PSIBIOM Technical Report LIS CNRS UTLN - 2024-07

Tristan Villepreux, Philémon Prevot, Lisa Ferre, Stéphane Chavin, Hervé Glotin

Technical Report LIS CNRS UTLN - 2024-07

Rapport d'avancement volet acoustique *online et offline*

Auteur.es :

Tristan VILLEPREUX, Philémon PRÉVOT, Lisa FERRE, Stéphane CHAVIN, Hervé GLOTIN

Université de Toulon, Aix Marseille Univ, CNRS, LIS, DYNI, Marseille, France
Centre Int. d'IA en Acoustique Naturelle, Toulon, France, https://cian.univ-tln.fr

Introduction

Ce rapport fait un état des lieux du projet PSIBIOM avant la fermeture annuelle du laboratoire d'août 2024. Il fait suite à une réunion avec les équipes PSIBIOM : Catherine et Guillaume de Terroiko, Christian et Etienne de SIConsult, Lisa, Philémon, Stéphane et Tristan du LIS. Le compte rendu de cette réunion est accessible au lien suivant :
https://docs.google.com/document/d/1tKq2HpsDcbMxYajHaK-o240WAuWu8sMKDDK9_JN0CCw/edit#heading=h.b0ed5qnow8m9

I. Indices éco-acoustiques

Les indices éco-acoustiques sont un moyen d'estimer la complexité d'un signal audio/chorus et peuvent être utilisés en tant que proxy de diversité de la faune jusqu'à un certain seuil.

Les indices que nous avons prévu d'utiliser pour le projet PSIBIOM sont:
ACI Acoustic Complexity Index
ADI Acoustic Diversity Index

A l'heure actuelle les indices ont été calculés sur les anciens enregistrements reçus avant 2024 et n'ont pas été encore calculés sur les derniers enregistrements. Les indices ne sont pas encore intégrés à la carte QHBv3 mais existent en langage python et nécessitent une conversion en langage C.

L'objectif étant de lancer ces deux indices sur le chorus général enregistré et selon le seuil obtenu pour chaque, déterminer la diversité ou présence potentielle d'un groupe pour le valider ensuite par une approche IA (Oiseaux) ou filtre de fréquences (Chiroptères). Ces indices éco-acoustiques serviront donc de proxi au suivi de la biodiversité locale.

Partie Chemin/Aide
Le code pour faire les indices éco-acoustiques se trouve ici: /nfs/NAS7/SABIOD/SITE/PSIBIOM/methode/INDICES avec les résultats sur les anciens enregistrements

II. Classification

A. Groupes Anoures, Oiseaux, Orthoptère et Hémiptère

Expérience 1. Création d'un classifier

Pierre Mahe a développé un classifieur inspiré du modèle ceta-cnn de Paul Best qui permet de différencier Anoures, Oiseaux, Orthoptère et Hémiptère. Les performances de son classifier sont les suivantes (résultats du test):
roc_auc: 0.994, mAP: 0.990
mAP_detail: ['1.000', '0.999', '0.996', '0.969'] (['Orthoptera', 'Birds', 'Anura', 'Hemiptera'])

Partie Chemin/Aide

DATASET:
/nfs/NAS7/SABIOD/SITE/PSIBIOM/methode/small_detec_multi/binary_annot_sampleDur4_multiclass.pkl

CODE TRAIN:

/nfs/NAS7/SABIOD/SITE/PSIBIOM/methode/small_detec_multi/train_stft_nfeat.py

POIDS

/nfs/NAS7/SABIOD/SITE/PSIBIOM/methode/small_detec_multi/runs/weights/dw_mel128_brown_32kHzhps256_512_k5_specBN_sch97_32k_0911.stdc

CODE TEST

/nfs/NAS7/SABIOD/SITE/PSIBIOM/methode/small_detec_multi/test_V2_PSIBIOM.py

Expérience 2. Adaptation du classifier pour l'embarqué

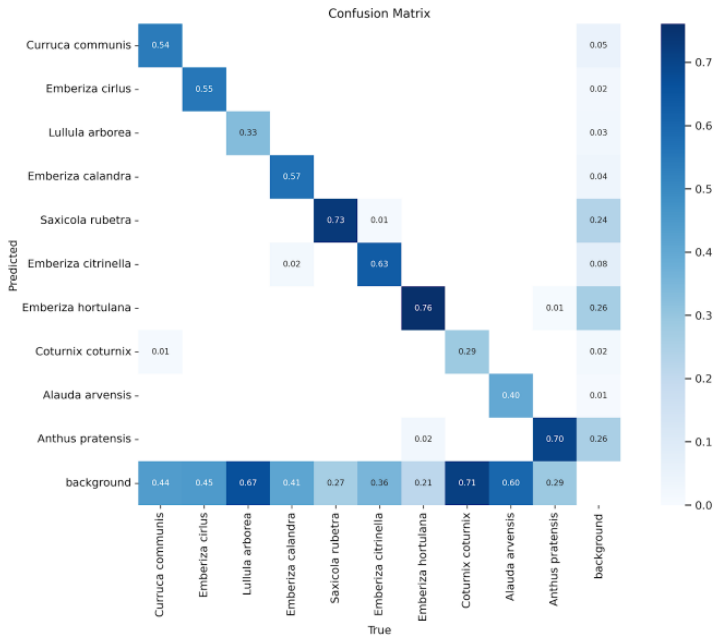
Afin de déployer ce classifier sur la carte QHB il faut le réadapter en N détecteurs pour éviter un grand changement du code C embarqué déjà à disposition et rester dans les temps. Ces N détecteurs ont été lancés en entraînements avant la fermeture du laboratoire LIS (19/07/2024). Il reste à récupérer leurs poids car le traitement et l'architecture est la même que celle des oiseaux.

Partie Chemin/Aide
/nfs/NAS7/SABIOD/SITE/PSIBIOM/méthode/small_detec_multi/runs/uniclass/ => résultats et poids lancé sur Bigpus screen PSIBIOM (session lisa.ferre)

B. Oiseaux

Expérience 1. Estimation des limites de classifications d'espèces d'oiseaux

Yolov5s un modèle lourd et offline/non embarqué a permis en premier lieu d'estimer les limites de la détection et classification d'espèces d'oiseaux par acoustique en IA. Les performances obtenues sont les suivantes:



espèces	label	nb_annotations_initial
Curruca communis	0	3823
Emberiza cirlus	1	2598
Lullula arborea	2	1777
Emberiza calandra	3	1413
Saxicola rubetra	4	1251
Emberiza citrinella	5	1185
Emberiza hortulana	6	839
Coturnix coturnix	7	812
Alauda arvensis	8	694
Anthus pratensis	9	561
Total		**14953**

Figure. *Matrice de confusion des prédictions Yolov5 faites sur le dataset oiseaux agricoles filtré contenant les 10 espèces d'intérêt*

Lors de cette première approche des problématiques liées aux annotations ont été relevées notamment l'absence des valeurs de fréquence minimum et maximum sur ⅔ des annotations reçues d'oiseaux. Ce manque a été justifié par l'annotation sur Birdnet et n'est désormais plus un problème sur le modèle embarqué car les annotations sont uniquement sur la dimension temporelle.

Expérience 2. Passage à un modèle conçu pour de l'embarqué

Le modèle utilisé pour la classification entre Anoures, Oiseaux, Orthoptère et Hémiptère a été entraîné pour classifier les 10 espèces d'oiseaux d'intérêt d'abord en multiclass puis en 10 détecteurs différents. Il y a une baisse de performance entre Yolov5 et ce modèle multiclass car beaucoup moins puissant mais aussi entre multiclass et 10 détecteurs (peut-être lié à la confusion entre espèces).

Résultats en multiclass:
mAP: 0.481, ROC AUC: 0.821
mAP_detail: ['0.492', '0.439', '0.512', '0.625', '0.200', '0.391', '0.161', '0.447', '0.294', '0.331']

Résultats avec 10 détecteurs:
mAP: , ROC AUC: 0,762
mAP_detail: [‘0.416‘, ’0.309’, ‘0.568’, ‘0.456’, ‘0.614’, ’0.261’, ‘0.430’, ‘0.265’, ‘0.465’, ’0.219’]

['Alauda_arvensis', 'Anthus_pratensis', 'Coturnix_coturnix', 'Curruca_communis', 'Emberiza_calandra', 'Emberiza_citrinella', 'Emberiza_cirlus', 'Emberiza_hortulana', 'Lullula_arborea', 'Saxicola_rubetra']

Les poids et architecture ont été exportés et intégrés dans la carte QHB test présente au laboratoire. Il reste à le tester sur la version de carte correspondante au futur manipulation terrain.

A. Chiroptères

Classification des différentes espèces trop complexes car signaux très semblables au sein d'un même genre en termes de fréquences et formes.
Donc décision de détecter les chiroptères et les classifier par genres :

Murins (= myotis), Pipistrelles, Noctules, Oreillards (= Plecotus) et Rhinolophes.

Expérience 1 - YOLOv5 sur annotations Yves Bas

YOLOv5 s'est montré inefficace à la détection de chiroptères. Une hypothèse a été que la différence de qualité des annotations était source d'erreur lors de l'apprentissage du modèle. Nous avons donc réalisé un train basé sur les annotations réalisées par Yves Bas uniquement. Le résultat n'a pas été concluant avec une matrice de confusion ne différenciant pas les espèces de chiroptères du bruit de fond. L'autre hypothèse est que l'expansion de temps ne permette pas de donner de résultats significatifs via les spectrogrammes.
La piste Yolo v5 est donc mise de côté pour la détection de chiroptères. De plus, le modèle étant particulièrement lourd, il ne sera pas le plus adapté à de la détection / classification embarquée.

Expérience 2 - Modèle IA embarqué basé sur des poids de détection de cachalot / Ceta_cnns Cachalot

Méthode :

La 1ere experience a été de lancer le modèle de détections de vocalises ceta_cnns pour détecter les chiroptères? Il a d'abord été utilisé avec les poids d'entrainement des cachalots. Puis dans un second temps avec les poids d'entrainement delphinidés. Cela à permis de vérifier le modèle le plus adapté et de valider l'architecture cachalots à améliorer.

Résultats :

141 fichiers ont été détectés comme contenant au moins une détection de chiro sur un total de 595 fichiers contenant chacun des vocalises de chiroptères. Modèle à approfondir mais peu concluant tel quel. Besoin de réentraîner les poids.

```
PSIBIOM/methode/Chiro/ceta-cnns/Ceta-cnn_exploit.py
prediction
0.000000    441
0.501094      1
0.502943      1
0.508159      1
0.510526      1
            ...
0.999999      1
1.000000      3
1.000000      1
1.000000      1
1.000000     13
Length: 141, dtype: int64
```

Modèle cachalot : Colonne 1 : taux de prédiction d'une présence de chiroptère, colonne 2 : nombres de fichiers concernés. Seuil de prise en compte à 0;5

En utilisant les poids d'entraînement des delphinidés, seulement 11 fichiers donnent lieu à une détection si l'on place un seuil de détection pour un taux de prédiction > 0,5. Le modèle delphinidé offre donc moins de potentiel que le modèle cachalot.

```
PSIBIOM/methode/Chiro/ceta-cnns/Ceta-cnn_exploit.py
prediction
0.000000    585
0.620069      1
0.689735      1
0.748426      1
0.752725      1
0.813230      1
0.861626      1
0.916715      1
0.949293      1
0.980186      1
0.990250      1
0.994446      1
dtype: int64
```

Modèle delphinidés : Colonne 1 : taux de prédiction d'une présence de chiroptère, colonne 2 : nombres de fichiers concernés. Seuil de prise en compte à 0;5

Expérience 3 - Modèle IA embarqué basé sur des poids personnalisés /Small_detec_multi

Méthode :

Réalisation d'un script de conversion csv to pkl convertisseur les noms d'espèces en genre de chiroptères dans un pkl au format d'entrée du script d'entraînement.
Conversion des csv de PSIBIOM en .pkl pour les 5 genres de chiroptère nécessaire.
/NAS7/SABIOD/SITE/PSIBIOM/methode/Chiro/Exp02_small_detec_multi_Chiro/
Entrainement du modèle sur les données PSIBIOM (5000 annotations par genre de chiroptères) via small_detect_multi en lançant le script :
python3
/NAS7/SABIOD/SITE/PSIBIOM/methode/Chiro/small_detec_multi_Chiro/train_stft_nfeat_Chiro_PSIBIOM.py -annot_pkl 'binary_annot_Murin_annot_input.pkl' -exp_name 'Murin_240715_15h16'

Résultats :
/NAS7/SABIOD/SITE/PSIBIOM/methode/Chiro/Exp02_small_detec_multi_Chiro/runs_TVX/PSIBIOM2024_Chiro_uniclass/

Les résultats obtenus suite aux entraînements sont très prometteurs et demandent à être vérifiés sur des données hors données d'entraînement.

Murins validation metrics :
loss: 0.030581936240196228
roc_auc: 0.9895447229610823
mAP: 0.987494609748264
mAP_detail: ['0.987']

Pipistrelles validation metrics :
loss: 0.05639248341321945
roc_auc: 0.9875586795330638
mAP: 0.9869339479435758
mAP_detail: ['0.987']

Noctules validation metrics :
loss: 0.02159726247191429
roc_auc: 0.9852164729261964
mAP: 0.9738505652275887
mAP_detail: ['0.974']

Plecotus validation metrics :
loss: 0.019728997722268105
roc_auc: 0.9872115729428453
mAP: 0.9755474009630744
mAP_detail: ['0.976']

Rhinolophes validation metrics :
loss: 0.04157653823494911
roc_auc: 0.996006411944522
mAP: 0.9943451552096044
mAP_detail: ['0.994']

A noter que ces résultats ont été obtenus avec les valeurs de l'architecture cachalot dont le modèle a pu être adapté à notre usage.

Expérience - Get_Clicks

Méthode :

Le script get_click est à adapter pour détecter les vocalises de chiroptère et les classifier par fréquences correspondant aux différents genres de chiroptères.

Résultats :

Les fréquences telles qu'annotées dans le csv d'annotations reçu pour chaque genre de chiroptères :

Général sur le csv : [28 rows x 64 columns]
La freq min est -10.0 Hz
La freq max est 191250 Hz
La freq moyenne est 9847 Hz

Pipistrelles :
La freq min est 0.0
La freq max est 191250.0
La freq moyenne est 10114

Murins :
La freq min est 0.0
La freq max est 191250.0
La freq moyenne est 14067

Rhinolophes :
La freq min est 0.0
La freq max est 191250.0
La freq moyenne est 7342

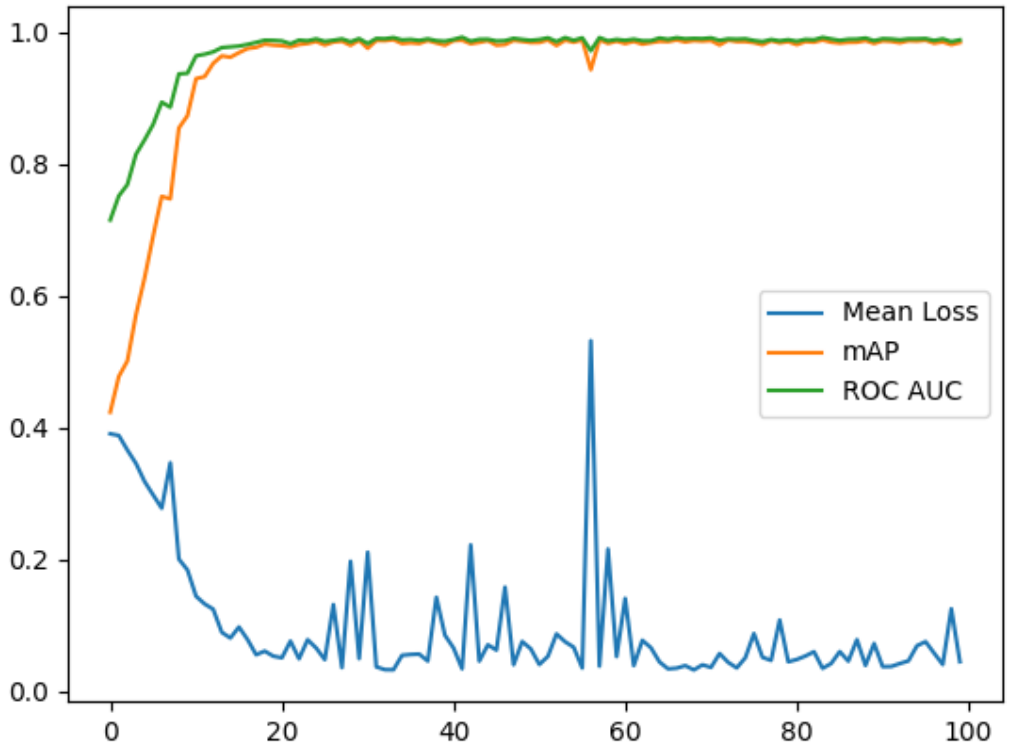
Noctules :
La freq min est 0.0
La freq max est 95250.0
La freq moyenne est 9374

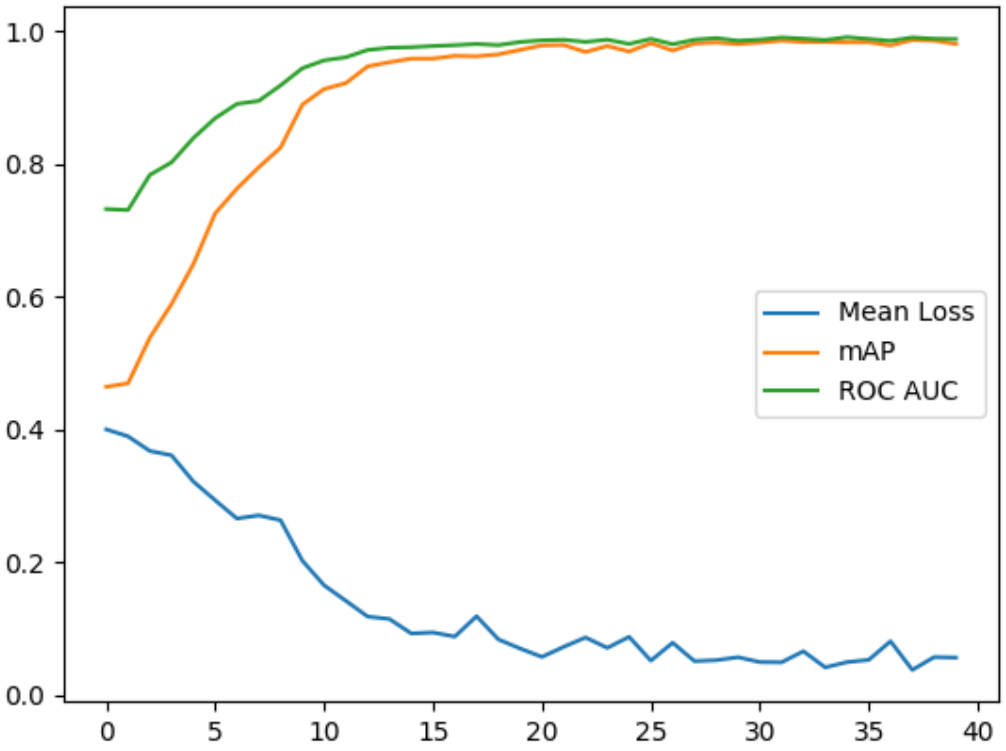
Plecotus (Oreillard) :
La freq min est 0.0
La freq max est 120000.0
La freq moyenne est 8836

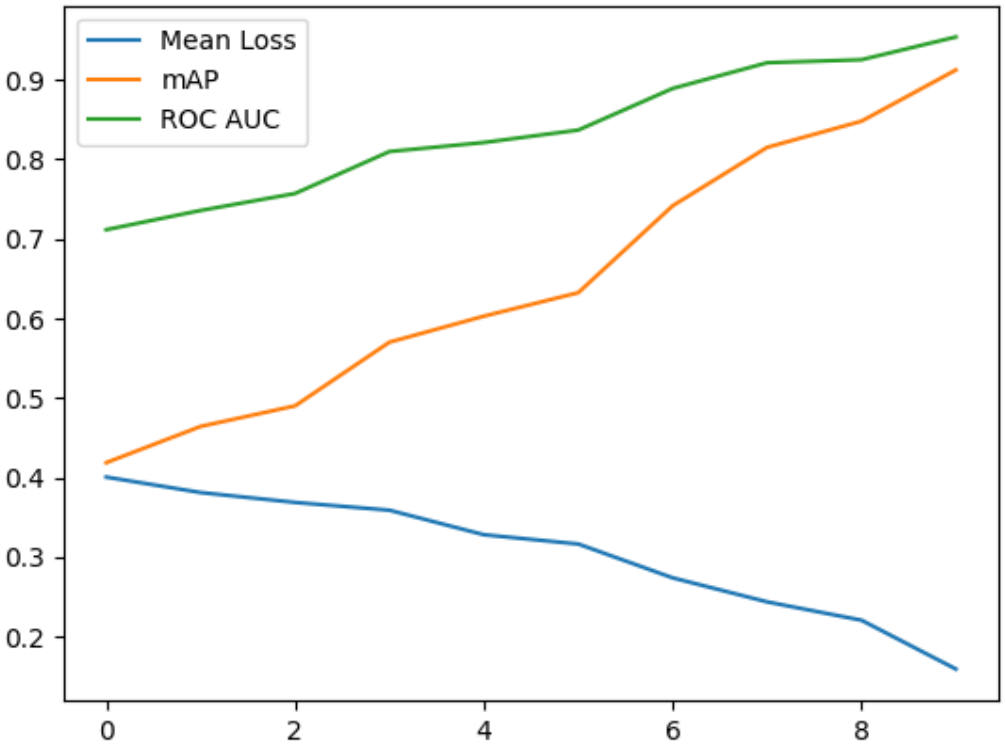
Expérience 4 - Small detec uniclass

Méthode :

Création d'un set de test en prenant les derniers fichiers contenant 10% des annotations. On obtient donc un set de test avec 10% des annotations, et un set d'entraînement avec 90% des annotations.
Le set d'entrainement est séparé en 2 par le script d'entrainement (/NAS7/SABIOD/SITE/PSIBIOM/methode/Chiro/Exp04_small_detec_Chiro_train+test/train_stft_nfeat_Chiro_PSIBIOM.py)
Il est découpé en 70% pour un set de train et 30% pour un set de validation.




1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25
30
35
40
Mean Loss
mAP
ROC AUC


Mean Loss
mAP
ROC AUC
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
2
4
6
8

Validation du modèle entraîné sur 15 epochs =

Tests du modèle 15 sur 10% du set avec Freq à 24kHz :
mAP 0.08381572452168186
ROC AUC 0.4318643450027537

Tests du modèle 10 epochs sur 10% du set avec Freq à 24kHz:
mAP 0.08041307698910265
ROC AUC 0.4202491657324738

Tests du modèle 20 epochs sur 10% du set : avec Freq à 32kHz :
mAP 0.2957695020378758
ROC AUC 0.5882162770495174

Tests du modèle 15 sur 10% du set avec Freq à 32kHz :
mAP 0.09794643652522673
ROC AUC 0.5083443725094671

Tests du modèle 15 sur 10% du set avec Freq à 32kHz :
```
nepoch, int16, norm, batch_size, fe, sampleDur = 20, True, False, 16, 32000, 2.5
```
mAP 0.09794643652522673
ROC AUC 0.5083443725094671

Tests du modèle 20 epochs sur 10% du set : avec Freq à 60kHz :
```
nepoch, int16, norm, batch_size, fe, sampleDur = 20, True, False, 16, 60000, 2.5
```
mAP 0.14998286996254828
ROC AUC 0.6025205670954195

Tests du modèle 20 epochs sur 10% du set : avec Freq à 60kHz :
```
nepoch, int16, norm, batch_size, fe, sampleDur = 20, True, False, 16, 60000, 5
```
mAP 0.18710335890077517
ROC AUC 0.6376751099038027

Tests du modèle 20 epochs sur 10% du set : avec Freq à 60kHz :
```
nepoch, int16, norm, batch_size, fe, sampleDur = 20, True, False, 16, 195000, 5
```
mAP 0.11059708751811921
ROC AUC 0.41146955549388886

Tests du modèle 20 epochs sur 10% du set : avec Freq à 60kHz :
```
nepoch, int16, norm, batch_size, fe, sampleDur = 20, True, False, 16, 100000, 5
```
mAP 0.08466003416438231
ROC AUC 0.43647583530708994

Tests du modèle 20 epochs sur 10% du set : avec Freq à 60kHz :
```
nepoch, int16, norm, batch_size, fe, sampleDur = 20, True, False, 16, 80000, 5
```

mAP 0.22455526808407072
ROC AUC 0.6509422407108973

Tests du modèle 20 epochs sur 10% du set : avec Freq à 60kHz :

```
nepoch, int16, norm, batch_size, fe, sampleDur = 20, True, False, 16, 90000, 5
```

Conclusion

Les indices éco-acoustiques sont des outils disponibles qui restent à être adaptés au langage C sur la carte embarquée. Concernant la partie IA/tâche de classification, son intégration sur la carte embarquée est en partie faite, il manque un test sur des enregistrements positifs contenant les espèces désirées.

Sur les Chiroptères nous adaptons un modèle de détection de pic d'énergie avec reconnaissance des genres basés sur les caractéristiques de fréquences et autres paramètres. En parallèle nous avons déjà un modèle IA entraîné en 5 détecteurs permettant la classification des 5 genres.

De façon générale, le déploiement de façon industriel de l'étude acoustique sur les 10 cartes QHB va demander un temps pour aménager la configuration dans le sens désiré: en premier lieu calcul des indices éco-acoustiques puis selon le cas/seuil lancement des modèles IA ou filtre-fréquences des Chiroptères. Un temps pour s'adapter à la version de la carte déployée QHBv3, dernière version de la carte son qui est différente de la QHBv2 testé au laboratoire sera aussi nécessaire.

Tâches restantes :

- Intégrer le calcul des indices éco-acoustiques sur la carte embarquée.
- Finir un modèle de classifications des 5 groupes de Chiroptères basé sur un filtre de fréquence
- Intégrer les poids et architecture des 2 classifieurs Anoures, Oiseaux, Orthoptère, Hémiptères et celui des 10 espèces d'Oiseaux ensemble
- Adapter le code configuration utilisé à la carte QHBv3
- 9 cartes QHBv3 à envoyer à SiConsult pour tests courant septembre.

ANNEXES

Script du fichier de lancement de test :

```
import matplotlib.pyplot as plt
from torch.utils.tensorboard import SummaryWriter
import torch
import numpy as np
from torch import nn, tensor, utils, load, device, cuda, optim, long, save
from torch.utils import data
import pandas as pd
import utils_copy as u
from tqdm import tqdm
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score, average_precision_score, roc_curve,
precision_recall_curve
from models import get
import argparse
import os

parser = argparse.ArgumentParser(description='')
parser.add_argument('-annot_csv', type=str, default='Test10_binary_annot_Murin_annot_input.csv')
#parser.add_argument('-weight', type=str,
default='Chiro_kHzhps256_512_k5_specBN_sch97_Murin_30ep_240829_14h08')
parser.add_argument('-nfeat',  type=int, default=512)
parser.add_argument('-kernel', type=int, default=5)
parser.add_argument('-nMel', type=int, default=128)
parser.add_argument('-repeat', type=str, default='')
parser.add_argument('-exp_name', type=str, default='')
args = parser.parse_args()

df = pd.read_csv(args.annot_csv)
#df = pd.read_csv(os.path(args.annot_csv))

nepoch, int16, norm, batch_size, fe, sampleDur = 100, True, False, 16, 24000, 2.5
modelname = f'Chiro_kHzhps256_512_k5_specBN_sch97_Murin_10ep_240903_14h05.stdc'
#modelname = f'{args.weight}.stdc'
#modelname =
f'dw_mel{args.nMel}_brown_24kHzhps256_{args.nfeat}_k{args.kernel}_specBN_sch97_{args.exp_
name}.stdc'

print('Go for model '+modelname)

#traindf, testdf = train_test_split(df, test_size=0.1, stratify=df.annot)
testdf = df
print('test size is '+str(len(testdf))+' with '+str((testdf.annot).sum())+' positives')
```

```
test_set = u.Dataset(testdf, fe=fe, norm=norm, int16=int16, sampleDur=sampleDur)
loaderTest = utils.data.DataLoader(test_set, shuffle=False, batch_size=batch_size, num_workers=4,
pin_memory=True)
model = get['chiro'](args.nfeat, args.kernel)
#model = get['chiro'](args.nfeat, args.kernel, hps=256, nMel=args.nMel)
model.load_state_dict(torch.load(modelname))
gpu = device('cuda:1')
model = model.to(gpu)
writer = SummaryWriter('runs/'+modelname)

with torch.inference_mode():
    model.eval()
    labels, preds, losses, fullpreds = [], [], [], []
    for batch in loaderTest:
        x, label = batch
        pred = model(x.to(gpu)).squeeze().cpu().detach()
        preds.extend(pred.view(-1))
        labels.extend(label.view(-1))

print('mAP', average_precision_score(labels, preds))
print('ROC AUC', roc_auc_score(labels, preds))
prec, rec, thr = precision_recall_curve(labels, preds)
plt.plot(rec, prec)
plt.savefig(f'AP_{modelname[:-4]}pdf')
```