OpenSoundID 2.0
Quick Start.

Open source AI for
wildlife identification and
biodiversity measurement

OpenSoundID is powered by

Xeno-Canto : Bird sounds Database
Essentia : Audio features extractor from university Pompeu Fabra of Barcelona
Weka : Machine Learning workbench with GUI from university of Hamilton, New Zealand
Classifier : Deep Learning for java

OpenSoundID is powered by

SOX : audio transformations
FFMPEG : audio transformations
Centos/Fedora : Linux distribution
Java 11 : opensoundid is developed in java
H2 : Database Engine
Ansible : Tasks automotion

Hardware prerequisite

- **Hardware for 100 birds species apart training step**
  - **Modern CPU (AMD 5950x, ...)**
  - **32 GO RAM**
  - **1 TO disk (M2 PCIE 4.0)**
- **Hardware for 100 birds species for training step**
  - **Modern CPU (AMD 5950x, ...)**
  - **32 GO RAM**
  - **1 TO disk (M2 PCIE 4.0)**
  - **GPU 32GO (Nvidia Tesla V100s, ...)**

Software prerequisite

- **OS : Fedora 34 (server or worksation edition)**
- **Ansible ≥ 3.9**
- **git**
- **Internet access without proxy**
- **Nvidia driver for GPU**

01 Install OpenSoundID

- **Two options : local or remote install**
  - **Remote install on host « hostname » with user « username » :**
    - ***git clone https://github.com/openSoundID/ansible-opensoundid-installer.git***
    - ***cd ansible-opensoundid-installer***
    - ***./install.sh -remote hostname username***
  - **Local install :**
    - ***git clone https://github.com/openSoundID/ansible-opensoundid-installer.git***
    - ***cd ansible-opensoundid-installer***
    - ***./install.sh -local***

02 Main directories

After install take a look at directories on the opensoundid home user with the following commands:

- ***su - opensounid***
- ***tree -L 1***

ansible	ansible playbooks directory
bin	binary directory
database	database directory
dataset	dataset directory (MP3, WAV, ...)
inventory	inventory directory
javasrc	java sources
lib	java libraries directory
logs	logs files directory
metadataRepository	directory for XENO-CANTO metadata
model	directory for deep-learning model
noises	directory for noises wav file
properties	directory for opensoundid parameters
results	directory for results of steps
scoreLogger	directory for analyze statistique
soundAnalyzer	directory for sound analyze

03 List the birds to be identified

- **create a yaml file named birds_list.yml containing the list of birds you want to identify, there is an example file in the documentation repository:**

```
birds_list:
  - bird_record:
      id: 1000 => bird id
      enName: "Great Tit" => bird name
      frName: "Mésange charbonnière" => French translation
      Genre: "Parus" : bird genre
      Espece: "major" : bird espece (be careful Genre and Espece are xeno-canto keys)
      claims:
        - claim_record:
            q: a : record Quality (a,b,c, …) a is best.
            cnt:
              - France : Country of recording
              - Spain : Country of recording
```

- **Important don’t forget to validate you yaml file with yaml validator online site !**
- **Then put birds_list.yml in the properties directory**

04 Compile java programs

- **Goal : compile java programs for your specific hardware**
  - **For GPU hardware**
    - ***osid -cpcuda-11.2***
  - **For CPU hardware with AVX/2 instructions**
    - ***osid -cpavx2***
  - ***For CPU hardware with AVX/512 instructions***
    - ***osid -cpavx512***
  - ***For CPU hardware with OneDNN and AVX/2 instructions***
    - ***osid -cponednn-avx2***
  - ***For CPU hardware with OneDNN and AVX/512 instructions***
    - ***osid -cponednn-avx512***
- ***After compilation check that jar file lib/engine-1.0.0-SNAPSHOT.jar exist.***

05 Create database

- **Goal : create H2 database and schema by typing the following command**
  - ***osid -cdb***
- ***After this command check that database file opensoundid.mv.db exist in database directory***

06 Get metadata from XC

- **Goal : get metadata from Xeno-Canto and make inventory**
  - ***osid -m***
- **After this commabd check that training inventory file « training_inventory.yml » and test inventory file « test_inventory.yml » exist in inventory directory.**

07 Make Dataset

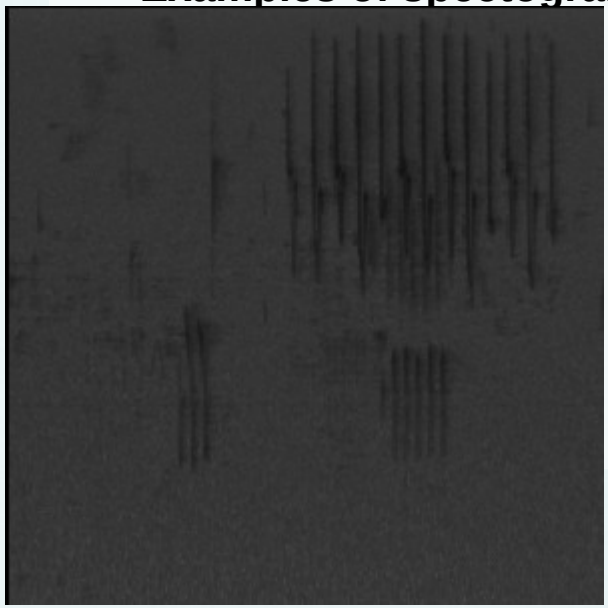
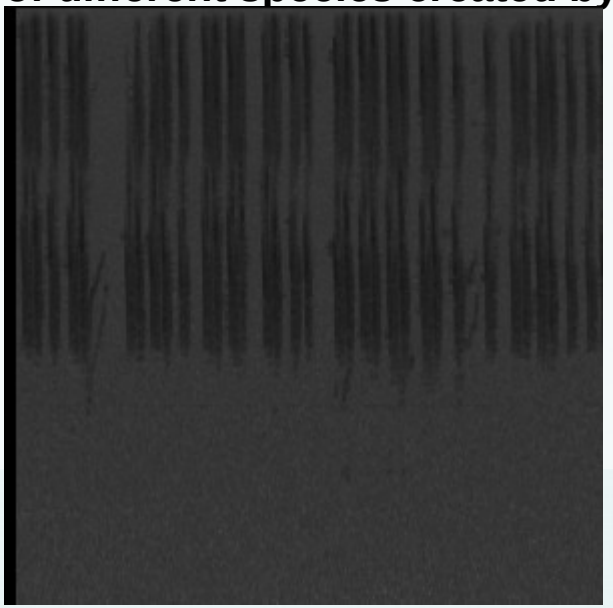
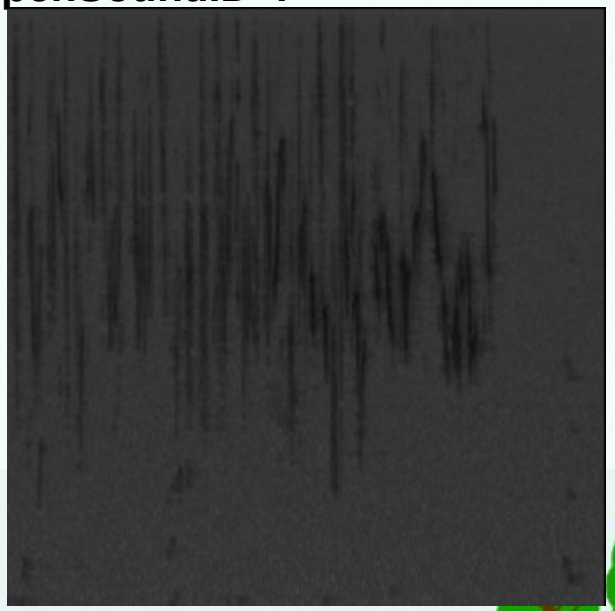
- **Goal : Download MP3 from Xeno-Canto, convert MP3 to MFCC spectrograms**
  - ***osid -md***
- **After executing this command check that MP3 files have been downloaded in « dataset/training/DownloadDirectory » and « dataset/test-xcalso/DownloadDirectory ».**
- **Check that MFCC spectrograms have been generated in dataset/training/JSONDirectory and dataset/test-xcalso/JSONDirectory**

08 Update Database

- **Update database to disable not suitable Xeno-Canto records. (too short records, too high pass filter, ...)**
  - ***osid -udb***

09 Make features

- **Transform MFCC spectograms to PNG images suitable for deep learning.**
  - ***osid -mf***
- **After executing this command check thas PNG images are available in « dataset/training/spectrogramsDirectory » and**
- **Examples of spectogram of different species created by openSoundID :**

10 Training step

- **Goal : train neural networks and get model file.**
  - ***osid -t***
- **Be careful, training can take a very long time, the use of a GPU is strongly recommended.**
- **After training step check that file named « bestGraph.bin » in model directory exists.**

11 Classification step

- **Goal : Get the score of your model from the two tests dataset.**
  - ***osid -cl***

- ***==========Evaluation Metrics==========***
- ***# of classes: 5***
- ***Accuracy: 0,8538***
- ***Precision: 0,8643***
- ***Recall: 0,8553***
- ***F1 Score: 0,8571***
- ***Precision, recall & F1: macro-averaged (equally weighted avg. of 5 classes)***
-
- ***==========Confusion Matrix==========***
- ***0 1 2 3 4***
- ***---***
- ***398 52 24 20 6 | 0 = 1000***
- ***17 470 10 3 0 | 1 = 2000***
- ***98 10 377 11 4 | 2 = 3000***
- ***44 21 6 429 0 | 3 = 4000***
- ***15 8 9 0 416 | 4 = 5000***

12 Analyze sound

- **Goal : Start soundAnalyzer daemon and Analyze wav files**
  - ***sudo systemctl enable soundAnalyzer.service***
  - ***sudo systemctl start soundAnalyzer.service***
  - ***analyze.sh -i fichier.wav -t yyyymmddhhmmss -w true -d true***

  Give the following result :

  - **Class xxx:yyy => the class xxx obtenaid a score of yyy**