Projet: *Embedded Machine Learning*

OBJECTIFS, DÉROULEMENT, LE CAS D'APPLICATION ET LES OUTILS

Sommaire

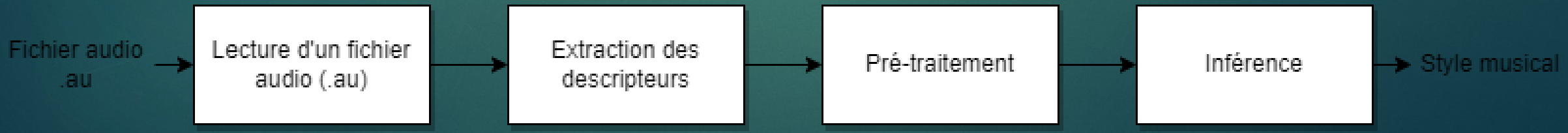
- Objectifs
- Présentation du cas d’application
- Déroulement des séances
- Principes de développement
- Lecture des fichiers .au
- Extraction des descripteurs audio
- Classification
- Moyens
- Évaluation

Objectifs

1. Implémenter les différentes phases des algorithmes de ML pour l’embarqué : Extraction de paramètres, apprentissage, inférence.
2. Apprendre à choisir les modèles ainsi qu’à les optimiser pour une implémentation sur une cible embarquée
3. Connaître les différents types d’implémentations en IA embarqué
4. Interpréter et implémenter un standard audio comme .au
5. Découvrir des outils comme emlearn et Tensorflow Lite
6. Implémenter un cas d’application sur une cible restreinte (RPI 4B)

Présentation du cas d'application
Reconnaissance de style musical

- La reconnaissance de style **musical** d'après un extrait audio de 30 secondes.
- Base de données GTZAN [1]:
  - 1000 pistes audio de 30 secondes
  - Format: .au (ou .wav)
  - Monocanal, 16 bits linear PCM (*Pulse Code Modulation*), échantillonné à 22050 Hz
  - 10 classes: Blues, Classique, Country, Disco, Hiphop, Jazz, Metal, Pop, Reggae, Rock



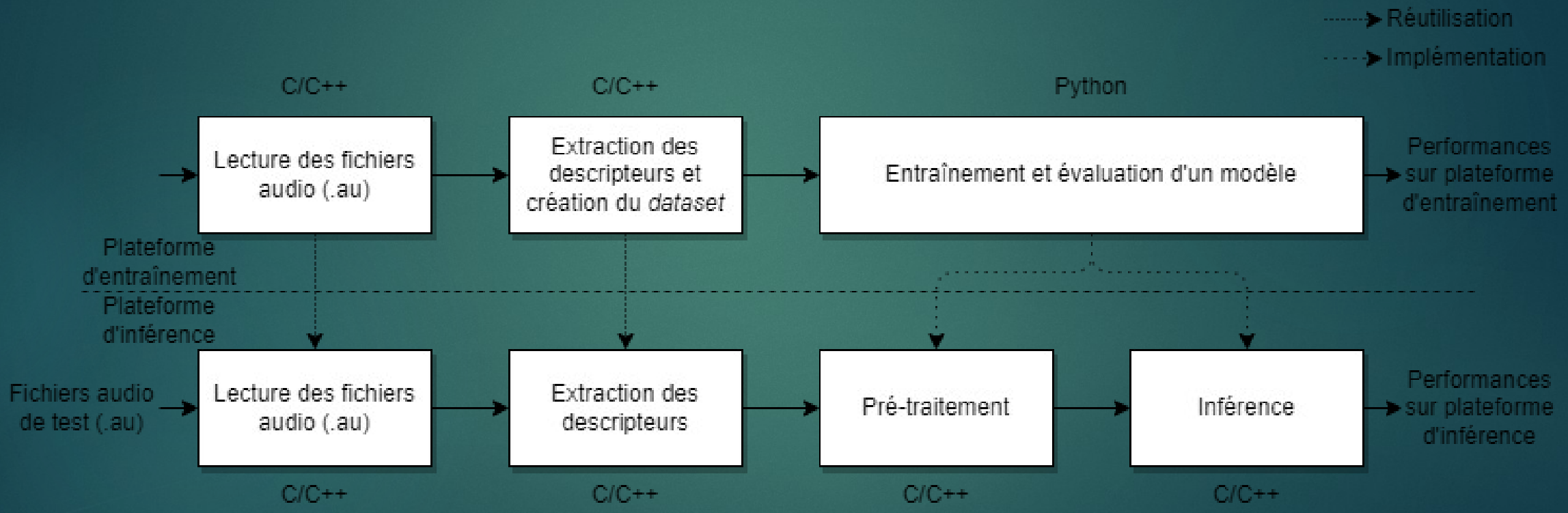
Implémentation finale

[1] – G. Tzanetakis et al. - Musical Genre Classification of Audio Signals

Déroulement des séances

- 4 Séances (16h):
  - Séance 1 (4h) – 12/12 (C1-C4):
    - Présentation du cours (2h)
    - Lecture des fichiers .au (2h)
  - Séance 2 (4h) – 19/12 (C1-C4):
    - Extraction des descripteurs (2h)
    - Séparateur à Vaste Marge (SVM) (2h)
  - Séance 3 (4h) – 09/01 (C1-C4):
    - Arbre de décision et forêts aléatoires (4h)
  - Séance 4 (4h) – 16/01 (C1-C4):
    - Réseaux de neurones avec Tensorflow Lite (4h)

Principes de développement
Le cas de la reconnaissance musical



Processus de développement

Lecture des fichiers .au

- Format .au: Format de fichier audio développé par Sun Microsystems en 1992 et très courant au début d'internet.

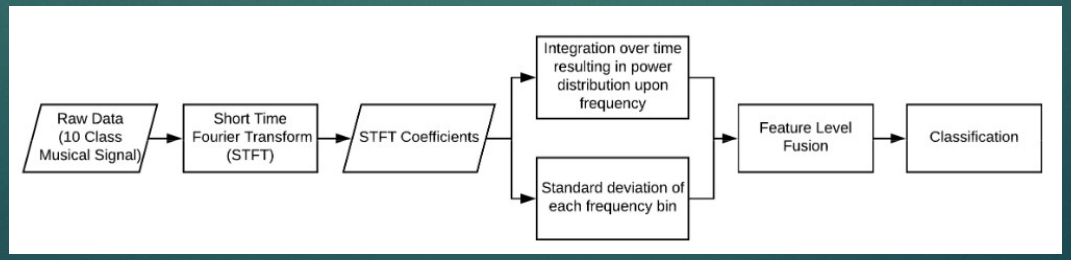
Mot 32 bits	Champ	Contenu pour GTZAN
0	Nombre magique	0x2e736e64 (779316836)
1	Décalage des données	-
2	Taille des données	~ 1 323 000 octets
3	Encodage	3 (16 bits linear PCM)
4	Fréquence d'échantillonnage	22050 Hz
5	Nombre de canaux	1 (monocanal)

Entête d'un fichier .au (GTZAN)

Extraction des descripteurs audio

- Spectrogramme (μ,σ):

- Implémentation à l'aide d'une transformée de Fourier rapide (FFT: *Fast Fourier Transform*)
- Descripteurs ([2]: μ,σ):
  - μ: Moyenne temporelle du spectrogramme
  - σ: Ecart-type temporel du spectrogramme

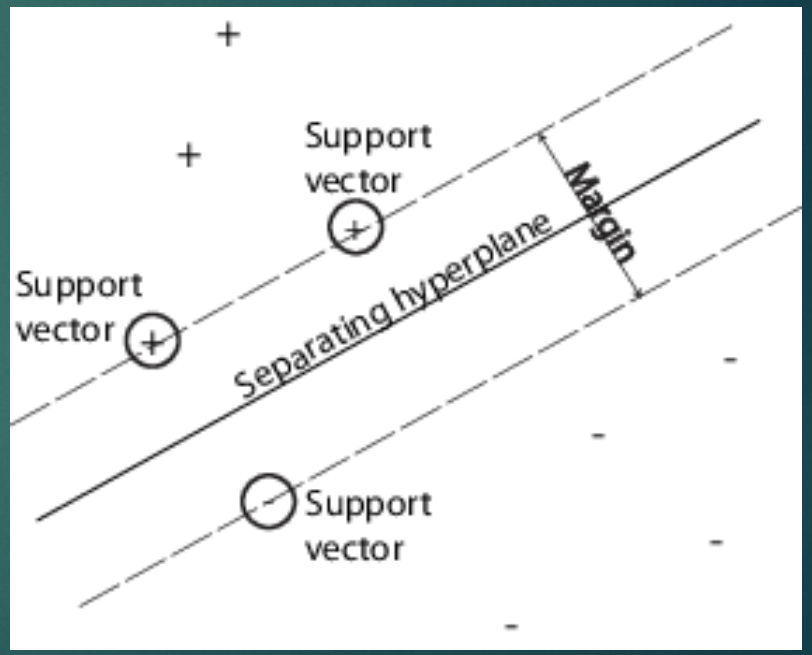


Processus d'extraction des descripteurs [2]

[7] Ahmet Elbir et al. - Short Time Fourier Transform Based Music Genre

Classification:
Séparateur à Vaste Marge (SVM)

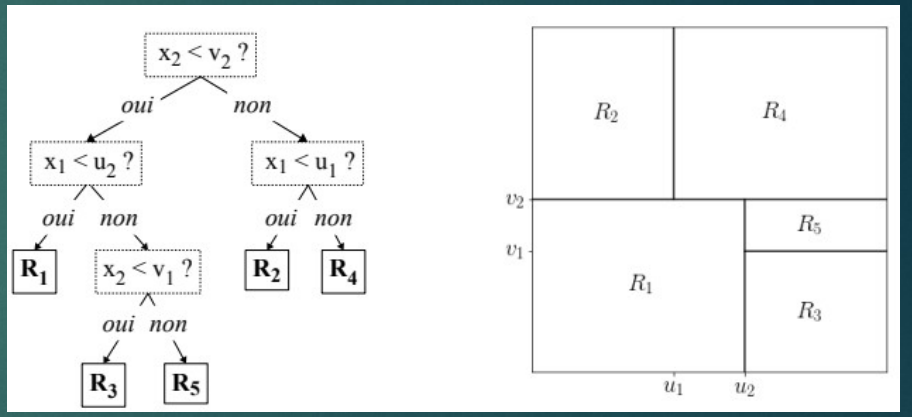
- Classifieur binaire linéaire [3]
- Classification multi-classe :
  - OneVsOne: classifieurs
  - OneVsAll: n classifieurs
- Décision interprétable par l'être humain.
- Programmation (problème primal):
  - Python: Scikit-Learn (*LinearSVC*)
  - C/C++: « Codage manuel »



[3] A. Guéron – Machine Learning avec Scikit-

Classification: Arbres de décision et forêts aléatoires (RF)

- Arbre de décision :
  - Utilisation de données quantitatives et qualitatives,
  - Création d'un arbre binaire,
  - Classification multi-classe.
- Décision interprétable par l'être humain.
- Programmation:
  - Python: Scikit-Learn (*DecisionTreeClassifier*)
  - C/C++: Génération de code (emlearn)
- Implémentation d'une forêt aléatoire (Python: *RandomForestClassifier*)

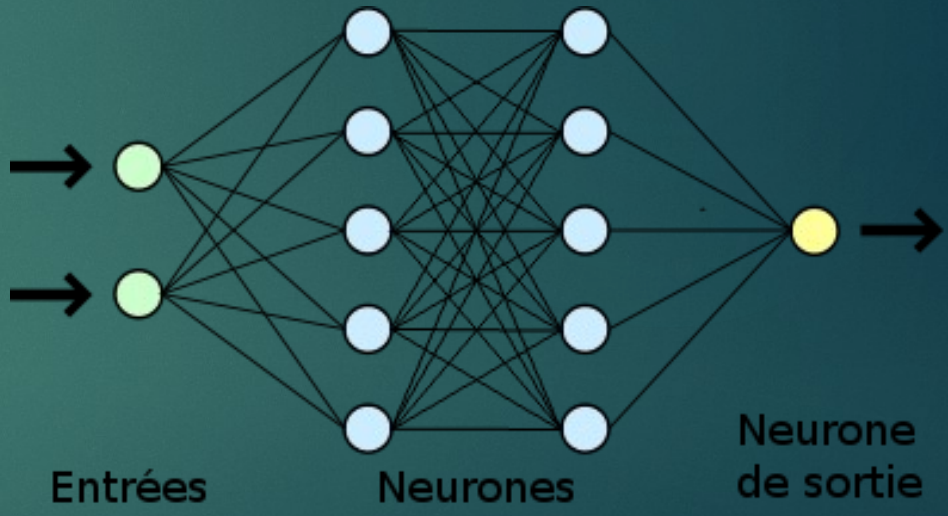


Partionnement [4]

[4] C.-A. Azencott – Introduction au Machine

Classification: Réseaux de neurones denses (NN)

- Classifieur multi-classe [5]
- Algorithmes de classification puissants mais interprétabilité réduite
- Programmation:
  - Python: Tensorflow (Keras)
  - C/C++: Moteur d'inférence (Tensorflow Lite)
- Evaluation de différentes optimisations du modèle [6, 7]



Réseau de neurones dense

[5] A. Guéron – Deep Learning avec Keras et Tensorflow
[6] Machine Learning sur des objets connectés – Linux Magazine n°239
[7] P. Warden et al. - TinyML: Machine Learning With Tensorflow Lite on Arduino and Ultra-Low-Power Microcontrollers

Moyens

- Mise à disposition:
  - Une Raspberry Pi 4 B
  - Une image avec Raspbian Buster (10.7) et Tensorflow Lite
  - Une implémentation C++ d’une FFT
  - Un ensemble de ressources pour l’implémentation

Raspberry Pi 4

- Kit Raspbbery Pi 4:
  - Raspberry Pi 4 Model B 4 GB
  - MicroSD 16 GB, préchargée avec NOOBs
  - Boîtier noir Raspberry Pi 4
  - Alimentation 5,1V 3A
  - Câble micro HDMI/HDMI

Évaluation

- **Évaluation pour le 30/01/2022 (dépôt Moodle) :**
  - Rapport au format PDF police 11 de 10 pages maximums qui rend compte précisément de votre travail. Ce rapport ne contient pas l'intégralité du code, mais propose une analyse des parties pertinentes. Une analyse comparative des performances entre les différents algorithmes et les différentes implémentations est attendue.
  - Le code source (C/C++ et Python).

Évaluation, attendus et critères

- (**obligatoire**) Veiller à donner la preuve que le code fonctionne sur la cible.
- (**obligatoire**) Évaluer (et mesurer) les complexités temporelle et mémoire de vos algorithmes.
- (**obligatoire**) Vérifier la performance de chaque classifieur embarqué sur les fichiers de test sélectionnés (avant l'entraînement) afin de les comparer.
- Programmer une extraction de paramètres en C/C++ :
  - (**obligatoire**) Implémenter l'extraction de paramètres à l'aide de l'implémentation C++ de la FFT fournie
- Programmer une SVM :
  - (**obligatoire**) Implémenter une SVM basé sur le problème primal, à l'aide du modèle appris par Scikit-Learn (*LinearSVC*)
- Arbres de décision et forêts aléatoires:
  - (**obligatoire**) Générer automatiquement le code C/C++ de l'arbre de décision à l'aide du modèle appris par Scikit-Learn (*DecisionTreeClassifier*),
  - (**obligatoire**) Générer automatiquement le code C/C++ de la forêt aléatoire à l'aide du modèle appris par Scikit-Learn (*RandomForestClassifier*).
- Programmer un réseau de neurones :
  - (**obligatoire**) Implémenter un réseau de neurones dense en C/C++ à l'aide de Tensorflow Lite,
  - (**obligatoire**) Etude des optimisations possibles du réseau de neurones.
- (**obligatoire**) Comparer les approches et faire un tableau synthétique des caractéristiques et des performances de chacune des approches (SVM, Arbre de décision, RF, NN).
- (**optionnelle**) Proposer un système complet avec les choix architecturaux sous-jacent (matériels, logiciels, …).

Évaluation, attendus et critères

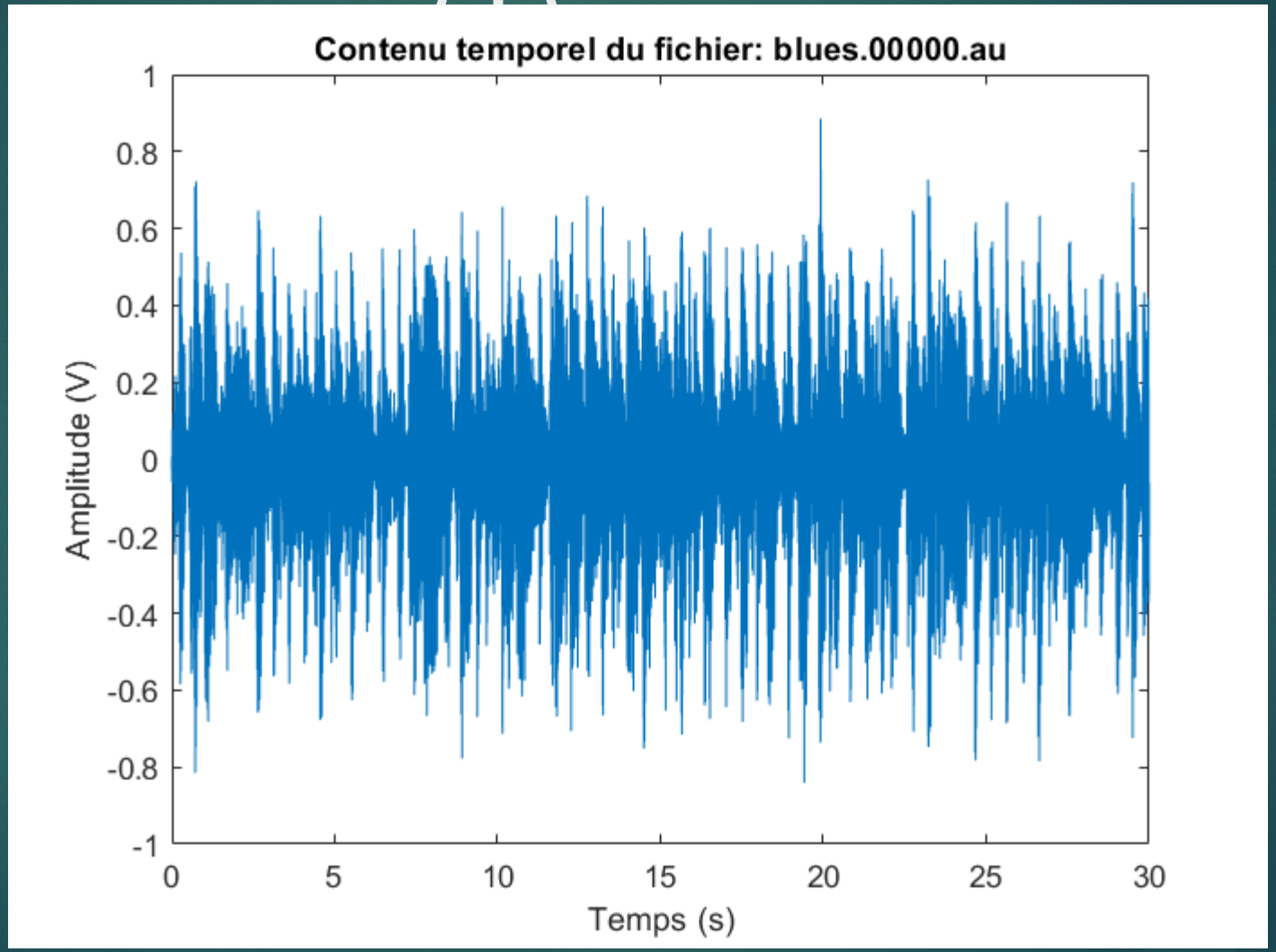
Rendu demandé	Points (obligatoire)	Points (facultatif)
Analyse des différentes approches	3	
Lecture des fichiers audio (.au)	2	
Extraction des descripteurs (FFT)	2	
Implémentation d'une SVM linéaire (problème primal)	2	
Implémentation d'un arbre de décision	2	
Implémentation d'une forêt aléatoire	2	
Implémentation d'un réseau de neurones dense	2	
Evaluation des optimisations du modèle	2	
Comparaison des différentes implémentations	3	
Proposition d'un système complet		2
Total	20	2

Merci pour votre écoute

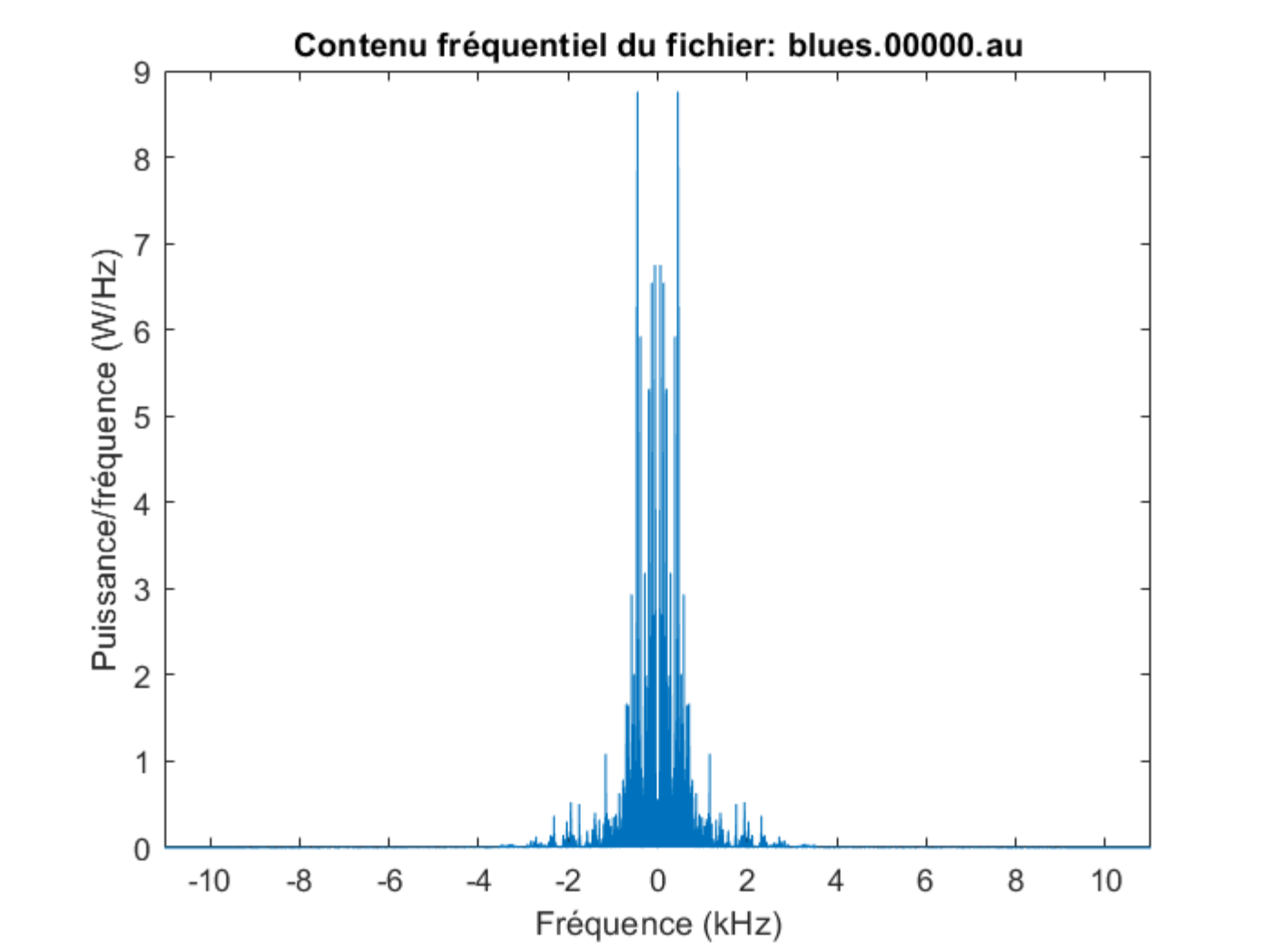
Bibliographie

- [1] G. Tzanetakis et al. - Musical Genre Classification of Audio Signals
- [2] Ahmet Elbir et al. - Short Time Fourier Transform Based Music Genre Classification
- [3] A. Guéron – Machine Learning avec Scikit-Learn
- [4] C.-A. Azencott – Introduction au Machine Learning
- [5] A. Guéron – Deep Learning avec Keras et Tensorflow
- [6] Machine Learning sur des objets connectés – Linux Magazine n°239
- [7] P. Warden et al. - TinyML: Machine Learning With Tensorflow Lite on Arduino and Ultra-Low-Power Microcontrollers

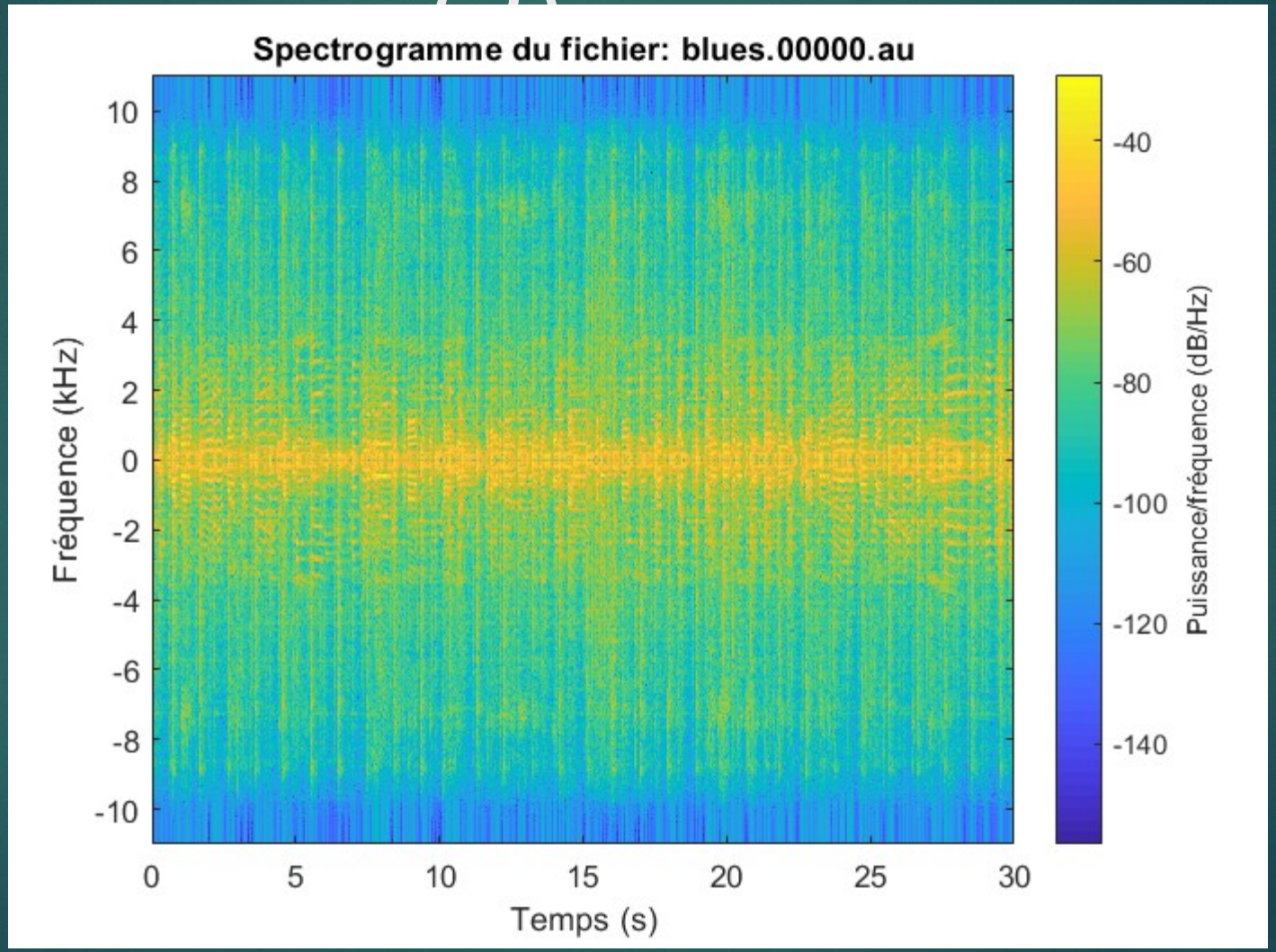
Spectrogramme et descripteurs (1)



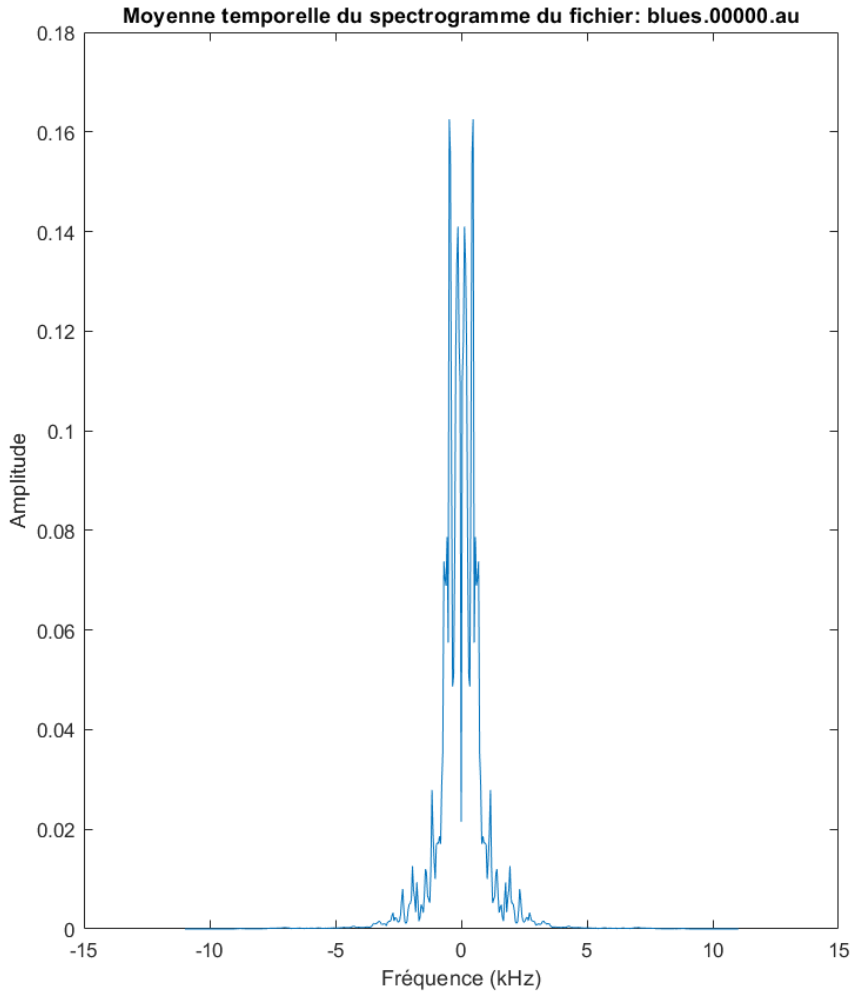
Spectrogramme et descripteurs (2)

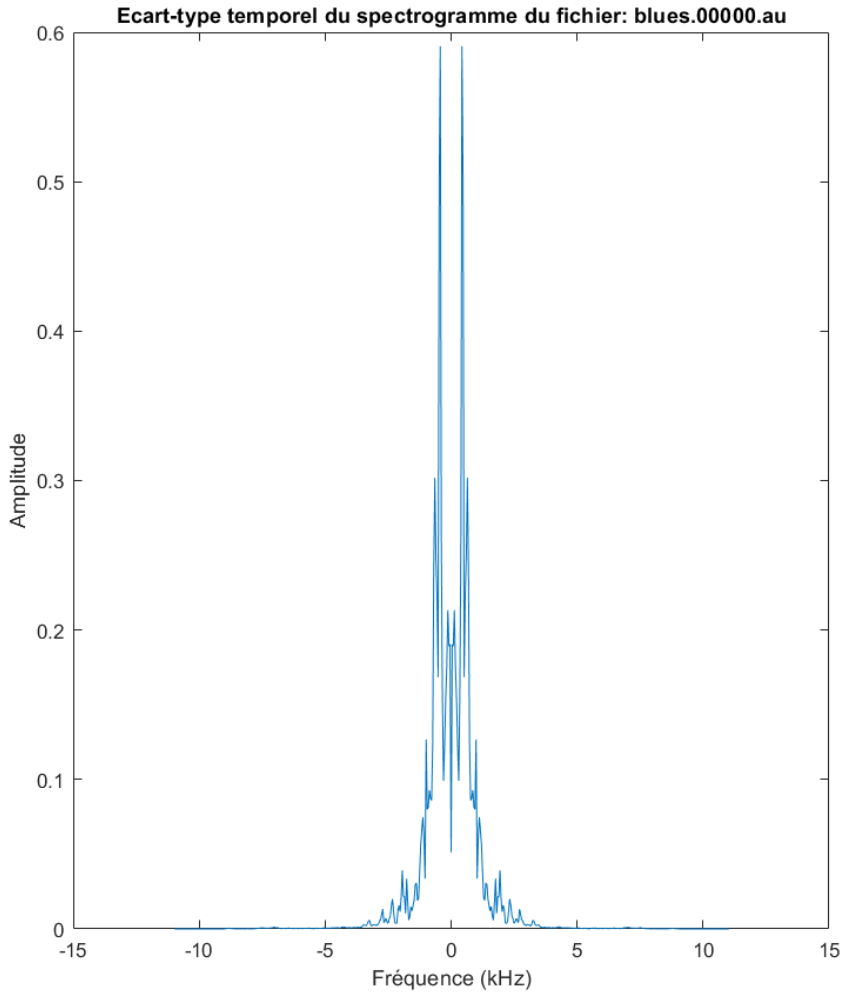


Spectrogramme et descripteurs (2)



Spectrogramme et descripteurs (4)





Comparaison des langages de programmation

Table 4. Normalized global results for Energy, Time, and Memory

Total					
	Energy		Time		Mb
(c) C	1.00	(c) C	1.00	(c) Pascal	1.00
(c) Rust	1.03	(c) Rust	1.04	(c) Go	1.05
(c) C++	1.34	(c) C++	1.56	(c) C	1.17
(c) Ada	1.70	(c) Ada	1.85	(c) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(c) C++	1.34
(c) Pascal	2.14	(c) Chapel	2.14	(c) Ada	1.47
(c) Chapel	2.18	(c) Go	2.83	(c) Rust	1.54
(v) Lisp	2.27	(c) Pascal	3.02	(v) Lisp	1.92
(c) Ocaml	2.40	(c) Ocaml	3.09	(c) Haskell	2.45
(c) Fortran	2.52	(v) C#	3.14	(i) PHP	2.57
(c) Swift	2.79	(v) Lisp	3.40	(c) Swift	2.71
(c) Haskell	3.10	(c) Haskell	3.55	(i) Python	2.80
(v) C#	3.14	(c) Swift	4.20	(c) Ocaml	2.82
(c) Go	3.23	(c) Fortran	4.20	(v) C#	2.85
(i) Dart	3.83	(v) F#	6.30	(i) Hack	3.34
(v) F#	4.13	(i) JavaScript	6.52	(v) Racket	3.52
(i) JavaScript	4.45	(i) Dart	6.67	(i) Ruby	3.97
(v) Racket	7.91	(v) Racket	11.27	(c) Chapel	4.00
(i) TypeScript	21.50	(i) Hack	26.99	(v) F#	4.25
(i) Hack	24.02	(i) PHP	27.64	(i) JavaScript	4.59
(i) PHP	29.30	(v) Erlang	36.71	(i) TypeScript	4.69
(v) Erlang	42.23	(i) Jruby	43.44	(v) Java	6.01
(i) Lua	45.98	(i) TypeScript	46.20	(i) Perl	6.62
(i) Jruby	46.54	(i) Ruby	59.34	(i) Lua	6.72
(i) Ruby	69.91	(i) Perl	65.79	(v) Erlang	7.20
(i) Python	75.88	(i) Python	71.90	(i) Dart	8.64
(i) Perl	79.58	(i) Lua	82.91	(i) Jruby	19.84

Comparaison des langues de programmation [A1]

[A1] Programmation : une étude révèle les langages les plus voraces en énergie
https://programmation.developpez.com/actu/253829/Programmation-une-etude-revele-les-langages-les-plus-voraces-en-energie-Perl-Python-et-Ruby-