E. and Virtanen, T. (2019). Convolutional recurrent neural networks for rare sound event detection. *Deep Neural Networks for Sound Event Detection*, 12.

Cances, L., Pellegrini, T., and Guyot, P. (2018). Sound event detection from weak annotations : weighted GRU versus multi-instance learning. Technical report, DCASE Challenge, Woking.

Cances, L., Pellegrini, T., and Guyot, P. (2019). Multi task learning and post processing optimization for sound event detection. Technical report, DCASE Challenge, New York.

Carratino, L., Cissé, M., Jenatton, R., and Vert, J.-P. (2020). On mixup regularization.

Chou, S.-Y., Jang, J.-S., and Yang, Y.-H. (2017). FrameCNN : A weakly-supervised learning framework for frame-wise acoustic event detection and classification. Technical report, DCASE2017 Challenge.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling.

Cireșan, D. C., Meier, U., Gambardella, L. M., and Schmidhuber, J. (2010). Deep, big, simple neural nets for handwritten digit recognition. *Neural Computation*, 22(12) :3207–3220.

Crocco, M., Cristani, M., Trucco, A., and Murino, V. (2016). Audio surveillance : A systematic review. *ACM Comput. Surv.*, 48(4) :52 :1–52 :46.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. (2019). Autoaugment : Learning augmentation policies from data.

Delphin-Poulat, L. and Plapous, C. (2019). Mean teacher with data augmentation for dcase 2019 task 4. Technical report, Orange Labs Lannion, France.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet : A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

Dey, R. and Salem, F. M. (2017). Gate-variants of gated recurrent unit (gru) neural networks.

Dietterich, T. G., Lathrop, R. H., and Lozano-Pérez, T. (1997). Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 89(1-2) :31–71.

Dorfer, M. and Widmer, G. (2018). Training general-purpose audio tagging networks with noisy labels and iterative self-verification. In *Proc. DCASE - Surrey*, pages 178–182.

Ebbers, J. and Haeb-Umbach, R. (2020). Convolutional recurrent neural networks for weakly labeled semi-supervised sound event detection in domestic environments. Technical report, DCASE2020 Challenge.

Fonseca, E., Favory, X., Pons, J., Font, F., and Serra, X. (2020). Fsd50k : an open dataset of human-labeled sound events.

Font, F., Roma, G., and Serra, X. (2013). Freesound technical demo. In *ACM International Conference on Multimedia (MM'13)*, pages 411–412, Barcelona, Spain. ACM, ACM.

Fukushima, K. and Miyake, S. (1982). Neocognitron : A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern Recognition*, 15(6) :455–469.

Gammerman, A., Vovk, V., and Vapnik, V. (1998). Learning by transduction. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, page 148–155, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Gemmeke, J. F., Ellis, D. P. W., Freedman, D., Jansen, A., Lawrence, W., Moore, R. C., Plakal, M., and Ritter, M. (2017). Audio set : An ontology and human-labeled dataset for audio events. In *Proc. IEEE ICASSP 2017*, New Orleans, LA.

Goodfellow, I., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples. In *proc. ICLR*, San Diego.

Goodfellow, I. J., Bulatov, Y., Ibarz, J., Arnoud, S., and Shet, V. (2014). Multi-digit number recognition from street view imagery using deep convolutional neural networks.

Grandvalet, Y. and Bengio, Y. (2005). Semi-supervised learning by entropy minimization. In Saul, L., Weiss, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems*, volume 17, pages 529–536. MIT Press.

Guo, Y., Xu, M., Wu, J., Wang, Y., and Hoashi, K. (2018). Multi-scale convolutional recurrent neural network with ensemble method for weakly labeled sound event detection. Technical report, DCASE Challenge, Woking.

Guyot, P., Pinquier, J., and André-Obrecht, R. (2013). Water sound recognition based on physical models. In *Proc. ICASSP*, pages 793–797. IEEE.

Hao, J., Hou, Z., and Peng, W. (2020). Cross-domain sound event detection : from synthesized audio to real audio. Technical report, DCASE2020 Challenge.

Harb, R. and Pernkopf, F. (2018). Sound event detection using weakly labeled semi-supervised data with gcrnns, vat and self-adaptative label refinement. Technical report, DCASE Challenge, Woking.

Harma, A., McKinney, M. F., and Skowronek, J. (2005). Automatic surveillance of the acoustic activity in our living environment. In *Proc. Multimedia and Expo*, Amsterdam. IEEE.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition.

Heck, J. and Salem, F. M. (2017). Simplified minimal gated unit variations for recurrent neural networks.

Hinton, G. E. (1987). Learning translation invariant recognition in a massively parallel networks. In de Bakker, J. W., Nijman, A. J., and Treleaven, P. C., editors, *PARLE Parallel Architectures and Languages Europe*, pages 1–13, Berlin, Heidelberg. Springer Berlin Heidelberg.

Hinton, G. E. and Salakhutdinov, R. R. (2008). Using deep belief nets to learn covariance kernels for gaussian processes. In Platt, J., Koller, D., Singer, Y., and Roweis, S., editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc.

Hinton, G. E. and van Camp, D. (1993). Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, COLT '93, page 5–13, New York, NY, USA. Association for Computing Machinery.

Ho-Phuoc, T. (2019). Cifar10 to compare visual recognition performance between deep neural networks and humans.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9 :1735–80.

Hou, Y. and Li, S. (2018). Semi-supervised sound event detection with convolutional recurrent neural network using weakly labelled data. Technical report, DCASE Challenge, Woking.

Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. (2018). Densely connected convolutional networks.

Ioffe, S. and Szegedy, C. (2015). Batch normalization : Accelerating deep network training by reducing internal covariate shift.

JiaKai, L. (2018). Mean teacher convolution system for dcase 2018 task 4. Technical report, DCASE Challenge, 2018, Surrey.

Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, page 200–209, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Joachims, T. (2003). Transductive learning via spectral graph partitioning. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, page 290–297. AAAI Press.

Kim, N. K. and Kim, H. K. (2020). Polyphonic sound event detection based on convolutional recurrent neural networks with semi-supervised loss function for dcase challenge 2020 task 4. Technical report, DCASE2020 Challenge.

Kingma, D. P. and Ba, J. (2017). Adam : A method for stochastic optimization.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, 220 :671–680.

Koh, C.-Y., Chen, Y.-S., Li, S.-E., Liu, Y.-W., Chien, J.-T., and Bai, M. R. (2020). Sound event detection by consistency training and pseudo-labeling with feature-pyramid convolutional recurrent neural networks. Technical report, DCASE2020 Challenge.

Kong, Q., Turab, I., Yong, X., Wang, W., and Plumbley, M. D. (2018). DCASE 2018 challenge baseline with convolutional neural networks. Technical report, DCASE2018 Challenge.

Kothinti, S., Imoto, K., Chakrabarty, D., Gregory, S., Watanabe, S., and Elhilali, M. (2018). Joint acoustic and class inference for weakly supervised sound event detection. Technical report, DCASE2018 Challenge.

Koutini, K., Eghbal-zadeh, H., and Widmer, G. (2018). Iterative knowledge distillation in r-cnns for weakly-labeled semi-supervised sound event detection. Technical report, DCASE Challenge, Woking.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6) :84–90.

Krogel, M.-A. and Scheffer, T. (2004). Multi-relational learning, text mining, and semi-supervised learning for functional genomics. In *Machine Learning*, pages 61–81.

Krogh, A. and Hertz, J. A. (1991). 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. Morgan Kaufmann Publishers Inc.

Kumar, A. and Raj, B. (2016). Audio event detection using weakly labeled data. In *Proceedings of the 2016 ACM on Multimedia Conference*, pages 1038–1047. ACM.

Laine, S. and Aila, T. (2017). Temporal ensembling for semi-supervised learning.

Lecomte, S. (2013). *Classification partiellement supervisée par SVM : application à la détection d'événements en surveillance audio*. PhD thesis, Ecole doctorale Sciences pour l'Ingénieur (Troyes, Aube). Thèse de doctorat dirigée par Lengellé, Régis et Richard, Cédric Optimisation et Sûreté des Systèmes Troyes 2013.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324.

Lee, D., Lee, S., Han, Y., and Lee, K. (2017a). Ensemble of convolutional neural networks for weakly-supervised sound event detection using multiple scale input. Technical report, DCASE2017 Challenge.

Lee, D.-H. (2013). Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks. *ICML 2013 Workshop : Challenges in Representation Learning (WREPL)*.

Lee, J., Park, J., and Nam, J. (2017b). Combining multi-scale features using sample-level deep convolutional neural networks for weakly supervised sound event detection. Technical report, DCASE2017 Challenge.

Lin, L. and Wang, X. (2019). Guided learning convolution system for dcase 2019 task 4. Technical report, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China.

Liu, B., Wu, Z., Hu, H., and Lin, S. (2019). Deep metric transfer for label propagation with limited annotated data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*.

Liu, J.-Y. and Yang, Y.-H. (2016). Event localization in music auto-tagging. In *Proceedings of the 24th ACM International Conference on Multimedia*, MM '16, page 1048–1057, New York, NY, USA. Association for Computing Machinery.

Liu, X., Zhang, F., Hou, Z., Mian, L., Wang, Z., Zhang, J., and Tang, J. (2021). Self-supervised learning : Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering*, page 1–1.

Liu, Y. L., Yan, J., Song, Y., and Du, J. (2018). Ustc-nelslip system for dcase 2018 challenge task 4. Technical report, DCASE2018 Challenge.

Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization.

Lu, K., Foo, C.-S., Teh, K. K., Tran, H. D., and Chandrasekhar, V. R. (2019). Semi-supervised audio classification with consistency-based regularization. In *proc. INTERSPEECH*, pages 3654–3658, Graz.

Mesaros, A., Heittola, T., Eronen, A., and Virtanen, T. (2010). Acoustic event detection in real life recordings. In *2010 18th European Signal Processing Conference*, pages 1267–1271.

Mesaros, A., Heittola, T., Virtanen, T., Mesaros, A., Heittola, T., and Virtanen, T. (2016). Metrics for Polyphonic Sound Event Detection. *Applied Sciences*, 6(6) :162.

Miech, A., Laptev, I., and Sivic, J. (2018). Learnable pooling with context gating for video classification.

Miyato, T., ichi Maeda, S., Koyama, M., and Ishii, S. (2018). Virtual adversarial training : A regularization method for supervised and semi-supervised learning.

Miyazaki, K., Komatsu, T., Hayashi, T., Watanabe, S., Toda, T., and Takeda, K. (2020). Convolution-augmented transformer for semi-supervised sound event detection. Technical report, DCASE2020 Challenge.

Morfi, V. and Stowell, D. (2018). Data-efficient weakly supervised learning for low-resource audio event detection using deep learning. *arXiv preprint arXiv :1807.06972*.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*.

Nguyen, T. N. T., Nguyen, N. K., Jones, D. L., and Gan, W. S. (2018). DCASE 2018 task 2 : iterative training, label smoothing, and background noise normalization for audio event tagging. In *Proc. DCASE Workshop - Surrey*, pages 54–58.

Odena, A. (2016). Semi-supervised learning with generative adversarial networks.

Parascandolo, G., Huttunen, H., and Virtanen, T. (2016a). Recurrent neural networks for polyphonic sound event detection in real life recordings. *arXiv preprint arXiv :1604.00861*.

Parascandolo, G., Huttunen, H., and Virtanen, T. (2016b). Recurrent neural networks for polyphonic sound event detection in real life recordings. In *Proc. ICASSP*, pages 6440–6444, Shanghai. IEEE.

Park, D. S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E. D., and Le, Q. V. (2019). Specaugment : A simple data augmentation method for automatic speech recognition. *Interspeech 2019*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch : An imperative style, high-performance deep learning library.

Pellegrini, T. and Cances, L. (2019). Cosine-similarity penalty to discriminate sound classes in weakly-supervised sound event detection. In *Proc. IJCNN*, Budapest. IEEE.

Pellegrini, T. and Masquelier, T. (2021). Fast threshold optimization for multi-label audio tagging using Surrogate gradient learning. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, Toronto, Canada.

Piczak, K. J. (2015). Esc : Dataset for environmental sound classification. In *proc. ACM Multimedia*, page 1015–1018, Brisbane.

Qiao, S., Shen, W., Zhang, Z., Wang, B., and Yuille, A. (2018). Deep co-training for semi-supervised image recognition. In *proc. ECCV*, pages 135–152, Munich.

Rasmus, A., Valpola, H., Honkala, M., Berglund, M., and Raiko, T. (2015). Semi-supervised learning with ladder networks.

Sajjadi, M., Javanmardi, M., and Tasdizen, T. (2016). Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29, pages 1163–1171. Curran Associates, Inc.

Salamon, J. and Bello, J. P. (2017). Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, 24(3) :279–283.

Salamon, J., Jacoby, C., and Bello, J. P. (2014). A dataset and taxonomy for urban sound research. In *proc. ACM Multimedia*, page 1041–1044.

Salamon, J., McFee, B., Li, P., and Bello, J. P. (2017). Multiple instance learning for sound event detection. Technical report, DCASE Challenge, Munich.

Serizel, R., Turpault, N., Eghbal-Zadeh, H., and Parag Shah, A. (2018). Large-scale weakly labeled semi-supervised sound event detection in domestic environments. In *Proc. DCASE*, Woking.

Shi, Z. (2019). Hodgepodge : Sound event detection based on ensemble of semi-supervised learning methods. Technical report, Fujitsu Research and Development Center, Beijing, China.

Simard, P., Steinkraus, D., and Platt, J. C. (2003). Best practices for convolutional neural networks applied to visual document analysis. *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, pages 958–963.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition.

Sohn, K., Berthelot, D., Li, C.-L., Zhang, Z., Carlini, N., Cubuk, E. D., Kurakin, A., Zhang, H., and Raffel, C. (2020). Fixmatch : Simplifying semi-supervised learning with consistency and confidence.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout : A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*.

Stowell, D., Wood, M., Stylianou, Y., and Glotin, H. (2016). Bird detection in audio : a survey and a challenge. *arXiv preprint arXiv :1608.03417*.

Su, T.-W., Liu, J.-Y., and Yang, Y.-H. (2017). Weakly-supervised audio event detection using event-specific gaussian filters and fully convolutional networks. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 791–795.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2014). Going deeper with convolutions.

Tarvainen, A. and Valpola, H. (2017). Mean teachers are better role models : Weight-averaged consistency targets improve semi-supervised deep learning results. In *proc. NeurIPS*, pages 1195–1204, Long Beach.

Tarvainen, A. and Valpola, H. (2018). Mean teachers are better role models : Weight-averaged consistency targets improve semi-supervised deep learning results.

Turpault, N., Serizel, R., Parag Shah, A., and Salamon, J. (2019). Sound event detection in domestic environments with weakly labeled data and soundscape synthesis. Technical report, Université de Lorraine, CNRS, Inria, Loria, France. working paper or preprint.

Valpola, H. (2015). From neural pca to deep unsupervised learning.

Verma, V., Kawaguchi, K., Lamb, A., Kannala, J., Bengio, Y., and Lopez-Paz, D. (2020). Interpolation consistency training for semi-supervised learning.

Virtanen, T., Plumbley, M. D., and Ellis, D. (2018). *Computational analysis of sound scenes and events*. Springer.

Warden, P. (2018, arxiv :1804.03209). Speech commands : A dataset for limited-vocabulary speech recognition.

Wu, Y., Liu, L., Bae, J., Chow, K.-H., Iyengar, A., Pu, C., Wei, W., Yu, L., and Zhang, Q. (2019). Demystifying learning rate policies for high accuracy training of deep neural networks.

Xia, X., Togneri, R., Sohel, F., and Huang, D. (2017). Frame-wise dynamic threshold based polyphonic acoustic event detection. In *Proc. Interspeech*, pages 474–478, Stockholm.

Yao, T., Shi, C., and Li, H. (2020). Sound event detection in domestic environments using dense recurrent neural network. Technical report, DCASE2020 Challenge.

Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. In *proc. BMVC*, pages 87.1–87.12, York.

Zhang, G., Wang, C., Xu, B., and Grosse, R. (2018a). Three mechanisms of weight decay regularization.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. (2018b). mixup : Beyond empirical risk minimization.

Zheng, X., Chen, H., and Song, Y. (2021). Zheng ustc team's submission for dcase2021 task4 – semi-supervised sound event detection. Technical report, DCASE2021 Challenge.

Zhu, X., Ghahramani, Z., and Lafferty, J. (2003). Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, page 912–919. AAAI Press.

Annexe A

Identifiant d'augmentation

Fait référence à l'image 4.13

augmentation identifiant	augmentation	paramètres
flip_ud	VerticalFlip	ratio=1.0
flip_lr	HorizontalFlip	ratio=1.0
usn_1	UniformSignNoise	ratio=1.0, epsilon=0.001, mini=-80, maxi=0
usn_2	UniformSignNoise	ratio=1.0, epsilon=0.005, mini=-80, maxi=0
usn_3	UniformSignNoise	ratio=1.0, epsilon=0.01, mini=-80, maxi=0
usn_4	UniformSignNoise	ratio=1.0, epsilon=0.02, mini=-80, maxi=0
usn_5	UniformSignNoise	ratio=1.0, epsilon=0.03, mini=-80, maxi=0
usn_6	UniformSignNoise	ratio=1.0, epsilon=0.04, mini=-80, maxi=0
usn_7	UniformSignNoise	ratio=1.0, epsilon=0.05, mini=-80, maxi=0
usn_8	UniformSignNoise	ratio=1.0, epsilon=0.06, mini=-80, maxi=0
usn_9	UniformSignNoise	ratio=1.0, epsilon=0.07, mini=-80, maxi=0
usn_10	UniformSignNoise	ratio=1.0, epsilon=0.08, mini=-80, maxi=0
usn_11	UniformSignNoise	ratio=1.0, epsilon=0.09, mini=-80, maxi=0
usn_12	UniformSignNoise	ratio=1.0, epsilon=0.1, mini=-80, maxi=0
usn_13	UniformSignNoise	ratio=1.0, epsilon=0.2, mini=-80, maxi=0
usn_14	UniformSignNoise	ratio=1.0, epsilon=0.3, mini=-80, maxi=0
usn_15	UniformSignNoise	ratio=1.0, epsilon=0.4, mini=-80, maxi=0
usn_16	UniformSignNoise	ratio=1.0, epsilon=0.5, mini=-80, maxi=0
usn_17	UniformSignNoise	ratio=1.0, epsilon=0.6, mini=-80, maxi=0
usn_18	UniformSignNoise	ratio=1.0, epsilon=0.7, mini=-80, maxi=0
usn_19	UniformSignNoise	ratio=1.0, epsilon=0.8, mini=-80, maxi=0
usn_20	UniformSignNoise	ratio=1.0, epsilon=0.9, mini=-80, maxi=0
usn_21	UniformSignNoise	ratio=1.0, epsilon=1.0, mini=-80, maxi=0
usn_22	UniformSignNoise	ratio=1.0, epsilon=2.0, mini=-80, maxi=0
usn_23	UniformSignNoise	ratio=1.0, epsilon=3.0, mini=-80, maxi=0
usn_24	UniformSignNoise	ratio=1.0, epsilon=4.0, mini=-80, maxi=0

usn_25	UniformSignNoise	ratio=1.0, epsilon=5.0, mini=-80, maxi=0
usn_26	UniformSignNoise	ratio=1.0, epsilon=6.0, mini=-80, maxi=0
usn_27	UniformSignNoise	ratio=1.0, epsilon=7.0, mini=-80, maxi=0
usn_28	UniformSignNoise	ratio=1.0, epsilon=8.0, mini=-80, maxi=0
usn_29	UniformSignNoise	ratio=1.0, epsilon=9.0, mini=-80, maxi=0
n_10	Noise	ratio=1.0, snr=10, mini=-80, maxi=0
n_15	Noise	ratio=1.0, snr=15, mini=-80, maxi=0
n_20	Noise	ratio=1.0, snr=20, mini=-80, maxi=0
n_25	Noise	ratio=1.0, snr=25, mini=-80, maxi=0
n_30	Noise	ratio=1.0, snr=30, mini=-80, maxi=0
n_40	Noise	ratio=1.0, snr=40, mini=-80, maxi=0
n_10_05	Noise	ratio=0.5, snr=10, mini=-80, maxi=0
n_15_05	Noise	ratio=0.5, snr=15, mini=-80, maxi=0
n_20_05	Noise	ratio=0.5, snr=20, mini=-80, maxi=0
n_25_05	Noise	ratio=0.5, snr=25, mini=-80, maxi=0
n_30_05	Noise	ratio=0.5, snr=30, mini=-80, maxi=0
n_40_05	Noise	ratio=0.5, snr=40, mini=-80, maxi=0
fts_1	FractalTimeStretch	ratio=1.0, min_chunk_size=2, max_chunk_size=4, rates=(0.8, 1.2)
fts_2	FractalTimeStretch	ratio=1.0, min_chunk_size=2, max_chunk_size=4, rates=(0.8, 1.2)
fts_3	FractalTimeStretch	ratio=1.0, min_chunk_size=2, max_chunk_size=8, rates=(0.8, 1.2)
fts_4	FractalTimeStretch	ratio=1.0, min_chunk_size=2, max_chunk_size=16, rates=(0.8, 1.2)
fts_5	FractalTimeStretch	ratio=1.0, min_chunk_size=2, max_chunk_size=32, rates=(0.8, 1.2)
fts_51	FractalTimeStretch	ratio=1.0, min_chunk_size=4, max_chunk_size=32, rates=(0.8, 1.2)
fts_52	FractalTimeStretch	ratio=1.0, min_chunk_size=8, max_chunk_size=32, rates=(0.8, 1.2)
fts_53	FractalTimeStretch	ratio=1.0, min_chunk_size=12, max_chunk_size=32, rates=(0.8, 1.2)
fts_54	FractalTimeStretch	ratio=1.0, min_chunk_size=16, max_chunk_size=32, rates=(0.8, 1.2)
fts_55	FractalTimeStretch	ratio=1.0, min_chunk_size=20, max_chunk_size=32, rates=(0.8, 1.2)
ffs_1	FractalFreqStretch	ratio=1.0, min_chunk_size=2, max_chunk_size=4, intra_ratio=0.1, rate=(0.8, 1.2)

ffs_2	FractalFreqStretch	ratio=1.0, min_chunk_size=2, max_chunk_size=8, intra_ratio=0.2, rate=(0.8, 1.2)
ffs_3	FractalFreqStretch	ratio=1.0, min_chunk_size=2, max_chunk_size=12, intra_ratio=0.3, rate=(0.8, 1.2)
ffs_4	FractalFreqStretch	ratio=1.0, min_chunk_size=2, max_chunk_size=16, intra_ratio=0.4, rate=(0.8, 1.2)
ffs_5	FractalFreqStretch	ratio=1.0, min_chunk_size=2, max_chunk_size=20, intra_ratio=0.5, rate=(0.8, 1.2)
ffs_51	FractalFreqStretch	ratio=1.0, min_chunk_size=6, max_chunk_size=20, intra_ratio=0.5, rate=(0.8, 1.2)
ffs_52	FractalFreqStretch	ratio=1.0, min_chunk_size=6, max_chunk_size=20, intra_ratio=0.5, rate=(0.7, 1.3)
ffs_53	FractalFreqStretch	ratio=1.0, min_chunk_size=6, max_chunk_size=20, intra_ratio=0.5, rate=(0.6, 1.4)
ffs_54	FractalFreqStretch	ratio=1.0, min_chunk_size=6, max_chunk_size=20, intra_ratio=0.5, rate=(0.5, 1.5)
ffs_55	FractalFreqStretch	ratio=1.0, min_chunk_size=6, max_chunk_size=20, intra_ratio=0.5, rate=(0.4, 1.6)
ftd_1	FractalTimeDropout	ratio=1.0, min_chunk_size=1, max_chunk_size=2, min_chunk=1, max_chunk=3, void=True
ftd_2	FractalTimeDropout	ratio=1.0, min_chunk_size=2, max_chunk_size=4, min_chunk=1, max_chunk=3, void=True
ftd_3	FractalTimeDropout	ratio=1.0, min_chunk_size=2, max_chunk_size=8, min_chunk=1, max_chunk=3, void=True
ftd_4	FractalTimeDropout	ratio=1.0, min_chunk_size=2, max_chunk_size=16, min_chunk=1, max_chunk=3, void=True
ftd_5	FractalTimeDropout	ratio=1.0, min_chunk_size=2, max_chunk_size=32, min_chunk=1, max_chunk=3, void=True
ffd_1	FractalFrecDropout	ratio=1.0, min_chunk_size=1, max_chunk_size=2, min_chunk=1, max_chunk=3, void=True
ffd_2	FractalFrecDropout	ratio=1.0, min_chunk_size=2, max_chunk_size=4, min_chunk=1, max_chunk=3, void=True
ffd_3	FractalFrecDropout	ratio=1.0, min_chunk_size=2, max_chunk_size=8, min_chunk=1, max_chunk=3, void=True
ffd_4	FractalFrecDropout	ratio=1.0, min_chunk_size=2, max_chunk_size=10, min_chunk=1, max_chunk=3, void=True
ffd_5	FractalFrecDropout	ratio=1.0, min_chunk_size=2, max_chunk_size=12, min_chunk=1, max_chunk=3, void=True
rtd_1	RandomTimeDropout	ratio=1.0, dropout=0.01
rtd_2	RandomTimeDropout	ratio=1.0, dropout=0.05
rtd_3	RandomTimeDropout	ratio=1.0, dropout=0.1
rtd_4	RandomTimeDropout	ratio=1.0, dropout=0.2
rtd_5	RandomTimeDropout	ratio=1.0, dropout=0.3
rtd_6	RandomTimeDropout	ratio=1.0, dropout=0.4
rtd_7	RandomTimeDropout	ratio=1.0, dropout=0.5
rtd_8	RandomTimeDropout	ratio=1.0, dropout=0.6
rtd_9	RandomTimeDropout	ratio=1.0, dropout=0.7

rtd_10	RandomTimeDropout	ratio=1.0, dropout=0.8
rtd_11	RandomTimeDropout	ratio=1.0, dropout=0.9
rfd_1	RandomFreqDropout	ratio=1.0, dropout=0.01
rfd_2	RandomFreqDropout	ratio=1.0, dropout=0.05
rfd_3	RandomFreqDropout	ratio=1.0, dropout=0.1
rfd_4	RandomFreqDropout	ratio=1.0, dropout=0.2
rfd_5	RandomFreqDropout	ratio=1.0, dropout=0.3
rfd_6	RandomFreqDropout	ratio=1.0, dropout=0.4
rfd_7	RandomFreqDropout	ratio=1.0, dropout=0.5
rfd_8	RandomFreqDropout	ratio=1.0, dropout=0.6
rfd_9	RandomFreqDropout	ratio=1.0, dropout=0.7
rfd_10	RandomFreqDropout	ratio=1.0, dropout=0.8
rfd_11	RandomFreqDropout	ratio=1.0, dropout=0.9
s_ts_1	TimeStretch	ratio=1.0, rate=(0.95, 1.05)
s_ts_2	TimeStretch	ratio=1.0, rate=(0.9, 1.1)
s_ts_3	TimeStretch	ratio=1.0, rate=(0.85, 1.15)
s_ts_4	TimeStretch	ratio=1.0, rate=(0.8, 1.2)
s_ts_5	TimeStretch	ratio=1.0, rate=(0.75, 1.25)
s_psr_1	PitchShiftRandom	ratio=1.0, sampling_rate=22050, steps=(-1, 1)
s_psr_2	PitchShiftRandom	ratio=1.0, sampling_rate=22050, steps=(-2, 2)
s_psr_3	PitchShiftRandom	ratio=1.0, sampling_rate=22050, steps=(-3, 3)
s_psr_4	PitchShiftRandom	ratio=1.0, sampling_rate=22050, steps=(-4, 4)
s_psr_5	PitchShiftRandom	ratio=1.0, sampling_rate=22050, steps=(-5, 5)
s_l_1	Level	ratio=1.0, rate=(0.95, 1.05)
s_l_2	Level	ratio=1.0, rate=(0.9, 1.1)
s_l_3	Level	ratio=1.0, rate=(0.85, 1.15)
s_l_4	Level	ratio=1.0, rate=(0.8, 1.2)

s_l_5	Level	ratio=1.0, rate=(0.75, 1.25)
s_o_1	Occlusion	ratio=1.0, sampling_rate=22050, max_size=0.1, max_occlu_size=2205, dim=0
s_o_2	Occlusion	ratio=1.0, sampling_rate=22050, max_size=0.2, max_occlu_size=4410, dim=0
s_o_3	Occlusion	ratio=1.0, sampling_rate=22050, max_size=0.3, max_occlu_size=6615, dim=0
s_o_4	Occlusion	ratio=1.0, sampling_rate=22050, max_size=0.4, max_occlu_size=8820, dim=0
s_o_5	Occlusion	ratio=1.0, sampling_rate=22050, max_size=0.5, max_occlu_size=11025, dim=0
s_o_6	Occlusion	ratio=1.0, sampling_rate=22050, max_size=0.6, max_occlu_size=13230, dim=0
s_o_7	Occlusion	ratio=1.0, sampling_rate=22050, max_size=0.7, max_occlu_size=15434, dim=0
s_o_8	Occlusion	ratio=1.0, sampling_rate=22050, max_size=0.8, max_occlu_size=17640, dim=0
s_o_9	Occlusion	ratio=1.0, sampling_rate=22050, max_size=0.9, max_occlu_size=19845, dim=0
s_o_10	Occlusion	ratio=1.0, sampling_rate=22050, max_size=1.0, max_occlu_size=22050, dim=0
s_c_1	Clip	ratio=1.0, range=(-0.95, 0.95)
s_c_2	Clip	ratio=1.0, range=(-0.9, 0.9)
s_c_3	Clip	ratio=1.0, range=(-0.85, 0.85)
s_c_4	Clip	ratio=1.0, range=(-0.8, 0.8)
s_c_5	Clip	ratio=1.0, range=(-0.75, 0.75)
s_n_5	Noise	ratio=1.0, target_snr=5
s_n_10	Noise	ratio=1.0, target_snr=10
s_n_15	Noise	ratio=1.0, target_snr=15
s_n_20	Noise	ratio=1.0, target_snr=20
s_n_25	Noise	ratio=1.0, target_snr=25
s_n_30	Noise	ratio=1.0, target_snr=30
s_n_35	Noise	ratio=1.0, target_snr=35
s_n_40	Noise	ratio=1.0, target_snr=40

Annexe B

Fonctions de similarités

Cohérence

Le score de « cohérence ». Il consiste à multiplier le spectrogramme original x et celui obtenu après augmentation y puis de diviser le résultat par le produit des normes de ces spectrogrammes. Plus le score est proche de 0, plus les spectrogrammes se ressemblent.

$$\text{cohérence} = \frac{x \times y}{|x| \times |y|} \tag{B.1}$$

Distance temporelle

La distance temporelle : Cette métrique permet d'établir la différence entre les deux spectrogrammes à chaque instant t du signal. Cela permet de localiser la différence d'un point de vue temporelle. Plus le score est proche de 0, les spectrogrammes se ressemblent.

$$\text{distance} = \frac{1}{t}(\sum_{i=0}^{t} x_i - \sum_{i=0}^{t} y_i) \tag{B.2}$$

Distance d'histogrammes

La distance d'histogramme entre les histogrammes des deux spectrogrammes. Cela permet d'être robuste au décalage temporel ou fréquentiel. Plus le score est proche de 0, plus les spectrogrammes se ressemblent.

$$\text{similarité} = 1 - \text{hist}(x) - \text{hist}(y) \tag{B.3}$$

Root Mean Square Error (RMSE)

RMSE. Permet de calculer une distance entre chaque valeur du spectrogramme, puis de la moyenner. Plus le score est proche de 0, plus les spectrogrammes se ressemblent.

$$\text{similarity} = \sqrt{(x-y)^2} \tag{B.4}$$