Sorbonne Université

Faculté des Sciences et Ingénierie

Master Informatique – M1 Parcours Données, Apprentissage, Connaissance

Projet logiciel et recherche

DAC

« Chants d'oiseaux »

Encadrant : SCHWANDER Olivier

DELEFOSSE Aymeric

BENCHECI Valentin

Année 2022-2023

Table des matières

Introduction

Contexte et motivation

La biodiversité et la conservation de la faune sont des préoccupations majeures dans le monde entier. Les oiseaux jouent un rôle crucial dans les écosystèmes, et l'étude de leurs comportements et de leurs vocalisations peut fournir des informations précieuses sur l'état de l'environnement. Cependant, l'identification manuelle des chants d'oiseaux à partir de vastes enregistrements audios est une tâche longue et fastidieuse.

Dans ce contexte, le développement d'un système de classification automatique des audios basé sur la présence ou l'absence de chants d'oiseaux présente de nombreux avantages. Une telle solution permettrait d'accélérer l'analyse des enregistrements audio, d'identifier rapidement les zones où les oiseaux sont présents et de faciliter la collecte de données pour les études de biodiversité.

Objectif du projet

L'objectif principal de ce projet est de concevoir et de développer un système de classification automatique des audios pour distinguer les enregistrements contenant des chants d'oiseaux de ceux qui n'en contiennent pas. L'approche utilisée est basée sur l'apprentissage automatique (*machine learning*) en combinant différentes techniques de prétraitement des audios et l'utilisation de modèles de classification.

Plus spécifiquement, les objectifs du projet sont les suivants :

1. Collecter et préparer un ensemble de données audio comprenant des enregistrements avec et sans chants d'oiseaux à partir de différentes sources, tels que les datasets Freefield, Warblr et BirdVox.
2. Explorer et comparer trois approches de prétraitement des audios : une division des audios en *chunks* de 2 secondes avec suppression de bruit, une suppression de bruit directement appliquée aux audios originaux et une représentation mel-spectrogramme des audios.
3. Utiliser quatre modèles d'apprentissage différents : un réseau de neurones convolutif (CNN), un réseau de neurones récurrent (RNN), un réseau de neurones « hybride » (CRNN) et une machine à vecteurs de support (SVM) pour la classification des audios.
4. Évaluer et comparer les performances de chaque modèle en termes de précision, de rappel et de F-mesure.

5. Analyser les résultats obtenus et identifier les forces et les faiblesses de chaque approche pour la classification des audios.
6. Fournir des recommandations et des perspectives d'amélioration pour de futures recherches dans le domaine de la classification automatique des audios basée sur les chants d'oiseaux.

État de l'art

Nombre de méthodes de *machine learning* et de *deep learning* ont été proposées au fil des ans pour la classification des chants d'oiseaux. Cet état de l'art se concentre sur les techniques de détection de sons d'oiseaux et les situe dans le contexte de la classification binaire de la présence de sons d'oiseaux.

1. Modèles de machine learning

Les modèles de machine learning traditionnels ont longtemps été utilisés pour la détection des sons d'oiseaux. Des méthodes comme les Gaussian Mixture Models (GMM) et les Support Vector Machines (SVM) ont été particulièrement populaires.

a. Les GMM [9] sont des modèles probabilistes qui supposent que les données sont générées à partir d'un mélange de plusieurs distributions gaussiennes. En bioacoustique, ces modèles ont été utilisés pour modéliser la distribution des caractéristiques de signal extraites des enregistrements de sons d'oiseaux. Les caractéristiques couramment utilisées incluent le Mel-frequency cepstral coefficients (MFCC), qui capturent les caractéristiques spectrales des sons d'oiseaux.

b. Les SVM [9] sont des modèles de classification qui cherchent à trouver un hyperplan qui sépare le mieux les classes dans l'espace des caractéristiques. Dans la détection de sons d'oiseaux, les SVM ont été utilisés avec diverses caractéristiques du signal, comme les MFCC, pour classer les enregistrements comme contenant des sons d'oiseaux ou non.

2. Modèles de deep learning

Ces modèles ont l'avantage de pouvoir apprendre des représentations de caractéristiques complexes à partir des données brutes, ce qui peut conduire à de meilleures performances de détection. [3–7,10]

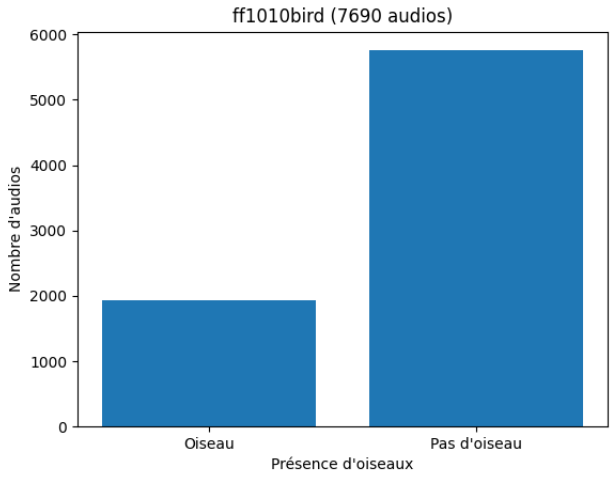
a. Convolutional Neural Networks (CNN) : Les CNN sont particulièrement efficaces pour le traitement de données avec une structure de grille, comme les images ou les spectrogrammes. Dans la détection de sons d'oiseaux, les CNN ont été utilisés pour classer les spectrogrammes des enregistrements de sons, avec des résultats prometteurs [3–7,10].
b. Recurrent Neural Networks (RNN) et Long Short-Term Memory (LSTM) : Les RNN et LSTM sont des types de réseaux de neurones qui sont efficaces pour traiter des séquences de données, comme les séquences temporelles dans les enregistrements de sons d'oiseaux. Ces modèles ont été utilisés pour classer les séquences de caractéristiques extraites des enregistrements de sons d'oiseaux. De plus, ils sont parfois employés avec des CNN [1,2].
c. Transformers : ces modèles d'embedding ont également été appliqués à la détection des sons d'oiseaux. Ces modèles peuvent capter les dépendances à long terme dans les données, ce qui peut être utile pour la détection des séquences de chants d'oiseaux [12].

Description des datasets utilisés (Freefield, Warblr, BirdVox)

1. Freefield

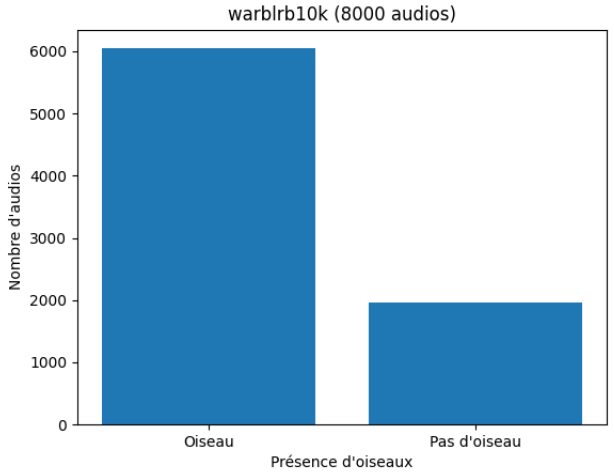
Freefield est un ensemble de données audio librement accessible et largement utilisé dans la communauté de la recherche en bioacoustique. Il contient des enregistrements de sons naturels provenant de différentes sources, y compris des chants d'oiseaux. Les enregistrements sont collectés à partir de diverses localisations

géographiques et peuvent présenter une grande variété de conditions acoustiques. Freefield fournit une ressource précieuse pour l'étude des sons de la nature, y compris les chants d'oiseaux.



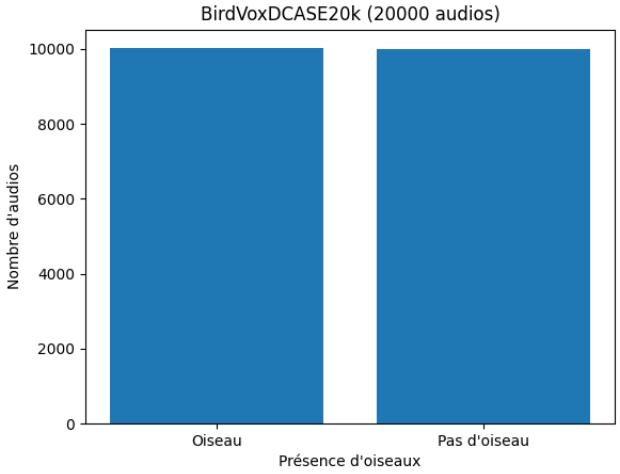
2. Warblr

Warblr est un dataset spécifiquement conçu pour la classification automatique des chants d'oiseaux. Il comprend une vaste collection d'enregistrements audio étiquetés avec des informations sur les espèces d'oiseaux présentes. Warblr a été créé en utilisant une application mobile qui permet aux utilisateurs d'enregistrer des chants d'oiseaux dans leur environnement. Les enregistrements ont été soumis à un processus de vérification et de validation pour garantir leur qualité et leur exactitude. Ce dataset constitue une ressource précieuse pour entraîner des modèles de classification des chants d'oiseaux.



3. BirdVox

BirdVox est un autre dataset largement utilisé dans la recherche en bioacoustique. Il se concentre spécifiquement sur les enregistrements des migrations d'oiseaux pendant la nuit, lorsque les oiseaux émettent des appels spécifiques. BirdVox comprend des enregistrements audios de plusieurs stations d'observation réparties géographiquement. Les enregistrements sont étiquetés avec des informations détaillées sur les espèces d'oiseaux et les événements d'appels. Ce dataset fournit une ressource précieuse pour l'étude des migrations nocturnes des oiseaux et peut être utilisé pour la classification des chants d'oiseaux.



Prétraitement des audios

Le prétraitement des audios est une étape cruciale dans le traitement des signaux sonores. Il permet d'améliorer la qualité de l'audio en éliminant les bruits indésirables et en divisant les signaux en segments plus gérables.

Première approche

Nous aborderons un algorithme de suppression de bruit utilisant les opérations d'érosion et de dilation, ainsi que la division des audios en chunks de 2 secondes [8].

1. **Suppression de bruit avec érosion et dilation :**

L'algorithme de suppression de bruit utilisé dans cette étude combine l'érosion et la dilation pour extraire un masque binaire du spectrogramme de l'audio. Voici comment cela fonctionne :

Étape 1 : Calcul du spectrogramme

Le spectrogramme est une représentation visuelle des fréquences présentes dans un signal sonore. Il est obtenu en appliquant la transformée de Fourier à court terme (STFT) sur l'audio. Cela permet de diviser le signal en petites fenêtres de temps et de visualiser les composantes fréquentielles à chaque instant.

Étape 2 : Calcul des médianes des lignes et des colonnes

Pour estimer le bruit de fond dans le spectrogramme, des médianes des lignes et des colonnes sont calculées. Ces médianes servent de seuils pour déterminer quels éléments du spectrogramme sont considérés comme du bruit et quels éléments sont considérés comme du signal.

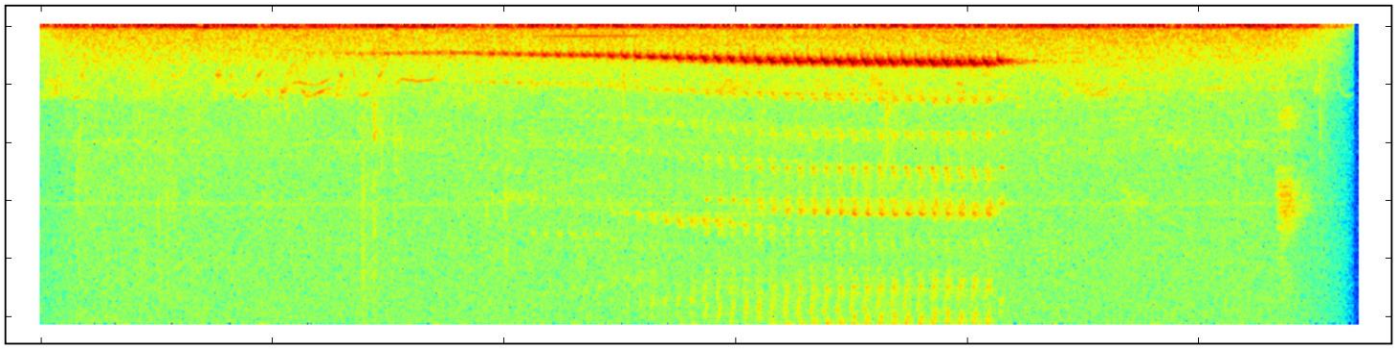
Étape 3 : Extraction du masque binaire

Le masque binaire est extrait en comparant chaque élément du spectrogramme avec les seuils définis par les médianes. Les valeurs qui dépassent ces seuils sont considérées comme du signal, tandis que les autres sont considérées comme du bruit. Le masque résultant est ensuite soumis à des opérations d'érosion et de dilation pour améliorer la séparation entre le signal et le bruit.

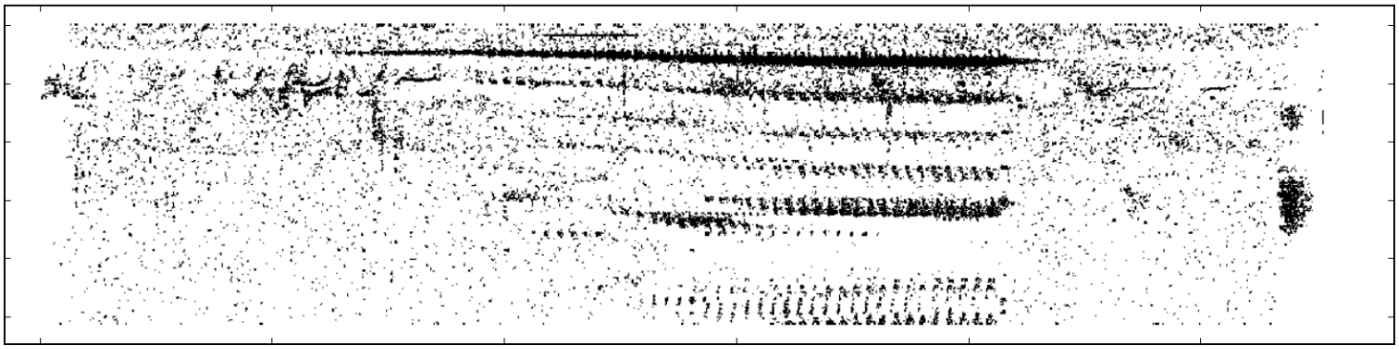
Étape 4 : Érosion et dilation

L'érosion et la dilation sont des opérations morphologiques qui permettent de modifier la forme des objets binaires dans une image. Dans le contexte de la suppression de bruit, ces opérations sont utilisées pour améliorer la cohérence spatiale du masque binaire. L'érosion réduit la taille des objets binaires et les discontinuités, tandis que la dilation les agrandit et comble les petites lacunes.

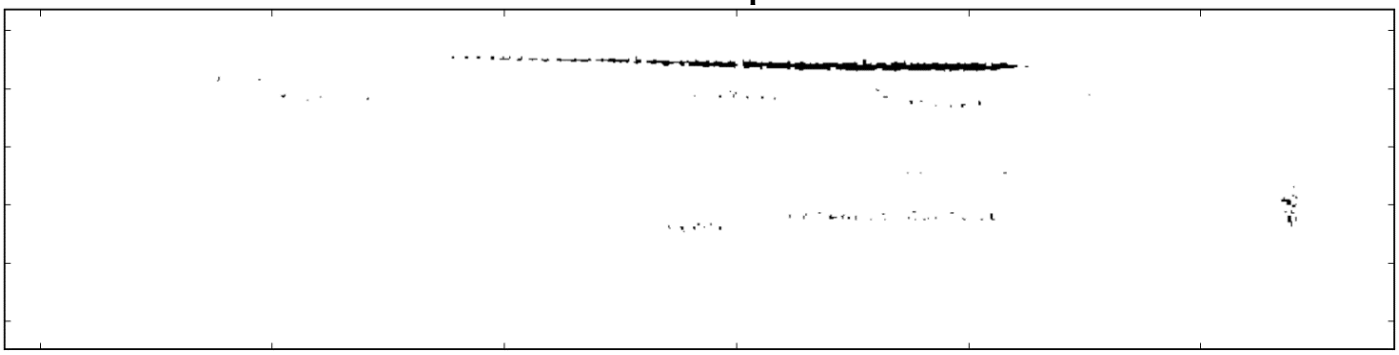
Spectrogramme original



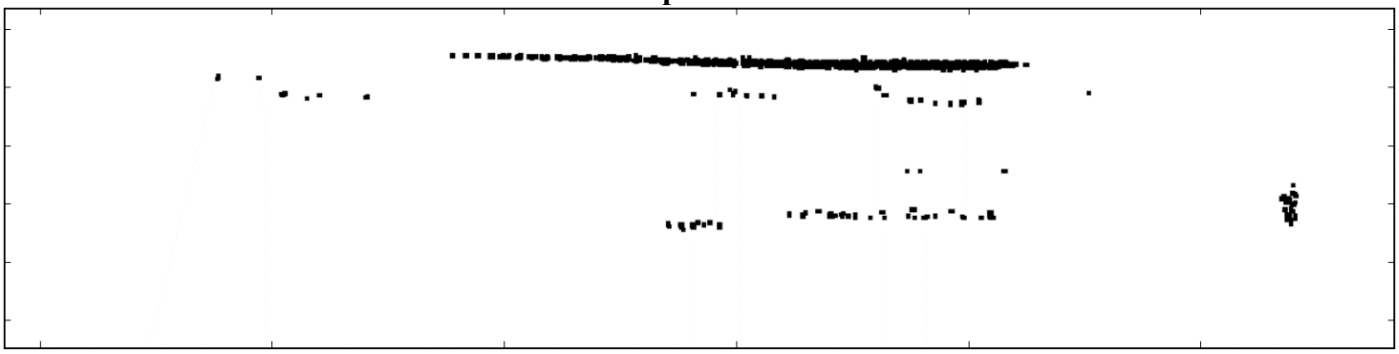
Pixels sélectionnés



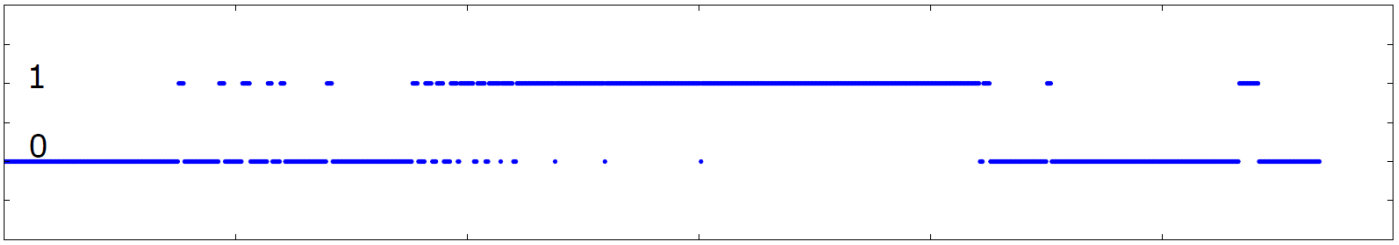
Pixels sélectionnés après érosion



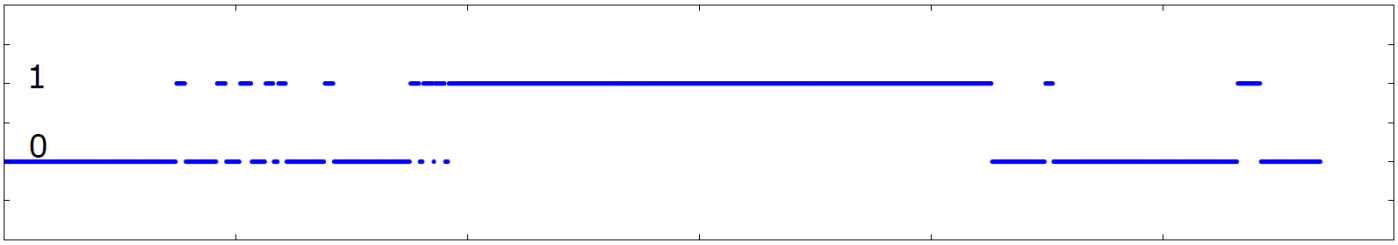
Pixels sélectionnés après érosion et dilatation



Colonnes sélectionnées



Colonnes sélectionnées après la première dilatation



Colonnes sélectionnées après la deuxième dilatation

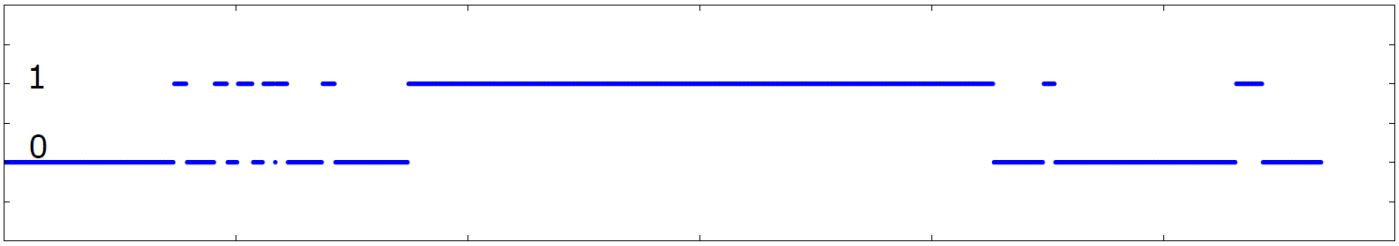


Figure 1 – Processus de suppression de bruit par érosion et dilatation

2. **Division des audios en chunks de 2 secondes:**

La division des audios en chunks de 2 secondes est une étape pratique pour gérer et traiter des enregistrements audios. Voici comment cette division est effectuée dans cet algorithme :

Étape 1 : Chargement de l'audio

L'audio d'origine est chargé à l'aide de la fonction "librosa.load". Cela permet d'obtenir le signal audio brut et les informations sur le taux d'échantillonnage.

Étape 2 : Calcul de la taille des chunks

La durée souhaitée pour chaque chunk est spécifiée en millisecondes, et la taille correspondante en échantillons est calculée en fonction du taux d'échantillonnage. Cela garantit que chaque chunk aura une durée cohérente.

Étape 3 : Suppression du bruit

Avant la division en chunks, le bruit est supprimé de l'audio en utilisant l'algorithme de suppression de bruit décrit précédemment. Cela permet d'améliorer la qualité du signal et de réduire les interférences indésirables.

Étape 4 : Division en chunks

L'audio prétraité est divisé en chunks de taille égale à l'aide de la fonction "np.array_split". Cela garantit que chaque chunk a la même durée, sauf le dernier chunk qui peut être redimensionné pour correspondre à la taille spécifiée.

Étape 5 : Enregistrement des chunks

Chaque chunk est enregistré individuellement en tant que fichier audio distinct à l'aide de la fonction "sf.write". Le nom du fichier de sortie est basé sur le nom d'origine du fichier, suivi d'un indicateur de numéro de chunk et du nombre total de chunks.

Deuxième approche

Dans cette approche, nous nous concentrerons sur la fonction reduce_noise de la bibliothèque noisereduce, qui est une technique couramment utilisée pour supprimer le bruit des enregistrements audio.

Principe de fonctionnement de reduce_noise:

La fonction reduce_noise utilise une technique appelée "réduction du bruit par seuil" (threshold-based noise reduction). Voici comment cette technique fonctionne :

Étape 1 : Calcul de la transformée de Fourier du signal audio

La fonction reduce_noise commence par calculer la transformée de Fourier du signal audio pour obtenir son spectrogramme. La transformée de Fourier permet de représenter le signal dans le domaine fréquentiel, en identifiant les différentes composantes de fréquence présentes dans l'audio.

Étape 2 : Identification du bruit

À partir du spectrogramme, reduce_noise identifie les parties du signal qui sont considérées comme du bruit. Il utilise généralement une méthode basée sur l'estimation de la puissance du bruit dans les parties silencieuses ou non vocales de l'audio.

Étape 3 : Estimation du bruit

Une fois le bruit identifié, reduce_noise estime sa puissance spectrale. Cette estimation est effectuée en calculant la médiane ou la moyenne des valeurs de puissance du bruit dans le spectrogramme. L'estimation de la puissance du bruit permet de déterminer un seuil au-dessus duquel les composantes fréquentielles sont considérées comme du signal.

Étape 4 : Application du seuil

À l'aide du seuil de puissance du bruit, reduce_noise compare chaque élément du spectrogramme avec ce seuil. Les composantes fréquentielles dont la puissance est inférieure au seuil sont considérées comme du bruit et sont réduites ou supprimées du spectrogramme.

Étape 5 : Reconstruction du signal audio

Après avoir appliqué le seuil de réduction du bruit, reduce_noise effectue la transformation inverse pour reconstruire le signal audio à partir du spectrogramme modifié. Cela permet d'obtenir un signal audio débarrassé du bruit indésirable.

Avantages de la fonction reduce_noise :

La fonction reduce_noise offre plusieurs avantages pour la suppression de bruit dans les enregistrements audio :

- Robustesse : Elle est capable de réduire différents types de bruit, qu'il s'agisse de bruit de fond constant, de bruit impulsionnel ou de bruit ambiant.
- Adaptabilité : La fonction reduce_noise peut être adaptée à différents environnements sonores en ajustant les paramètres, tels que le seuil de réduction du bruit.
- Préservation du signal : Elle tente de minimiser la distorsion du signal audio tout en supprimant le bruit, afin de préserver les informations importantes du signal d'origine.

Troisième approche

Dans cette approche, on décide de faire au minimum le nombre de pré-traitements possibles. Ainsi, on se limite à une extraction des caractéristiques spectro-temporelles à partir des enregistrements audios bruts pour les utiliser comme représentation sonore. Les caractéristiques utilisées sont les *log-mel band energies*, extraites à partir de courtes trames. Il a été démontré que ces caractéristiques fonctionnent bien dans diverses tâches de marquage audio et de détection d'événements sonores [2].

Pour cela, nous obtenons d'abord le spectre de magnitude des signaux audio en utilisant la transformée de Fourier à court terme (STFT) sur des trames audios de 40 ms avec un chevauchement de 50 % et une fenêtre de Hamming. La durée de chaque fichier audio dans le jeu de données du défi étant de 10 secondes, chaque représentation comporte 500 trames.

Ensuite, nous extrayons 40 caractéristiques d'énergie de bande mélodique logarithmiques à partir du spectre de magnitude. Le choix de se limite à 40 vient du principe que les sons d'oiseaux sont souvent présents dans une partie relativement restreinte de la plage de fréquences (principalement autour de 2-8 kHz). Cependant, l'extraction de caractéristiques sur l'ensemble de la plage de fréquences (de 0 Hz à la fréquence de Nyquist de 22050 Hz) peut donner de meilleurs résultats, mais nous avons préféré nous limiter à 40.

Cette approche a été choisie en raison de sa performance dans d'autres tâches de traitement audio.

Exemples de deux spectrogrammes extraits des données Freefield utilisant ce pré-traitement :

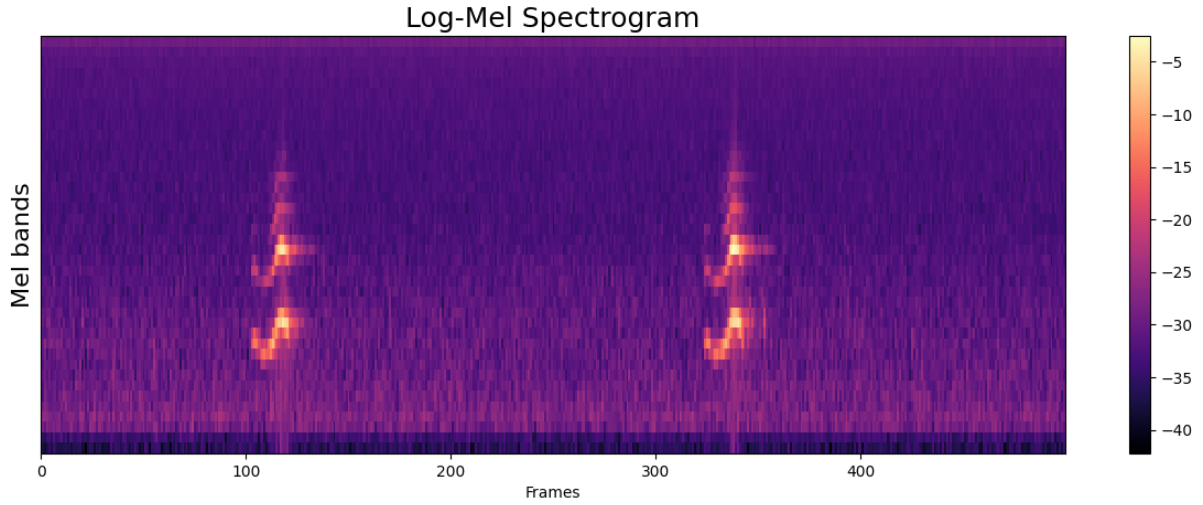


Figure 2 – Audio contenant des sons d'oiseaux

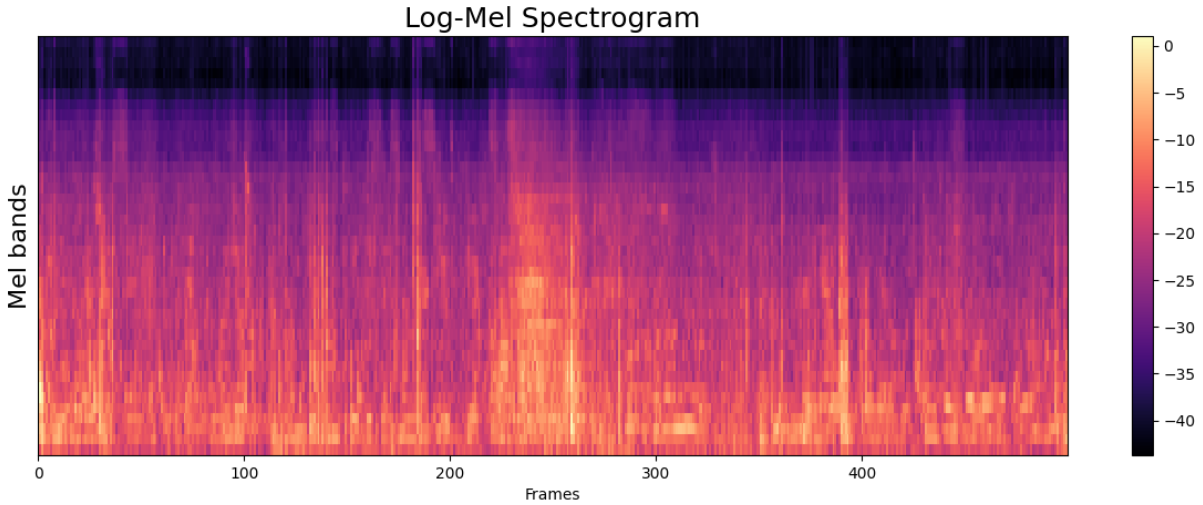


Figure 3 – Audio ne contenant pas de sons d'oiseaux

Description des modèles d'apprentissage

CNN (Réseau de neurones convolutif)

1. Architecture du CNN utilisé

a. Couche de convolution 1 :
- Entrée : 1 canal (pour les données audio)
- Sortie : 32 canaux
- Taille du noyau : (5, 7)
- Stride : (1, 1)
- Remplissage : (2, 3)

b. Couche de pooling :
- Taille du noyau : (2, 2)
- Stride : (2, 2)
- Pas de remplissage

c. Couche de convolution 2 :
- Entrée : 32 canaux (résultat de la couche précédente)
- Sortie : 64 canaux
- Taille du noyau : (5, 7)
- Stride : (1, 1)
- Remplissage : (2, 3)

d. Couche entièrement connectée 1 (FC1) :
- Entrée : 64 * 32 * 43 (dimension calculée à partir de la sortie de la deuxième couche de convolution)
- Sortie : 128

e. Fonction d'activation ReLU

f. Couche entièrement connectée 2 (FC2) :
- Entrée : 128
- Sortie : nombre de classes (défini lors de l'initialisation, par défaut 10)

La fonction forward définit le flux de données à travers le réseau. Elle applique les opérations dans l'ordre suivant :

I. La première couche de convolution est appliquée à l'entrée, suivie d'une fonction d'activation ReLU et d'une opération de pooling.
II. La deuxième couche de convolution est appliquée à la sortie de la première couche, suivie de la même séquence de ReLU et de pooling.
III. La sortie de la deuxième couche de convolution est ensuite remodelée (flatten) en un vecteur unidimensionnel.
IV. Ce vecteur est alimenté dans la couche entièrement connectée FC1, suivie d'une fonction d'activation ReLU.
V. Enfin, la sortie de FC1 est alimentée dans la couche entièrement connectée FC2 pour obtenir les probabilités de classe.

2. Les avantages pour la classification audio

Les avantages de ce modèle pour la classification audio :

a. Représentation spatio-temporelle : Ce modèle utilise des couches de convolution qui sont capables de capturer à la fois les caractéristiques spatiales et temporelles des données audios. Les convolutions permettent

de détecter des motifs locaux dans les spectrogrammes audio, ce qui est important pour la classification des signaux sonores.

b. Hiérarchie des caractéristiques : En utilisant plusieurs couches de convolution suivies de couches de pooling, le modèle est capable d'apprendre des caractéristiques hiérarchiques à différents niveaux d'abstraction. Les premières couches de convolution peuvent détecter des motifs simples tels que des bords ou des contours, tandis que les couches supérieures peuvent apprendre des caractéristiques plus complexes et spécifiques à la tâche de classification audio.

c. Invariance aux translations : Les couches de pooling utilisées dans ce modèle permettent d'obtenir une certaine invariance aux translations. Cela signifie que le modèle est capable de reconnaître les mêmes motifs peu importe leur position exacte dans le spectrogramme audio, ce qui est souhaitable pour la classification.

d. Capacité à gérer des données de grande taille : Les couches de pooling permettent de réduire la taille des caractéristiques extraites à chaque étape, ce qui réduit la complexité computationnelle et permet de traiter des données audios de grande taille de manière plus efficace.

e. Apprentissage de représentations discriminantes : Les couches entièrement connectées du modèle permettent d'apprendre des représentations discriminantes des données audios. Ces couches finales sont responsables de la classification en attribuant des poids aux caractéristiques extraites précédemment, permettant ainsi de distinguer différentes classes d'audio.

RNN (Réseau de neurones récurrent)

1. Architecture du RNN utilisé

a. Couche LSTM (self.lstm) :
- Taille d'entrée : input_size (définie lors de l'initialisation)
- Taille cachée : hidden_size (définie lors de l'initialisation, par défaut 64)
- Nombre de couches : num_layers (défini lors de l'initialisation, par défaut 2)
- Dropout : dropout (défini lors de l'initialisation, par défaut 0.3)
- Batch_first : True (utilisation du batch en première dimension)

b. Couche entièrement connectée (self.fc) :
- Taille d'entrée : hidden_size (résultat de la dernière couche LSTM)
- Taille de sortie : num_classes (défini lors de l'initialisation, par défaut 2)

La fonction forward définit le flux de données à travers le réseau. Les étapes effectuées :

I. La dimension de taille 1 est supprimée de l'entrée à l'aide de la fonction squeeze(1). Cela est fait pour enlever une dimension inutile qui est souvent ajoutée lors de l'encodage des données audio.
II. L'entrée est alimentée dans la couche LSTM, qui traite les séquences d'entrée. La LSTM est conçue pour capturer les relations temporelles à long terme dans les données séquentielles.
III. La sortie de la LSTM est passée à la couche entièrement connectée (FC), où la dernière sortie de la séquence est extraite (x[:, -1, :]) et alimentée dans la couche FC pour obtenir les probabilités de classe.

2. Les avantages pour la classification audio

Les avantages de ce modèle pour la classification audio :

a. Gestion des séquences temporelles : La nature récurrente du modèle LSTM lui permet de prendre en compte les informations temporelles des données audios. Les LSTMs sont capables de mémoriser des informations à long terme, ce qui est particulièrement important pour la classification de séquences audio où des motifs et des relations temporelles complexes peuvent exister.

b. Capture des dépendances à long terme : Contrairement aux réseaux de neurones classiques, les LSTMs sont conçus pour gérer les dépendances à long terme entre les éléments d'une séquence. Cela signifie qu'ils

peuvent prendre en compte les contextes plus larges et les relations complexes dans les signaux audio, ce qui est crucial pour une classification précise.

c. Traitement de séquences de longueur variable : Les LSTMs sont adaptés à la classification audio car ils peuvent gérer des séquences de longueur variable. Cela permet de traiter des enregistrements audios de différentes durées sans nécessiter un prétraitement complexe.

d. Adaptabilité aux différentes fréquences d'échantillonnage : Les LSTMs peuvent être utilisés avec succès pour traiter des données audios à différentes fréquences d'échantillonnage. Ils peuvent s'adapter aux caractéristiques spécifiques des données audio, qu'elles soient échantillonnées à des taux élevés ou bas.

SVM (Support Vector Machine)

1. **Description de l'algorithme SVM**

Étape 1 : Préparation des données

Les données d'entraînement doivent être dans un format approprié pour l'algorithme SVM. Cela signifie que les échantillons doivent être représentés par des vecteurs de caractéristiques numériques.

Étape 2 : Sélection des vecteurs de support

L'algorithme SVM cherche à trouver un hyperplan dans un espace de grande dimension qui sépare au mieux les échantillons de différentes classes. Pour ce faire, il identifie un sous-ensemble d'échantillons appelés "vecteurs de support" qui se trouvent à proximité de la frontière de décision.

Étape 3 : Représentation des données

Les échantillons d'entraînement sont projetés dans un espace de grande dimension en utilisant une fonction de noyau (*kernel function*). Cette projection permet de traiter des problèmes non linéaires en transformant les données dans un espace où elles peuvent être linéairement séparables.

Étape 4 : Optimisation de la marge

L'objectif principal de SVM est de maximiser la marge, c'est-à-dire la distance entre l'hyperplan de décision et les vecteurs de support les plus proches de chaque classe. Cela permet de favoriser une meilleure généralisation du modèle en minimisant le risque de sur-ajustement (*overfitting*).

Étape 5 : Résolution du problème d'optimisation

L'algorithme SVM résout un problème d'optimisation convexe qui consiste à trouver les poids (coefficients) du modèle qui minimisent une fonction objective, tout en satisfaisant les contraintes de marge. Cette optimisation est généralement réalisée à l'aide de techniques d'optimisation numérique, telles que la descente de gradient ou les méthodes de programmation quadratique.

2. **Les avantages pour la classification audio**

Les avantages pour la classification audio :

a. Gestion des espaces de grande dimension : SVM peut gérer efficacement les espaces de grande dimension. Cela en fait un choix adapté pour les problèmes de classification avec des caractéristiques de haute dimensionnalité, tels que les données audios représentées par des spectrogrammes.

b. Contrôle du sur-ajustement : En maximisant la marge entre les classes, SVM favorise une meilleure généralisation et réduit le risque de sur-ajustement (*overfitting*). Cela permet d'obtenir de bonnes performances de classification sur de nouvelles données.

c. Flexibilité du choix de la fonction de noyau : SVM offre la possibilité de choisir différentes fonctions de noyau, telles que linéaire, polynomial, gaussien (RBF) ou sigmoïde. Cela permet d'adapter le modèle aux caractéristiques spécifiques des données audio.

d. Efficacité en présence de données déséquilibrées : SVM peut être utilisé pour la classification audio lorsque les classes sont déséquilibrées. En ajustant les poids des classes ou en utilisant des techniques de sur-échantillonnage/sous-échantillonnage, SVM peut gérer les déséquilibres de classe et fournir de bons résultats pour les classes minoritaires.

CRNN (Réseau de neurones convolutif-récurrent)

1. Architecture du CRNN utilisé

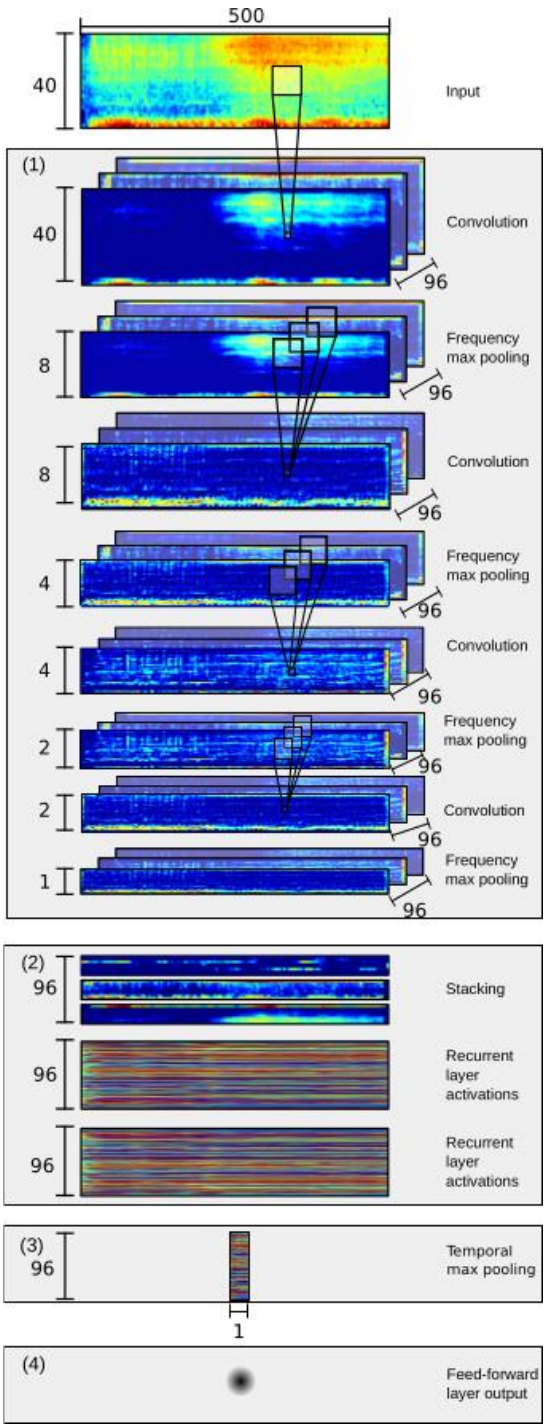


Figure 4 – Architecture du CRNN [2]

2. **Les avantages pour la classification audio**

Ce modèle combine les avantages d'un réseau de neurones convolutif et d'un réseau de neurones récurrent. De plus, il est particulièrement adapté à notre troisième approche de prétraitement des données. Cependant, de par sa publication en 2016, il peut être considéré comme relativement ancien. Par conséquent, nous nous sommes engagés à explorer et à améliorer ce modèle pour répondre aux besoins actuels et aux avancées dans le domaine de la classification audio.

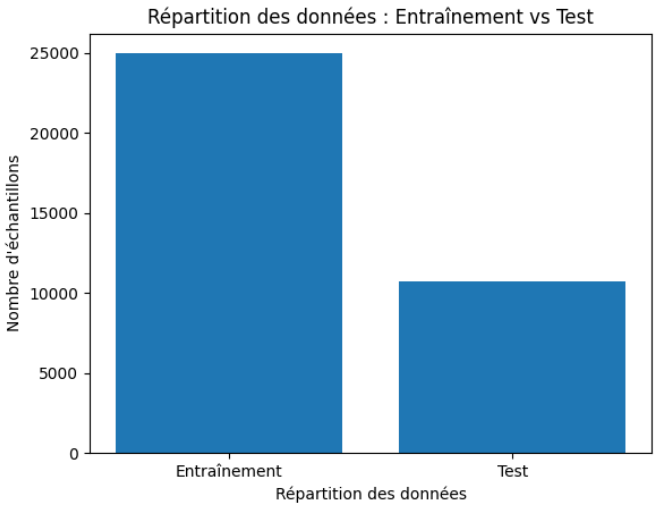
Méthodologie expérimentale

Division des données en ensembles d'apprentissage et de test

Pour préparer notre modèle de classification, nous avons divisé notre ensemble de données en deux parties : un ensemble d'apprentissage et un ensemble de test. La division des données a été effectuée en respectant une proportion de 70% pour l'apprentissage et 30% pour le test.

Il est important de noter que la division des données a été réalisée de manière aléatoire afin d'éviter tout biais potentiel. Cela garantit que les deux ensembles sont représentatifs de la distribution des données d'origine.

Nous avons utilisé cette division des données en ensembles d'apprentissage et de test pour évaluer les performances de notre modèle de classification audio et mesurer sa capacité à généraliser sur des données inconnues.



Paramètres spécifiques pour chaque modèle et approche de prétraitement

Pour explorer différentes configurations et optimiser les performances de nos modèles de classification audio, nous avons considéré des paramètres spécifiques pour chaque modèle et approche de prétraitement. Nous avons examiné plusieurs variations pour chaque type de modèle, en ajustant les architectures et les hyperparamètres pour obtenir de meilleures performances.

Pour notre modèle CNN, nous avons expérimenté trois configurations différentes en plus de celle présentée. Tout d'abord, nous avons ajouté une couche dense supplémentaire à la fin du réseau afin d'introduire une complexité et une capacité de représentation plus élevées. Ensuite, nous avons remplacé la couche de Max Pooling par une couche de Average Pooling pour explorer l'effet d'une agrégation différente des informations spatiales. Enfin, nous avons utilisé la fonction Softmax au lieu de Max Pooling pour obtenir des probabilités de classement pour chaque classe. Ces variations nous ont permis de comparer les performances et de sélectionner la meilleure configuration pour notre tâche de classification audio.

Dans le cas de notre modèle RNN, nous avons également expérimenté différentes architectures. Nous avons ajouté une couche de sortie supplémentaire avec une activation ReLU pour introduire une non-linéarité supplémentaire dans le modèle. De plus, nous avons remplacé le LSTM par un GRU (Gated Recurrent Unit) pour évaluer les performances et les avantages de cette variante de cellule récurrente. Enfin, nous avons utilisé une architecture de Bidirectional LSTM pour capturer les informations contextuelles dans les deux directions temporelles. Ces variations nous ont permis d'évaluer la contribution de chaque modification et de choisir la configuration optimale pour la classification audio.

Concernant notre modèle SVM, nous avons utilisé une approche de recherche de grille de paramètres pour déterminer les meilleurs hyperparamètres. Nous avons considéré une combinaison de valeurs pour les paramètres C, kernel, degree et gamma. Nous avons exploré différentes valeurs pour C pour réguler la complexité du modèle, ainsi que différents types de noyaux (linear et sigmoid) et de degrés pour évaluer leur impact sur la séparabilité des données. Nous avons également utilisé 'gamma' avec la valeur 'scale' pour ajuster l'influence des échantillons sur la décision frontalière. Cette recherche exhaustive des paramètres nous a permis de trouver la configuration optimale pour notre modèle SVM dans le contexte de la classification audio.

Concernant le modèle CRNN, nous avons dans un premier temps cherché à savoir quelle couche récurrente est la plus apte pour notre tâche, entre une couche RNN classique, une couche LSTM ou une couche GRU. Deuxièmement, nous nous sommes attelés à la représentation des données en sorties de la couche récurrente. Par défaut, le modèle propose une couche de pooling temporelle située à la sortie de la couche récurrente. Elle permet d'agréger les informations temporelles et produire une représentation globale du signal audio. Elle est essentielle pour réduire la dimensionnalité de la séquence de sortie du GRU, tout en préservant les caractéristiques les plus discriminantes. Ainsi, cette couche permet de capturer les aspects importants du signal audio sur des segments de temps plus larges, facilitant ainsi la prise de décision finale de classification. Partant de ce postulat, nous avons exploré deux approches :

a. La première approche consiste à utiliser une simple couche dense qui prend la probabilité maximale sur toute la séquence en tant que sortie, afin de ne considérer que la caractéristique temporelle la plus dominante. Cependant, cette approche peut potentiellement perdre des informations temporelles plus fines présentes dans la séquence.
b. La deuxième approche se base sur le principe de l'attention présenté en 2017 [11]. Ainsi, on utilise une couche d'attention pour calculer une pondération pour chaque instant de temps de la séquence de sortie du GRU. Cette pondération est ensuite utilisée pour agréger les caractéristiques temporelles en fonction de leur importance relative. L'attention permet de mettre l'accent sur les parties temporelles les plus pertinentes et de donner plus de poids aux segments de temps contenant des informations discriminantes pour la classification.

En conclusion, l'exploration de différents paramètres spécifiques pour chaque modèle et approche de prétraitement nous a permis de sélectionner les configurations les plus performantes et adaptées à notre tâche de classification audio.

Résultats et analyse

Présentation des résultats obtenus pour chaque modèle

Première approche

Modèle	Pooling	Freefield1010		Warblrb10K		BirdVox	
		AUC	F1	AUC	F1	AUC	F1
CNN	-	.549	.585	**.718**	.605	**.736**	.710
CNN	Dense	.596	.533	.692	.575	.730	**.752**
CNN	Average Pooling	.591	.545	.586	**.704**	.674	.736
RNN	-	.525	.612	.654	.664	.725	.717
RNN	Dense	.564	**.645**	.625	.586	.635	.615
RNN	GRU	**.610**	**.645**	.694	.615	.705	.735

Tableau 1 - Performances pour la première approche

Deuxième approche

Modèle	Kernel	Freefield1010		Warblrb10K		BirdVox	
		AUC	F1	AUC	F1	AUC	F1
CNN	-	**.599**	**.678**	.459	.395	**.691**	**.760**
RNN	-	.584	.643	**.554**	.495	.604	.575
SVM	Linéaire de degré 2	.446	.467	.415	.399	.552	.526
SVM	Linéaire de degré 3	.511	.455	.435	.421	.564	.527
SVM	Sigmoïde de degré 2	.557	.524	.535	**.504**	.495	.485
SVM	Sigmoïde de degré 3	.533	.543	.553	.446	.554	.590

Tableau 2 - Performances pour la première approche

Troisième approche

Modèle	Pooling	Freefield1010		Warblrb10K		BirdVox	
		AUC	F1	AUC	F1	AUC	F1
CNN	Max probability	.847	.730	.849	.926	.873	.860
CNN	Temporal	.842	.706	.853	**.937**	.888	**.885**
CNN	Soft Attention	.847	.764	.851	.904	.852	.837
$CRNN_{RNN}$	Temporal	.841	.708	.789	.921	.877	.868
$CRNN_{LSTM}$	Temporal	.857	.783	.868	.906	.882	.873
$CRNN_{LSTM}$	Soft Attention	.849	.783	.858	.908	.870	.857
$CRNN_{Transformers}$	Temporal	.820	.755	.844	884	.836	.811
$CRNN_{GRU}$	Max probability	**.861**	.778	.860	.908	.882	.872
$CRNN_{GRU}$	Temporal	.840	.719	.818	.923	.890	.883
$CRNN_{GRU}$	Soft Attention	.859	**.784**	**.870**	.916	**.891**	.883

Tableau 3 - Performances du CRNN

Comparaison des performances entre les modèles et les approches de prétraitement

Nous avons testé nos modèles sur trois jeux de données : Freefield1010, Warblrb10K et BirdVox.

Dans la première approche de prétraitement, les modèles CNN ont généralement obtenu de meilleurs résultats, en particulier le CNN Dense pour Freefield1010 **(AUC: 0.596, F1: 0.533)** et **BirdVox (AUC: 0.730, F1: 0.752)**, et le CNN simple pour Warblrb10K **(AUC: 0.718, F1: 0.605)**. En comparaison, les performances des modèles RNN étaient légèrement inférieures, bien que le RNN GRU ait obtenu des résultats comparables sur le jeu de données BirdVox **(AUC: 0.705, F1: 0.735)**.

Avec la deuxième approche de prétraitement, les modèles CNN et RNN ont de nouveau montré de bons résultats, en particulier le CNN sur Freefield1010 **(AUC: 0.599, F1: 0.678)** et BirdVox **(AUC: 0.691, F1: 0.760)**. Les modèles SVM, malgré leur flexibilité, ont eu des performances généralement inférieures, avec les meilleurs résultats obtenus par le SVM Sigmoïde de degré 3 sur BirdVox **(AUC: 0.554, F1: 0.590)**.

Dans la troisième approche de prétraitement, nous avons introduit des modèles CRNN, qui combinent les caractéristiques des réseaux de neurones convolutionnels (CNN) et récurrents (RNN). Ces modèles ont obtenu les meilleures performances, en particulier le CRNNGRU Max probability sur Freefield1010 **(AUC: 0.861, F1: 0.778)**, le CNN Temporal sur Warblrb10K **(AUC: 0.853, F1: 0.937)** et le CRNNGRU Soft Attention sur BirdVox **(AUC: 0.891, F1: 0.883)**.

En conclusion, la comparaison des performances suggère que la troisième approche de prétraitement, associée aux modèles CRNN, est la plus efficace pour la classification d'audio de chants d'oiseaux sur les trois jeux de données testés.

Limitations et pistes d'amélioration

Limitations du projet et des approches utilisées

Il est important de noter certaines limitations qui peuvent affecter notre projet et les approches que nous avons utilisées pour la classification audio.

Tout d'abord, l'une des limitations concerne la taille et la diversité de notre ensemble de données. Bien que nous ayons fait de notre mieux pour rassembler un ensemble de données représentatif, il est possible qu'il ne couvre pas toutes les variations et les nuances présentes dans les données audio réelles. Une plus grande diversité de données pourrait améliorer la capacité de généralisation de nos modèles.

De plus, les approches de prétraitement que nous avons utilisées, telles que les spectrogrammes ou les coefficients MFCC, peuvent présenter certaines limitations. Ces techniques sont largement utilisées et efficaces pour de nombreuses tâches de classification audio, mais elles peuvent ne pas capturer tous les aspects importants des signaux audios. D'autres méthodes de prétraitement ou des combinaisons d'approches pourraient être explorées pour améliorer encore les performances.

Une autre limitation concerne les hyperparamètres choisis pour nos modèles. Bien que nous ayons effectué des recherches et des expérimentations pour trouver les meilleures configurations, il est possible que d'autres combinaisons de paramètres puissent améliorer davantage les performances. Une recherche plus exhaustive des hyperparamètres pourrait être envisagée pour obtenir des résultats encore meilleurs.

Les caractéristiques des signaux audio, les conditions d'enregistrement, le bruit de fond et d'autres facteurs peuvent influencer les performances des modèles. Il est important de prendre en compte ces spécificités lors de l'application des modèles dans des contextes réels.

Perspectives de recherche et d'exploration

Ce projet de classification audio offre de nombreuses perspectives de recherche et d'exploration pour améliorer la compréhension et les performances dans ce domaine en constante évolution. Voici quelques pistes prometteuses qui pourraient être explorées :

1. Apprentissage semi-supervisé : L'exploration de techniques d'apprentissage semi-supervisé pourrait être bénéfique dans le contexte de la classification audio. L'utilisation de données étiquetées et non étiquetées pour entraîner les modèles pourrait permettre d'améliorer les performances en exploitant efficacement les informations disponibles.
2. Classification audio en temps réel : Le développement de techniques de classification audio en temps réel est une perspective importante pour de nombreuses applications, telles que la détection d'événements sonores ou la surveillance acoustique. L'exploration de modèles et de méthodes adaptés à la classification en temps réel, y compris l'utilisation de fenêtres glissantes et de techniques d'apprentissage en ligne, pourrait permettre des applications plus dynamiques et réactives.
3. Apprentissage multimodal : L'intégration de l'audio avec d'autres modalités, comme la vidéo ou les capteurs environnementaux, peut enrichir la compréhension des scènes audio et améliorer les performances de classification. L'exploration de modèles multimodaux et de techniques de fusion de données pourrait permettre une meilleure exploitation des informations complémentaires.
4. Transfert de connaissances : Le transfert de connaissances à partir de tâches ou de domaines connexes peut être une approche prometteuse pour la classification audio. L'utilisation de modèles pré-entraînés sur de grandes bases de données audio ou sur des tâches similaires peut accélérer l'apprentissage et améliorer les performances en exploitant des connaissances préalables.

Ces perspectives offrent un aperçu des domaines de recherche et d'exploration passionnants qui peuvent contribuer à l'avancement de la classification audio. En poursuivant ces investigations, nous pourrions atteindre une meilleure compréhension des signaux audio et développer des systèmes de classification plus performants pour répondre aux défis et aux exigences de diverses applications audio.

Conclusion

En conclusion, ce projet de classification audio nous a permis d'explorer différentes approches et modèles pour la classification précise des signaux audios. En analysant les résultats et en évaluant les performances, nous avons pu tirer plusieurs conclusions clés :

Tout d'abord, les modèles CNN (Convolutional Neural Networks) ont démontré leur capacité à extraire des caractéristiques discriminantes à partir de représentations spectrogrammes des signaux audios. En utilisant une architecture de réseau CNN adaptée à notre tâche, nous avons obtenu des résultats prometteurs, montrant que ces modèles sont efficaces pour la classification audio.

De plus, les modèles RNN (Recurrent Neural Networks), tels que les LSTM (Long Short-Term Memory) et les GRU (Gated Recurrent Unit), ont montré leur capacité à capturer les dépendances temporelles des signaux audio. En exploitant la mémoire à court terme des RNN, nous avons obtenu de bonnes performances de classification et une meilleure compréhension des séquences audio.

En outre, l'algorithme SVM (Support Vector Machine) s'est révélé efficace pour la classification audio, offrant une approche alternative robuste et performante. En explorant différentes combinaisons de paramètres et en utilisant une recherche de grille, nous avons pu trouver la configuration optimale pour notre modèle SVM, fournissant ainsi de bons résultats de classification.

Enfin, les CRNN (Convolutional Recurrent Neural Networks) ont démontré leur potentiel pour la classification audio, exploitant la force des modèles CNN et RNN. De plus, l'attention est une approche

extrêmement prometteuse dans le domaine de la classification audio et représente clairement une piste à explorer. En permettant au modèle de se concentrer sur des parties spécifiques d'un signal audio, l'attention offre une capacité d'apprentissage plus fine et une meilleure compréhension des caractéristiques discriminantes. Cela permet de mettre l'accent sur les parties les plus pertinentes des séquences audios, en négligeant les informations moins importantes ou perturbatrices. Grâce à l'attention, les modèles peuvent prendre en compte les variations temporelles et fréquentielles des signaux audio, ce qui permet une classification plus précise et une meilleure généralisation aux données de test.

En ce qui concerne les approches de prétraitement, l'utilisation de spectrogrammes et de coefficients MFCC a été largement bénéfique pour la représentation des signaux audio. Ces techniques ont permis de capturer des caractéristiques pertinentes et discriminantes pour la classification.

De plus, nous avons identifié certaines limitations du projet et des approches utilisées, telles que la taille limitée de l'ensemble de données, les choix spécifiques des hyperparamètres et les techniques de prétraitement traditionnelles. Ces limitations offrent des opportunités pour des améliorations futures et une exploration plus approfondie.

En somme, ce projet nous a permis de mieux comprendre les différentes approches et modèles pour la classification audio. Nous avons pu observer les avantages et les performances de chaque méthode, ainsi que les limitations et les pistes d'amélioration. Ces conclusions nous orientent vers de futures recherches et explorations pour améliorer la classification audio et répondre aux besoins spécifiques de diverses applications.

Bibliographie

[1] Sharath Adavanne, Konstantinos Drossos, Emre Çakir, and Tuomas Virtanen. 2017. Stacked convolutional and recurrent neural networks for bird audio detection. In *2017 25th European Signal Processing Conference (EUSIPCO)*, 1729–1733. DOI:https://doi.org/10.23919/EUSIPCO.2017.8081505

[2] Emre Cakir, Sharath Adavanne, Giambattista Parascandolo, Konstantinos Drossos, and Tuomas Virtanen. 2017. Convolutional recurrent neural networks for bird audio detection. In *2017 25th European Signal Processing Conference (EUSIPCO)*, IEEE, Kos, Greece, 1744–1748. DOI:https://doi.org/10.23919/EUSIPCO.2017.8081508

[3] Thomas Grill and Jan Schluter. 2017. Two convolutional neural networks for bird detection in audio signals. In *2017 25th European Signal Processing Conference (EUSIPCO)*, IEEE, Kos, Greece, 1764–1768. DOI:https://doi.org/10.23919/EUSIPCO.2017.8081512

[4] Chih-Yuan Koh, Jaw-Yuan Chang, Chiang-Lin Tai, Da-Yo Huang, Han-Hsing Hsieh, and Yi-Wen Liu. 2019. Bird Sound Classification using Convolutional Neural Networks. (2019).

[5] Qiuqiang Kong, Yong Xu, and Mark D. Plumbley. 2017. Joint detection and classification convolutional neural network on weakly labelled bird audio detection. In *2017 25th European Signal Processing Conference (EUSIPCO)*, 1749–1753. DOI:https://doi.org/10.23919/EUSIPCO.2017.8081509

[6] Mario Lasseck. 2018. Audio-based Bird Species Identification with Deep Convolutional Neural Networks. (2018).

[7] Yanxiong Li, Wenchang Cao, Konstantinos Drossos, and Tuomas Virtanen. 2022. Domestic Activity Clustering from Audio via Depthwise Separable Convolutional Autoencoder Network.

[8] Elias Sprengel, Martin Jaggi, Yannic Kilcher, and Thomas Hofmann. 2016. Audio Based Bird Species Identification using Deep Learning Techniques. (2016).

[9] Anshul Thakur, R. Jyothi, Padmanabhan Rajan, and A.D. Dileep. 2017. Rapid bird activity detection using probabilistic sequence kernels. In *2017 25th European Signal Processing Conference (EUSIPCO)*, IEEE, Kos, Greece, 1754–1758. DOI:https://doi.org/10.23919/EUSIPCO.2017.8081510

[10] Bálint Pál Tóth and Bálint Czeba. 2016. Convolutional Neural Networks for Large-Scale Bird Song Classification in Noisy Environment. (2016).

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. DOI:https://doi.org/10.48550/arXiv.1706.03762

[12] Liwen You, Erika Pelaez Coyotl, Suren Gunturu, and Maarten Van Segbroeck. 2023. Transformer-Based Bioacoustic Sound Event Detection on Few-Shot Learning Tasks. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, Rhodes Island, Greece, 1–5. DOI:https://doi.org/10.1109/ICASSP49357.2023.10097081