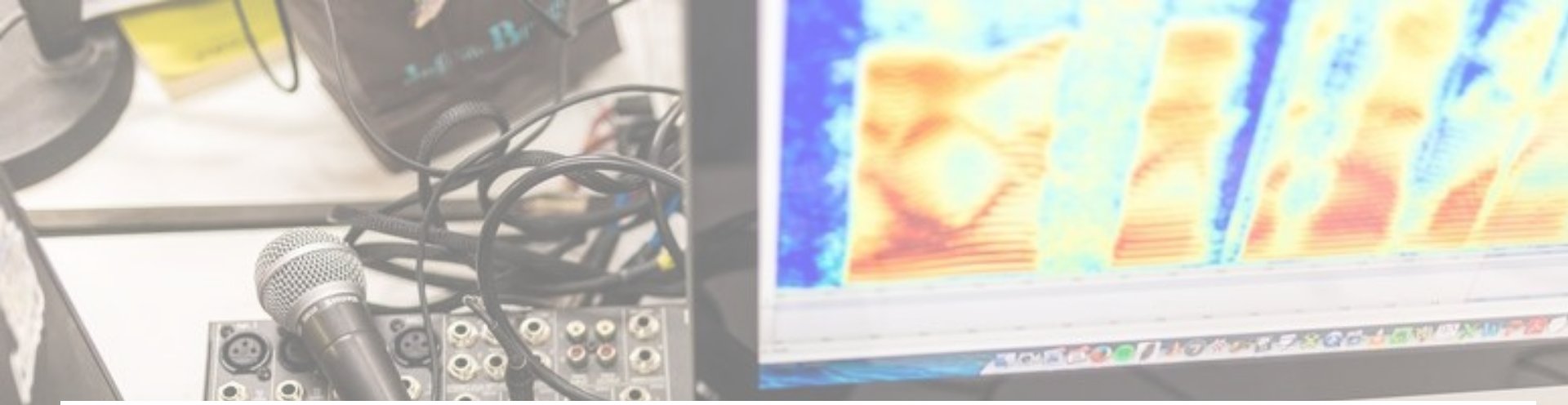


Traitement avancé de la parole et des signaux audio: Music Information Retrieval

06/11/2024
Alice Cohen-Hadria
alice.cohenhadria@gmail.com

Introduction

Dans ce cours, Music Information Retrieval (MIR) → Analyse automatique de la musique.

Nombreuses tâches :

- Détection de voix, d'instruments
- Estimation de structures musicales
- Transcription automatique: estimation de F0
- Séparation de sources
- Cover detection
- …

Comment ?

Avant le deep learning :

Deux phases :

1. Choix de représentations (traitement du signal)
2. Apprentissage (SVM, classifieur simple ..)

Après :

Les deux phases sont faites en même temps, on apprend la représentation en même temps que la tâche

En pratique

Librairie open source en python pour l'analyse de musique et de l'audio:

https://librosa.org/doc/latest/index.html

Plan du cours

1. Représentations musicales
   a. Représentations temps-fréquences
   b. Descripteurs et SSM

2. Séparation de sources musicales
   a. Convolutional neural networks
   b. U-Net et Wave-U-Net

3. Estimations de F0 et Cover detection
   a. Harmonic CQT
   b. Représentation saillantes
   c. Cover detection et triplet loss

Représentation musicales

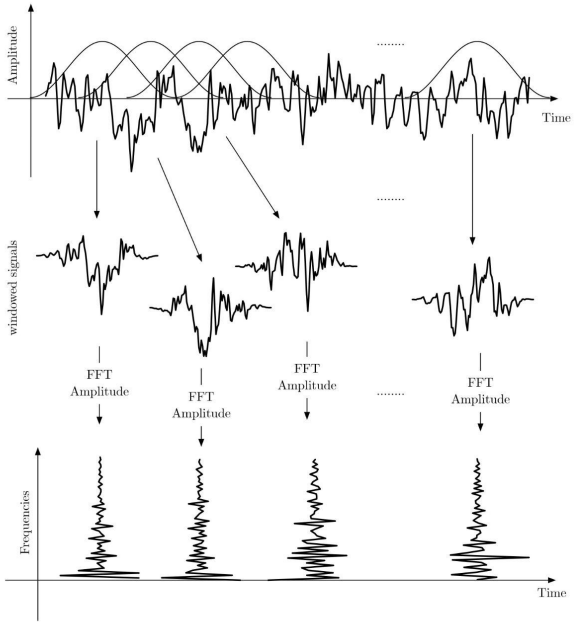
Short Time Fourier Transform (STFT)

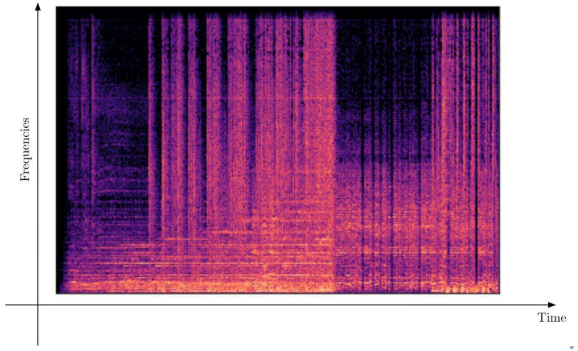
En francais TFCT (transformée de fourier a court terme).

Permet de représenter un signal audio (de musique) en temps et en fréquence.

On calcule des transformée de Fourier discrète (DFT, discrete Fourier transform) sur des petites fenêtres qui se superposent.

$$X[k,m] = \sum_{n=0}^{N-1} W[n-m]x[n]e^{-j2\pi kn/N}.$$

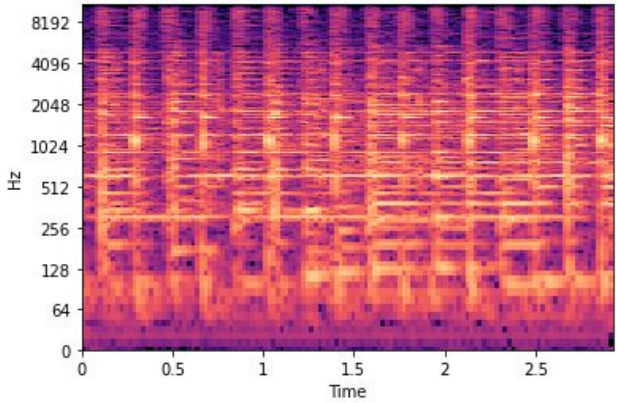




Short Time Fourier Transform (STFT)

Entrée [5]:
```
#load .wav
filename = "september.wav"
signal, sr = librosa.load(filename, sr=16000)

#compute abs(stft)
spec = np.abs(librosa.stft(signal[sr:sr*5]))

#show stft in db
librosa.display.specshow(librosa.amplitude_to_db(spec, ref=np.max),
                         y_axis='log', x_axis='time')
```

Out[5]: <matplotlib.collections.QuadMesh at 0x25a7b471100>



Constant Q transform (CQT)

Variation de la DFT, spécialement adaptée à l'analyse de la musique.

Dans la TFCT, la taille de la fenêtre est identique pour toutes les fréquences, et les fréquences sont espacées linéairement.

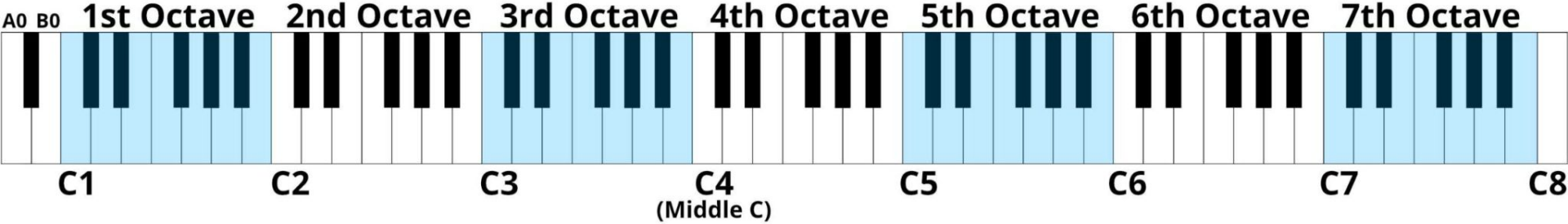
MAIS l'oreille humaine a une réponse logarithmique, et les notes suivent une progression logarithmique.

Notes de musique

Les fréquences sont calculées en utilisant la série logarithmique pour chaque octave, doublant la fréquence à chaque octave.

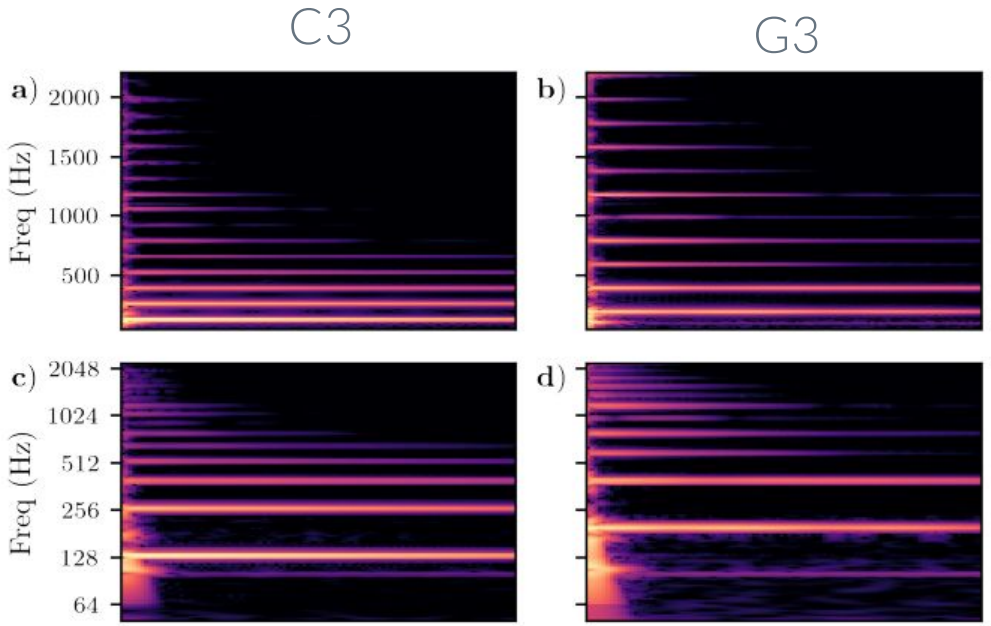
Les notes sont en accordage égal-tempéré, ce qui signifie que la fréquence double exactement tous les 12 demi-tons.

Notes de musique



Note	Octave 1	Octave 2	Octave 3	Octave 4	Octave 5
A	55 Hz	110 Hz	220 Hz	440 Hz	880 Hz
B	61.74 Hz	123.47 Hz	246.94 Hz	493.8 Hz	987.77 Hz
C	32.70 Hz	65.41 Hz	130.81 Hz	261.63 Hz	523.25 Hz

Constant Q transform (CQT)



Motif harmonique invariant, contrairement à la TFCT (la taille des espaces entre “les dents du peigne” harmonique varie).

Source : Automatic Tablature Estimation with Convolutional Neural Networks: Approaches and Limitations

Constant Q transform (CQT)

La transformation peut être considérée comme une série de filtres fk espacés logarithmiquement en fréquence, où le k-ième filtre a une largeur spectrale **δ**fk égale à un multiple de la largeur du filtre précédent.

$$X[k] = \frac{1}{N[k]} \sum_{n=0}^{N[k]-1} W[k,n]x[n]e^{\frac{-j2\pi Qn}{N[k]}}.$$

Fenêtre qui dépend de la fréquence k, taille N[k]

$$N[k] = q\frac{f_s}{f_k}$$

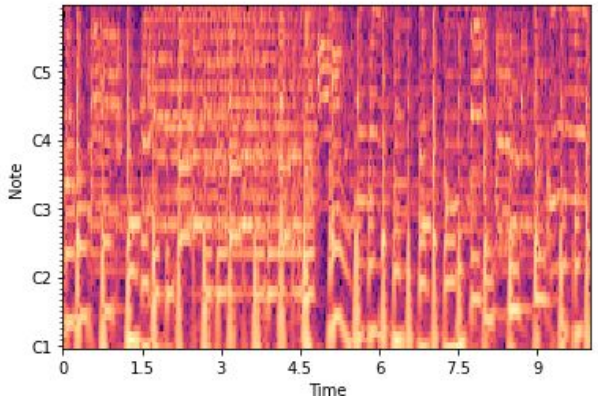
Facteur de qualité Q $\quad Q = \frac{f_k}{\delta f_k}.$

$$\delta f_k = 2^{1/n} \cdot \delta f_{k-1} = \left(2^{1/n}\right)^k \cdot \delta f_{\min},$$

Constant Q transform (CQT) - code

```
#loading file
filename = "september.wav"
signal, sr = librosa.load(filename, sr=16000)

#computing cqt
cqt = np.abs(librosa.cqt(signal[30*sr:40*sr], sr=sr,
                         fmin=librosa.note_to_hz('C2'),n_bins=60))

#plot
librosa.display.specshow(librosa.amplitude_to_db(cqt, ref=np.max),
                           sr=sr, x_axis='time', y_axis='cqt_note')
```

```
<matplotlib.collections.QuadMesh at 0x29ef6a55670>
```



Echelle des Mels

Vient de melody → mel

Échelle perceptive de la hauteurs de sons (unité mel). Elle se base sur la manière dont les humaines perçoivent les sons : sensibilité plus haute pour les sons aïgus que graves.

Conversion Hz ↔Mel:

$$m = 1128 \cdot \ln\left(1 + \frac{f}{700}\right) \qquad f = 700 \cdot \left(e^{\frac{m}{1127}} - 1\right)$$

Echelle des Mels

Multiplication du spectrogramme par des filtres triangles

En pratique réduit la dimensionnalité en fréquence.

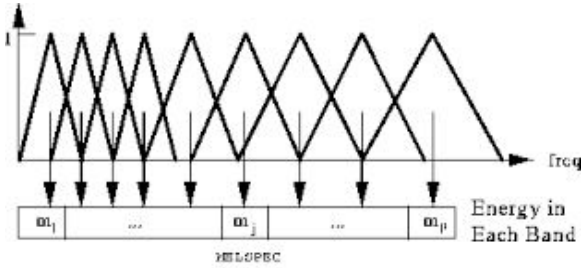


Fig. 5.3 Mel-Scale Filter Bank

```
n_mels = 128
fmin = 0.0
fmax = sr/2
n_fft = 2048
melfb = librosa.filters.mel(sr, n_fft, n_mels=n_mels, fmin=fmin, fmax=fmax)
mel_spec =  np.dot(melfb, spec)
librosa.display.specshow(librosa.amplitude_to_db(mel_spec, ref=np.max),
                         y_axis='log', x_axis='time')
```

Descripteurs audio

Énormément de descripteurs qui décrivent différentes propriétés audio.

Ici on présente deux :

1. MFCC qui représente le timbre
2. Chroma représente l'harmonie

Peuvent ensuite être utilisés comme entrée d'un algo de classification, ou pour calculer une matrice d'autosimilarité.

Descripteurs audio - MFCC

MFCC : Mel Frequency Cepstral Coefficients.

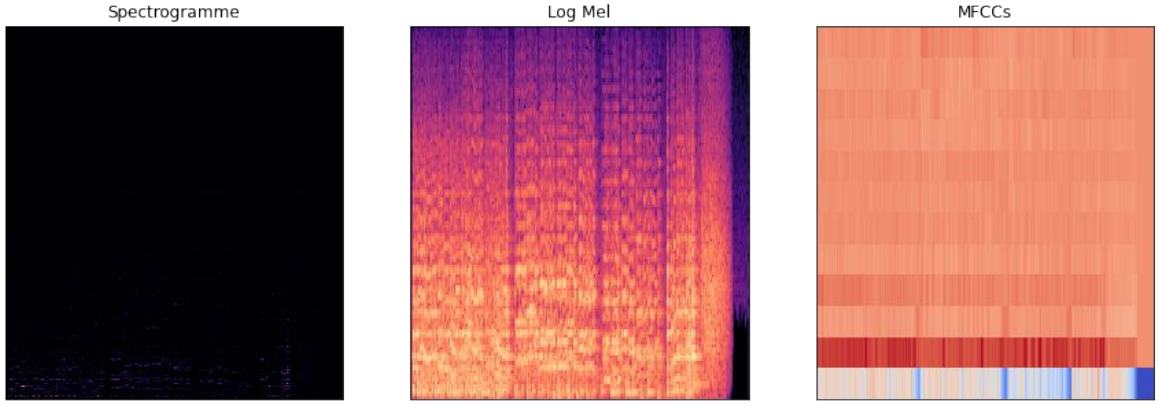
Très utilisé en parole et en musique. Capture le timbre en musique.

Calcul des MFCCs d'un signal s(t) :

1. Calcul de la STFT de s(t)
2. Filtrer par un banc de filtre de Mel pour obtenir un mel-spectrogramme
3. Appliquer un log (Mel-Log-Spectrogram, MLS)
4. Pour chaque trame du MLS on calcule la Transformée en cosinus discrète (similaire a la DFT mais avec des nombres réels uniquement).
5. Les coefficients MFCCs sont les amplitudes de ce spectrogramme.

On prend en général les coefficients d'ordre faible.

Descripteurs audio MFCC




```
filename = librosa.example('vibeace')
y, sr = librosa.load(filename)

librosa.feature.mfcc(y=y, sr=sr, dct_type=2)
```

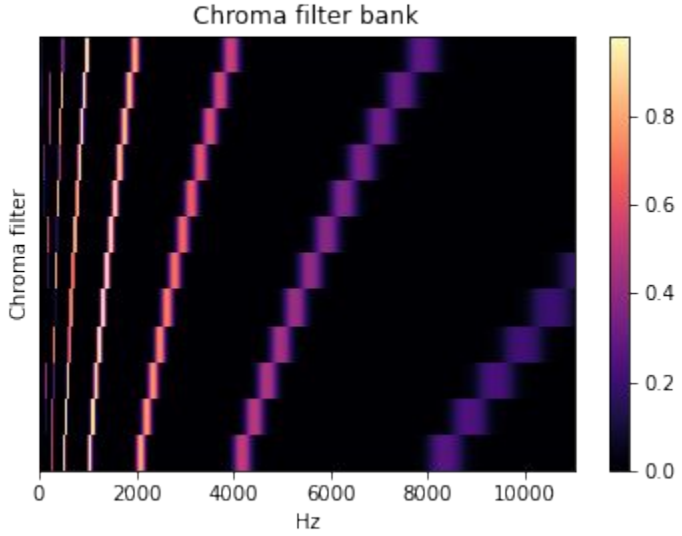
Descripteurs audio - Chroma

Chroma: descripteurs très populaires en musique. Représente l'harmonie.

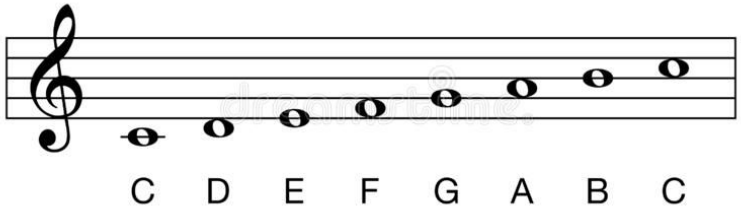
Décrit un signal en le décomposant suivant les 12 hauteurs de notes (C -> B)

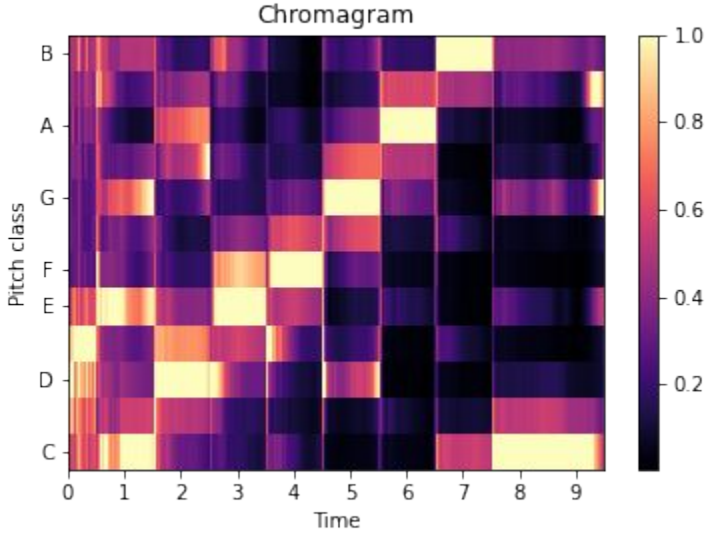
Calcul (plusieurs possibles):

1. Calculer le spectrogramme de s(t)
2. Calculer un filtre de chroma : Pour chaque hauteur/note, on récupère toutes les fréquences associées.
3. Filtrer le spectrogramme par le filtre obtenu



Descripteurs audio - Chroma






```
y, sr = librosa.load("c_scale.wav", duration=15)
chroma = librosa.feature.chroma_stft(y=y, sr=sr)
```

Descripteurs audio - SSM

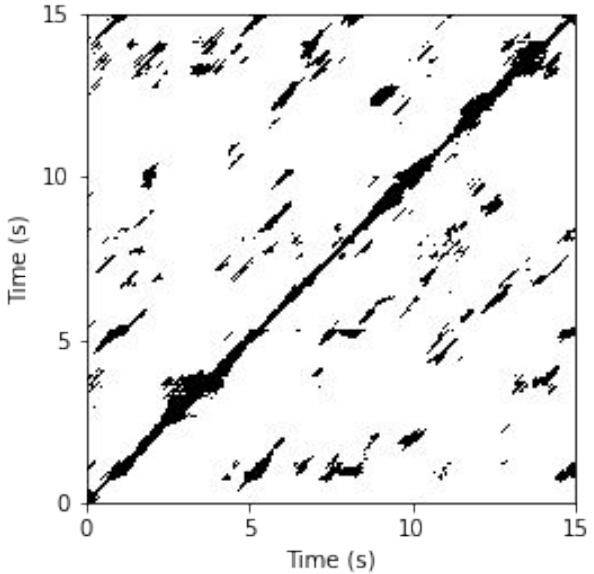
SSM : Self Similarity Matrix

Mettre en lumière les passages similaires dans une piste (utilisées pour l'estimation de structure musicale notamment).

$$S(j,k) = s(v_j, v_k) \quad j,k \in (1,\ldots,n)$$

Mesure de similarité

Vecteur de descripteurs

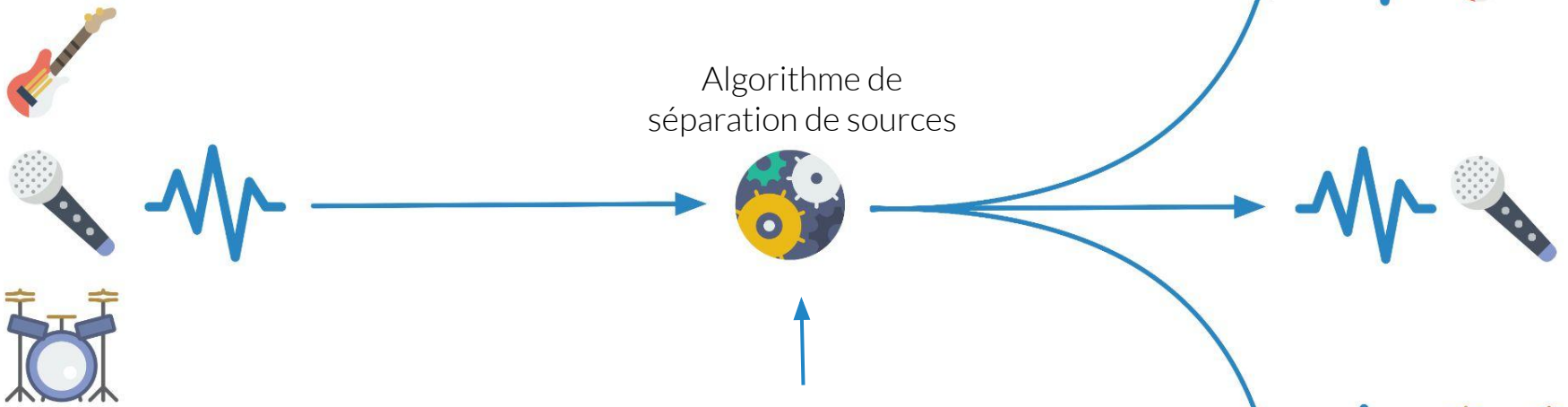


Séparation de sources audio

Mix
Contient tous
les instruments

Algorithme de
séparation de sources

Sources séparées
(Stem)



Algorithme de
séparation de sources

Réseaux de neurones convolutifs

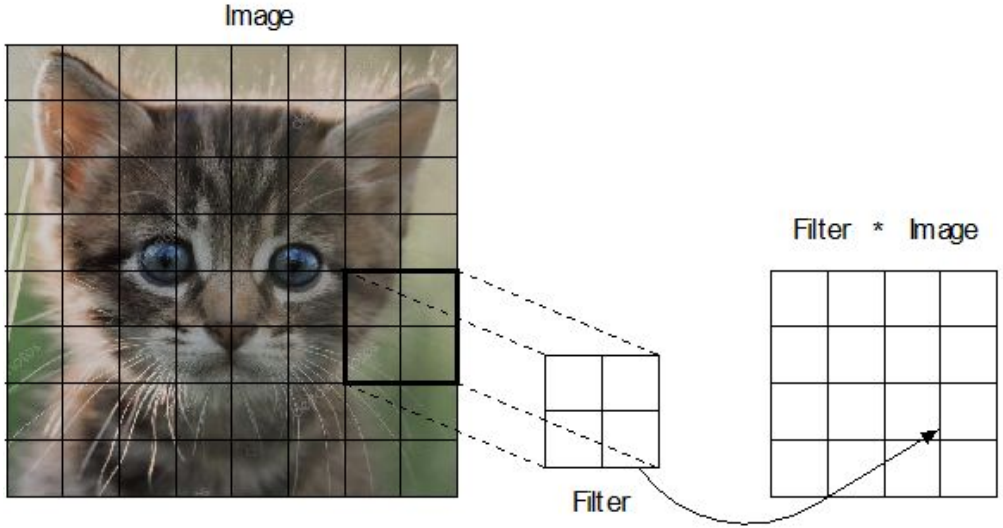
Mix
Contient tous
les instruments

Sources séparées
(Stem)

Convolutional Neural Networks (ConvNet) - 1

Pour l'analyse d'images à la base.

L'opération de base dans les ConvNet est la convolution de filtres sur l'image d'entrée

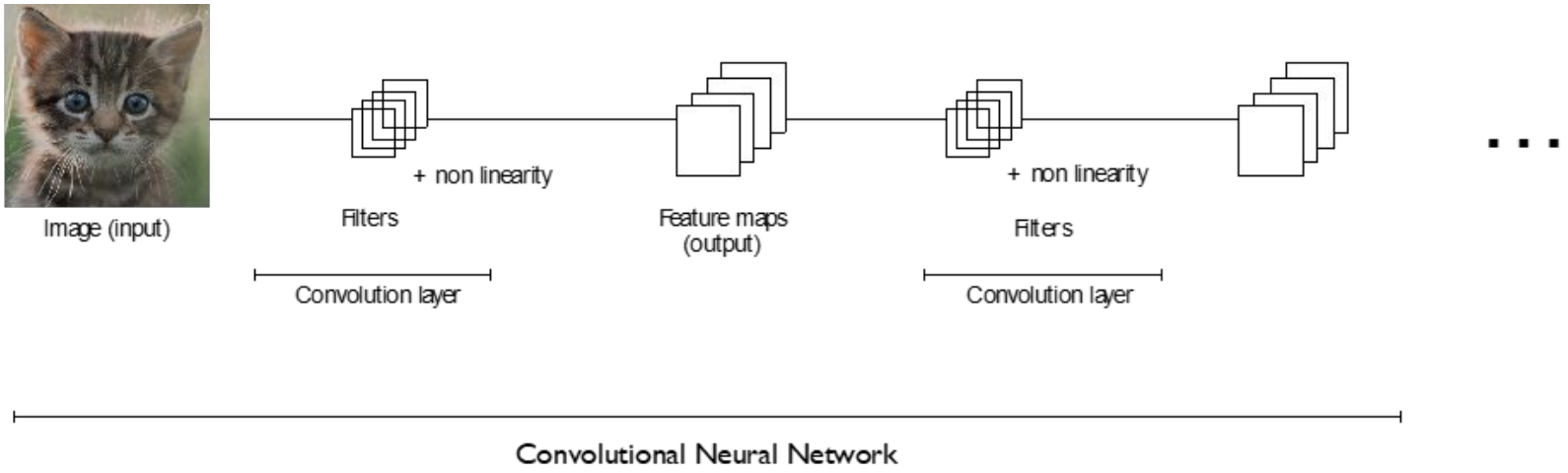


Un filtre représente un type de motif (une ligne horizontale par exemple).

Un filtre s'active dans toute l'image quelque soit l'emplacement du motif

Convolutional Neural Networks (ConvNet) - II

Pour former un réseau, on connecte les sorties d'une couche N-1 aux entrées d'une couche N.

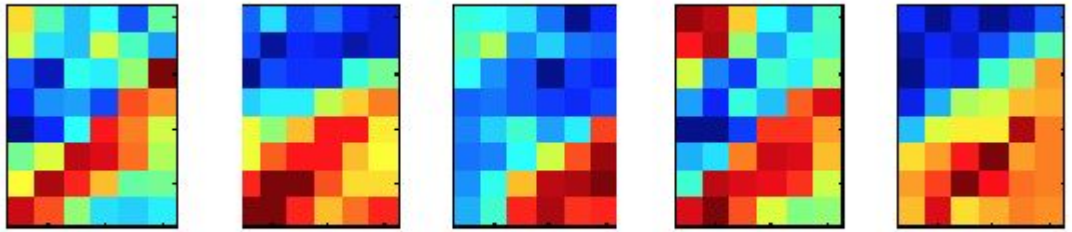


Appliquer les ConvNets aux sons

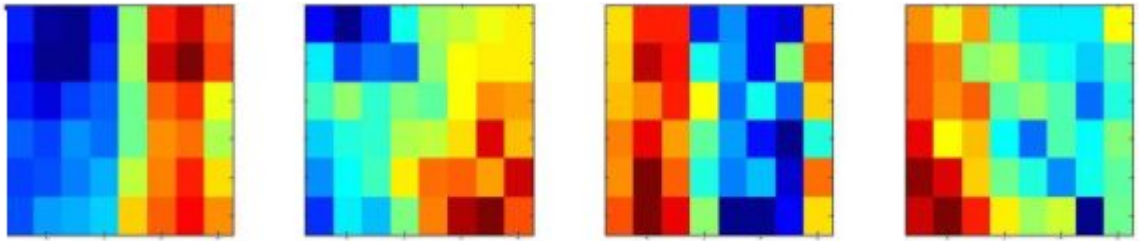
Deux choix (que l'on va voir avec la séparation) :

1. On utilise une représentation 2D (spectrogramme, SSM …)

SSM



Spectrogramme



Appliquer les ConvNets aux sons

Deux choix (que l'on va voir avec la séparation) :

1. On utilise une représentation 2D

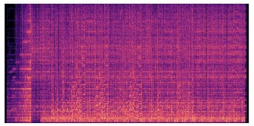
2. On utilise directement la forme d'onde, des filtres 1D et des convolutions 1D
   - Représentations moins informées -> besoin de plus de paramètres dans le réseau
   - Pour certaines tâches, a été montré que les première couches d'un ConvNet recontruisent une représentation temps fréquences.

Pour la séparation de sources

Deux modèles semblables:

1. U-net
   - Entrée : Spectrogramme
   - Article : Jansson, Andreas et al. “Singing Voice Separation with Deep U-Net Convolutional Networks.” ISMIR (2017).
2. Wave-U-Net
   - Entrée : Forme d’onde (end-to-end)
   - Article : Stoller, Daniel et al. “Wave-U-Net: A Multi-Scale Neural Network for End-to-End Audio Source Separation.” ISMIR (2018).

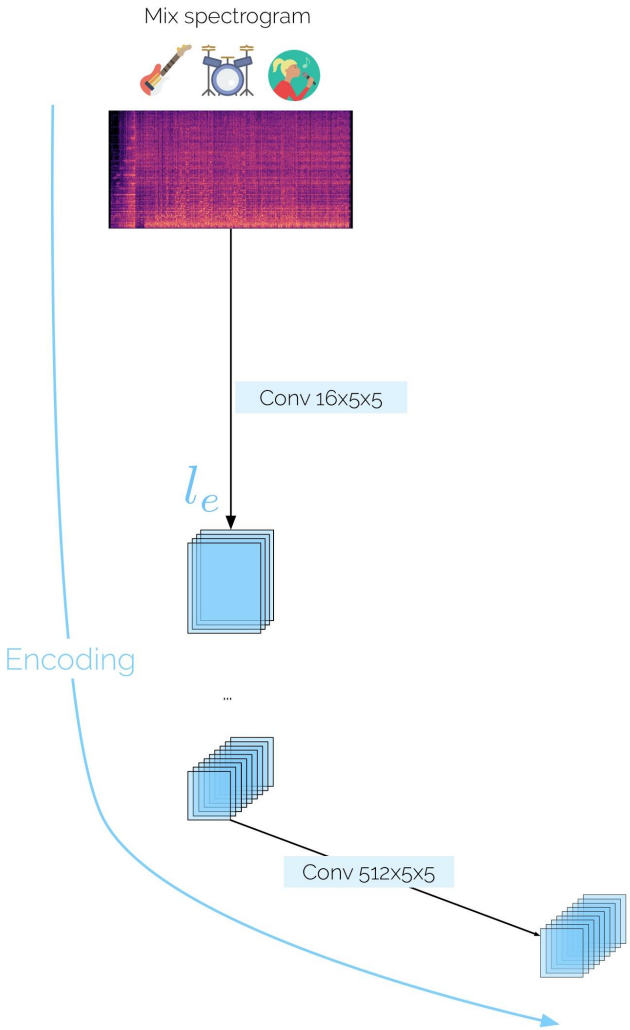
Mix spectrogram



U-Net

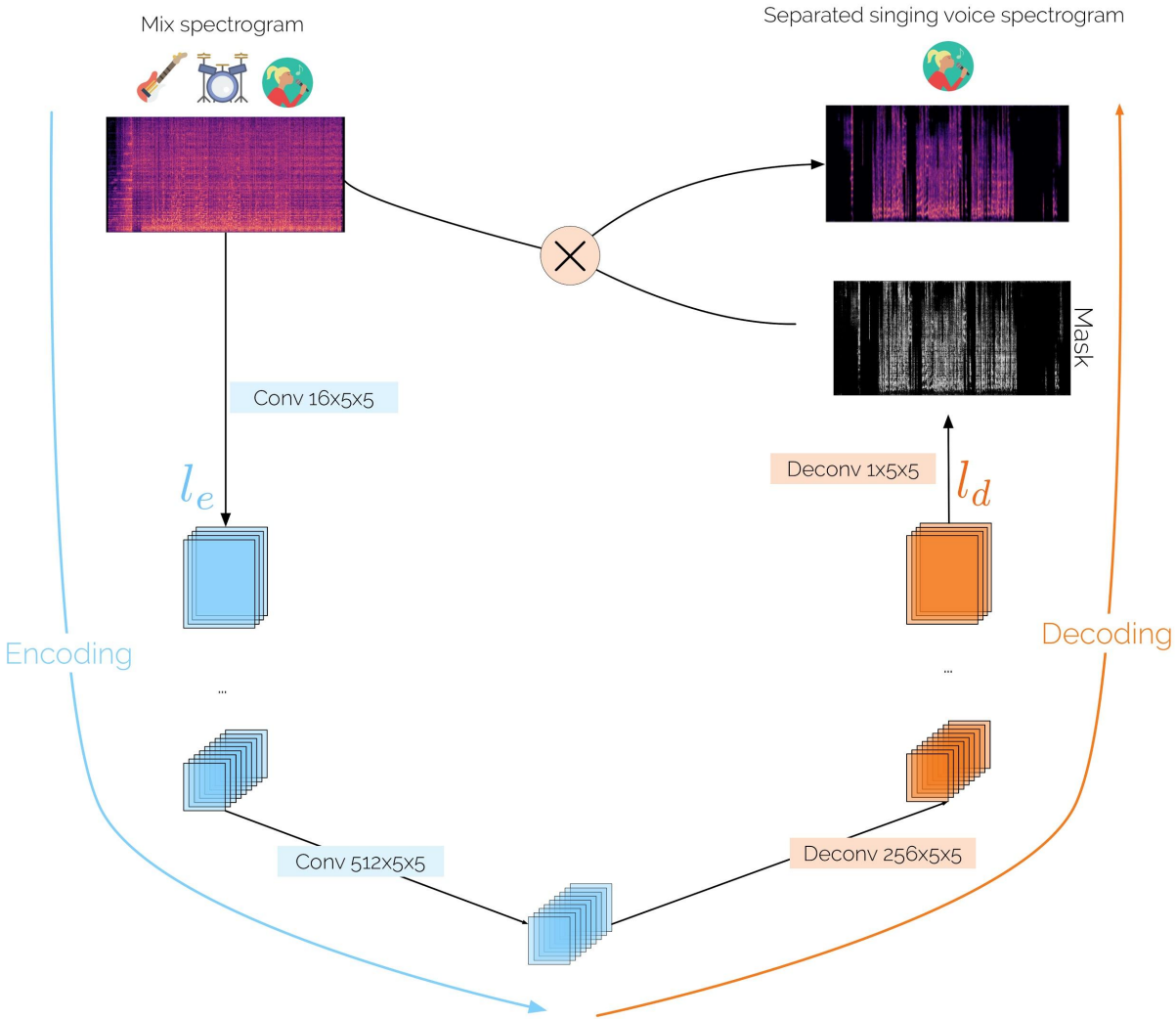
U-Net

Encoding-decoding scheme



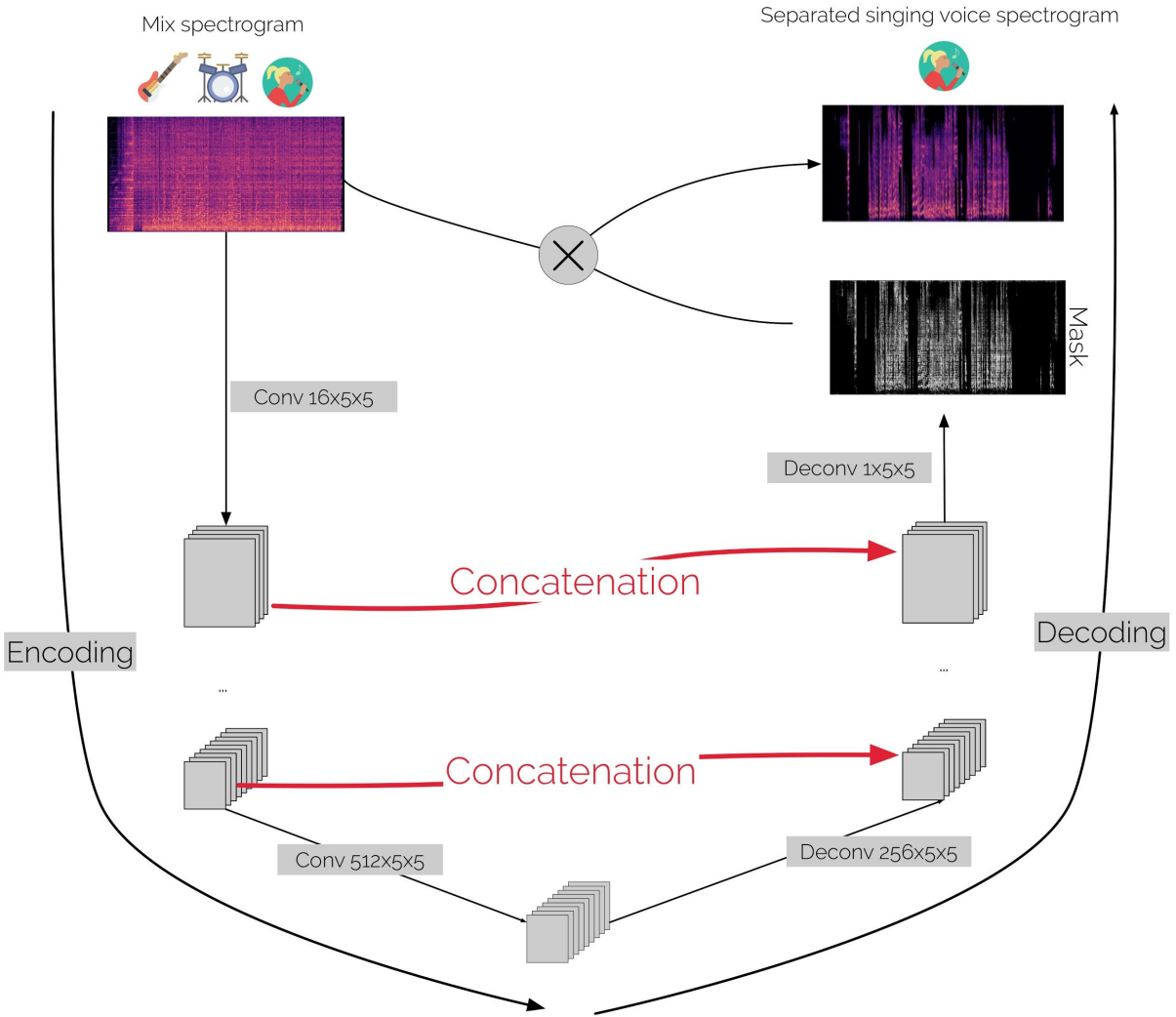
U-Net

Encoding-decoding scheme

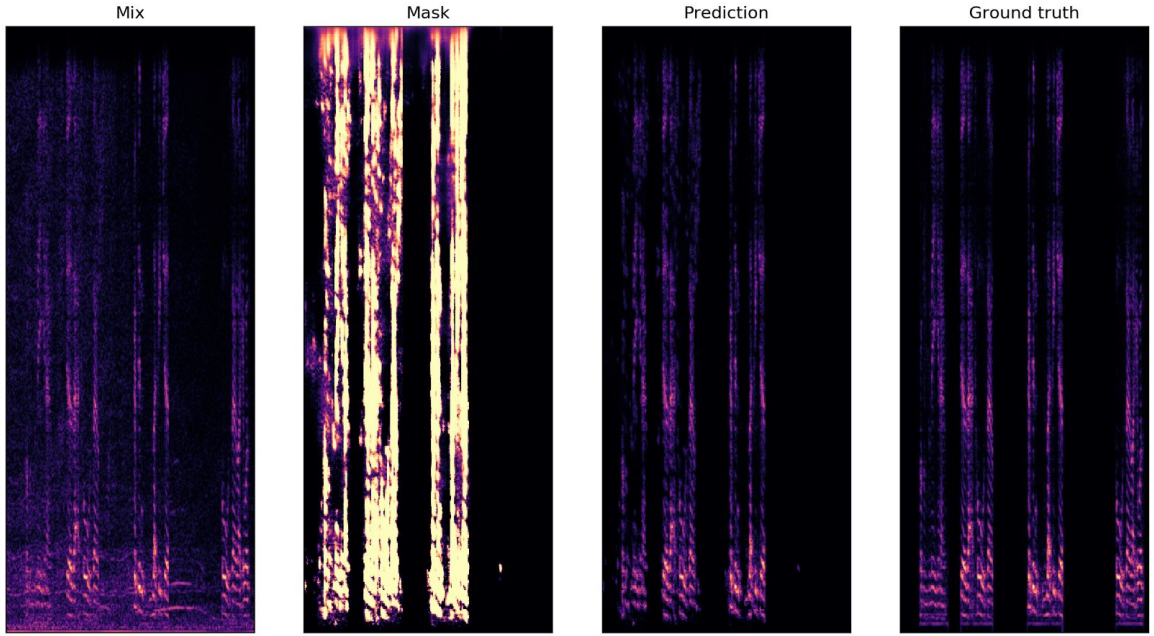


U-Net

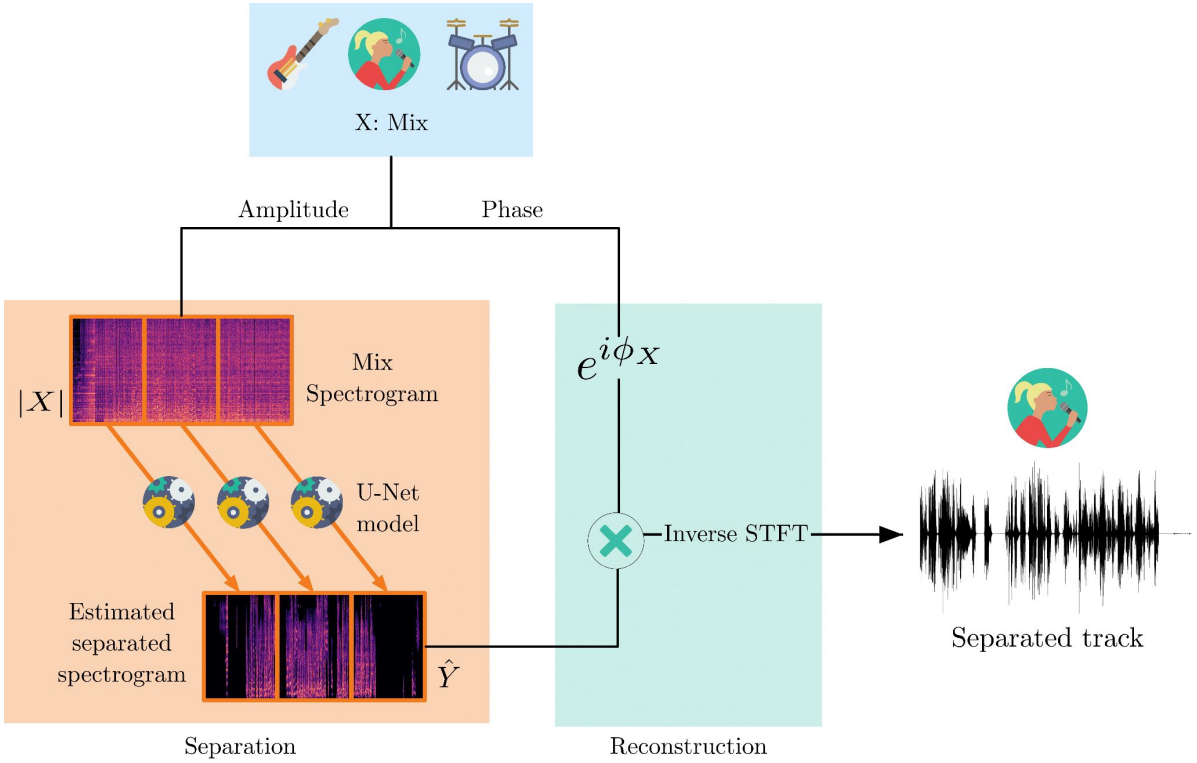
Skip Connections



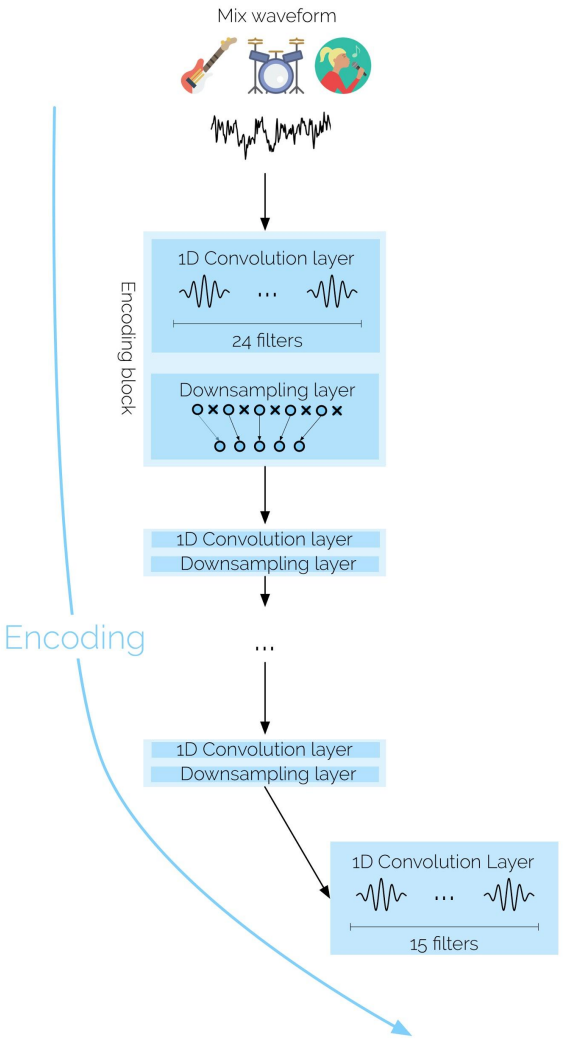
Exemples -Spectrogrammes et masque tf



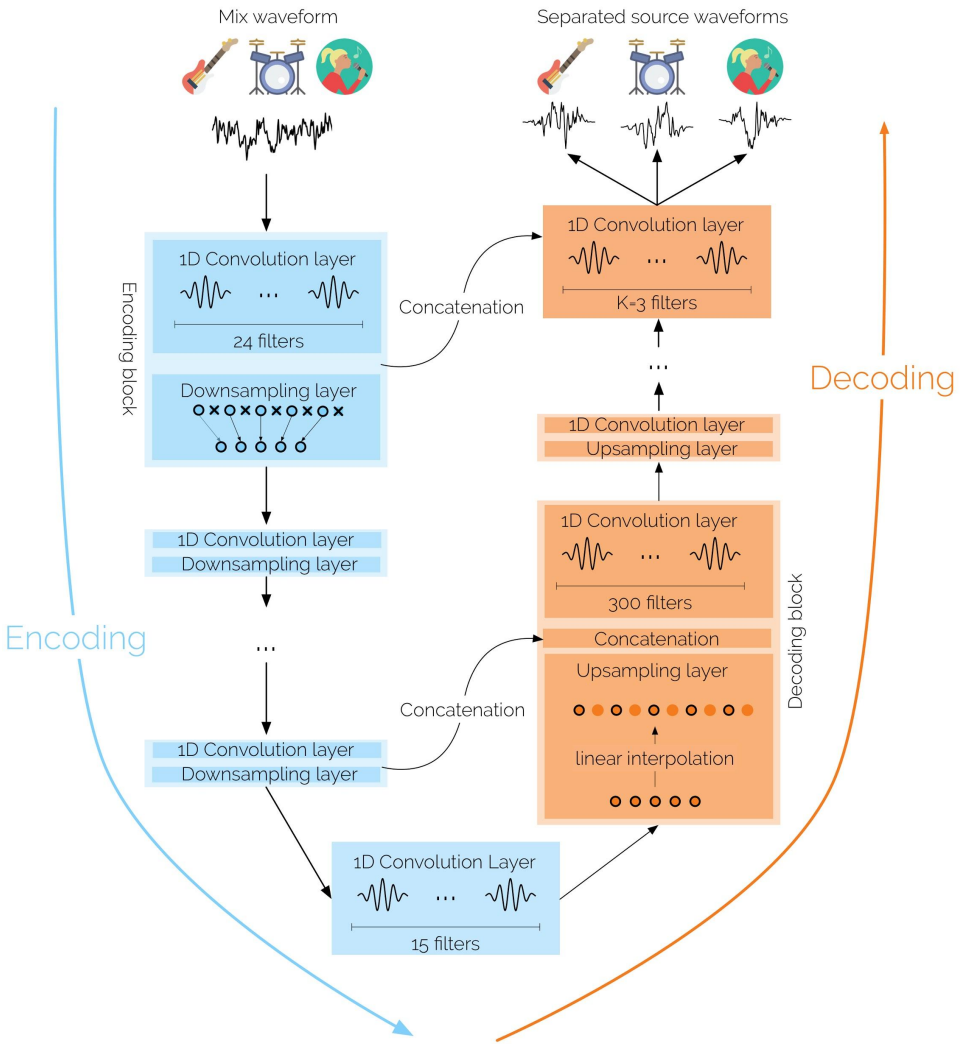
Reconstruction d'un signal temporel



Wave-U-Net


Mix waveform
Encoding
Encoding block
1D Convolution layer
24 filters
Downsampling layer
1D Convolution layer
Downsampling layer
...
1D Convolution layer
Downsampling layer
1D Convolution Layer
15 filters

Wave-U-Net


Mix waveform
Separated source waveforms
1D Convolution layer
24 filters
Encoding block
Downsampling layer
1D Convolution layer
Downsampling layer
1D Convolution layer
Downsampling layer
1D Convolution Layer
15 filters
Concatenation
Concatenation
1D Convolution layer
K=3 filters
1D Convolution layer
Upsampling layer
1D Convolution layer
300 filters
Concatenation
Upsampling layer
linear interpolation
Decoding block
Encoding
Decoding

Evaluer la séparation - Métriques usuelles

- SDR (Source to Distortion Ratio) : mesure la qualité globale de la séparation en évaluant la **distorsion dans la source reconstruite** par rapport à la source d'origine. Un SDR élevé indique une bonne fidélité de séparation.
- SIR (Source to Interference Ratio) : quantifie la **capacité à isoler une source des autre**s, en mesurant le niveau de suppression des interférences. Un SIR élevé indique que les autres sources ont été efficacement atténuées.
- SAR (Source to Artifacts Ratio) : évalue la **quantité d'artéfacts introduits** pendant le processus de séparation. Un SAR élevé indique que peu d'artéfacts ont été ajoutés, ce qui améliore la qualité perçue du signal séparé.

Evaluer la séparation

- Limites de SDR, SIR, SAR : ne captent pas toutes les nuances perceptuelles (distorsions subtiles, artefacts audibles).

- Métriques complémentaires : PESQ (Perceptual Evaluation of Speech Quality) ou STOI pour une évaluation plus précise de la qualité et de l'intelligibilité.

- Importance de l'évaluation humaine : juger la qualité sonore finale et valider la satisfaction auditive.

Exemples audio

 Money for nothing Mix

 Hammer to fall Mix

 Spice girls Mix

 Money for nothing Voice

 Hammer to fall Voice

 Spice girls Voice

Estimation de F0 et cover detection

Estimation de F0 - Définition

Représentation temps-fréquence qui visent à mesurer la saillance des fréquences au fil du temps.

Les sons que les humains perçoivent comme ayant une hauteur ont une structure harmonique.

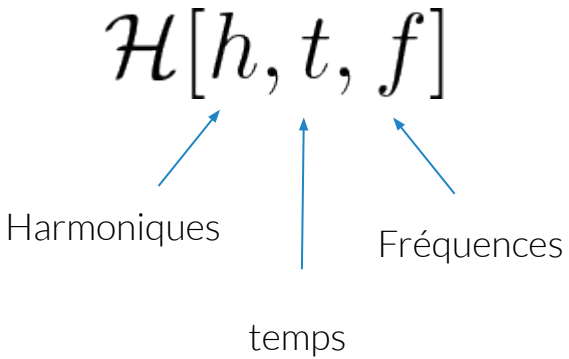
La fonction de saillance idéale est nulle partout où il n'y a pas de hauteur perceptible, et une valeur positive qui reflète le volume perçu des hauteurs à la fréquence fondamentale.

Estimation de F0 - HCQT

Deep Salience Representations For F0 Estimation In Polyphonic Music, Bittner, ISMIR 2017

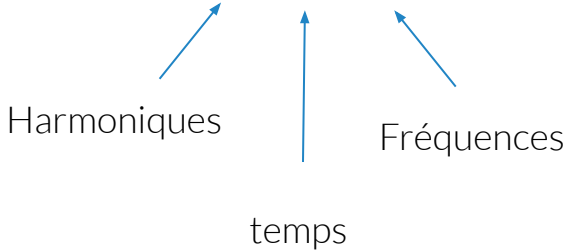
Code : https://github.com/rabitt/ismir2017-deepsalience

Proposition d'une représentation 3 dimensions (stack de représentation temps fréquence CQT) CQT Harmonique:



Estimation de F0 - HCQT

$$\mathcal{H}[h, t, f]$$



Pour chaque harmonique h, $\mathcal{H}(h)$ est une CQT avec comme fréquence minimum $h \cdot fmin$ avec $h \in \{0.5, 1, 2, 3, 4, 5\}$ et $fmin = 32.7$ Hz

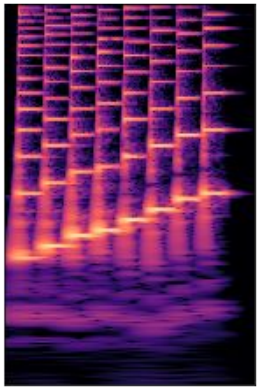
Estimation de F0 - HCQT code

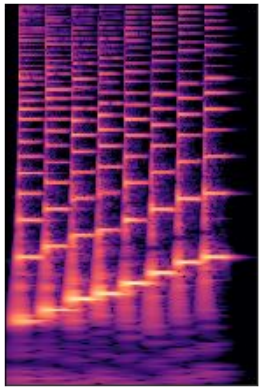
```
BINS_PER_OCTAVE = 60
N_OCTAVES = 6
HARMONICS = [0.5, 1, 2, 3, 4, 5]
SR = 22050
FMIN = 32.7
HOP_LENGTH = 256

y, sr = librosa.load("c_scale.wav", sr= SR)
cqt_list = []
shapes = []

for h in HARMONICS:
    cqt = librosa.cqt(
            y, sr=sr, hop_length=HOP_LENGTH, fmin=FMIN*float(h),
            n_bins=BINS_PER_OCTAVE*N_OCTAVES,
            bins_per_octave=BINS_PER_OCTAVE
        )
    cqt_list.append(cqt)
    shapes.append(cqt.shape)
```

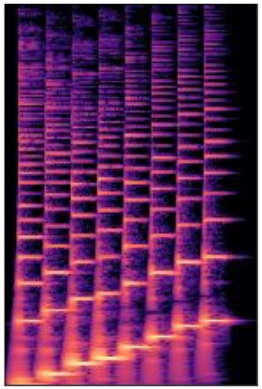
$$\mathcal{H}[h, t, f]$$

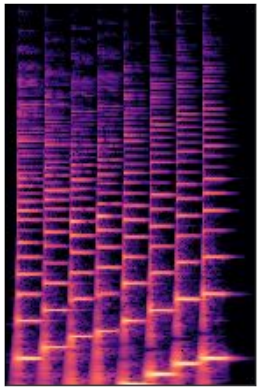
Estimation de F0 - HCQT

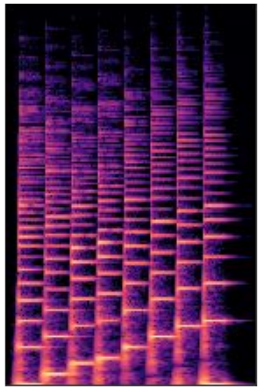
Exemple de la gamme de Do

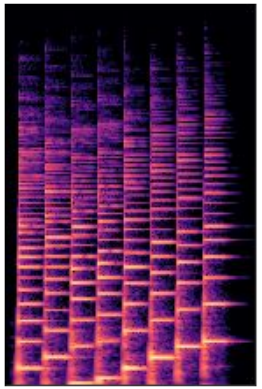

Harmonique 0.5


Harmonique 1


Harmonique 2

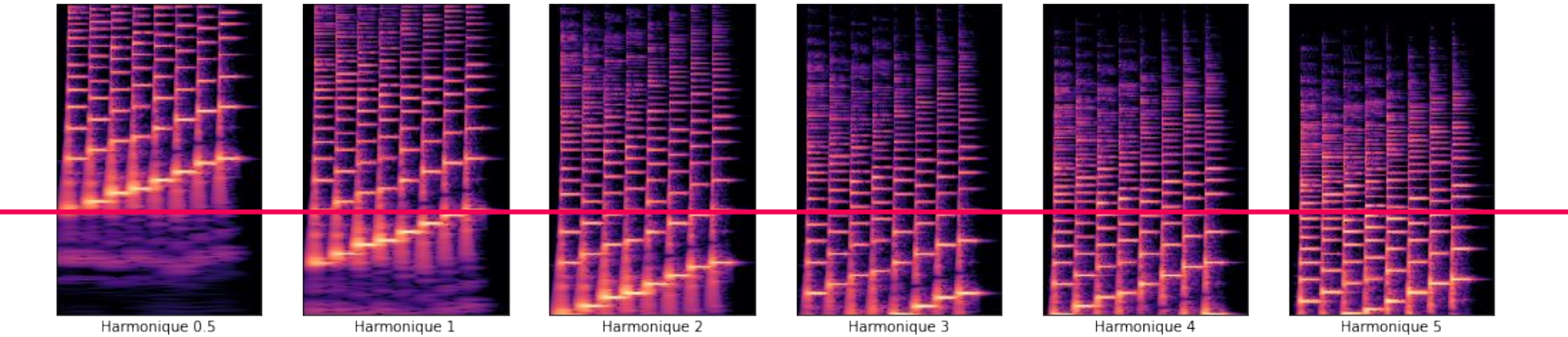

Harmonique 3


Harmonique 4


Harmonique 5

Estimation de F0 - HCQT

Exemple de la gamme de Do: les harmoniques d'une note s'alignent



Estimation de F0 - ConvNet

On utilise la HCQT en entrée d'un réseau de neurones convolutif :

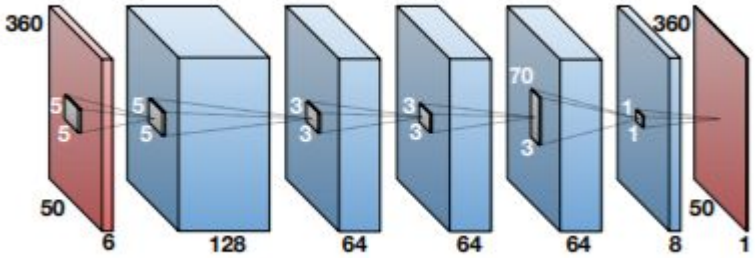
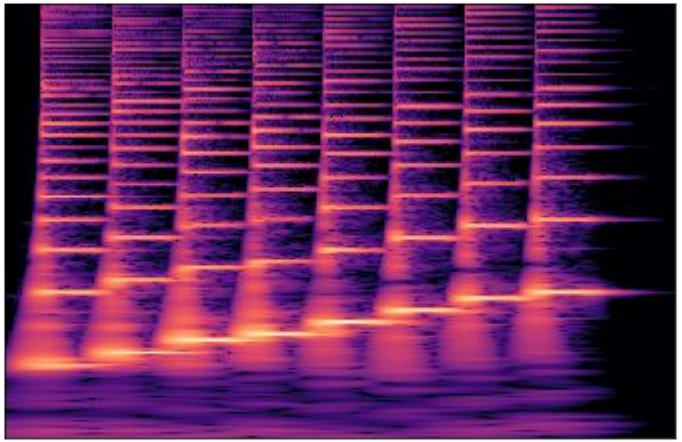


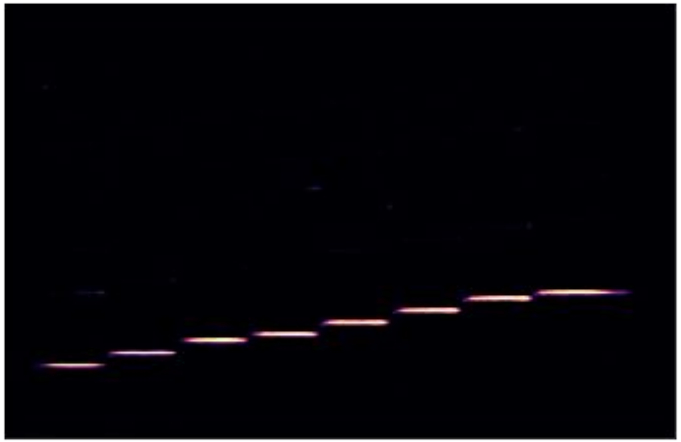
Figure tirée iré de l'article Bittner, ISMIR 2017

Entrainement avec les bases MedleyDB, évaluation avec Bach10 et Su datasets.

Exemples - Résultat gamme de do

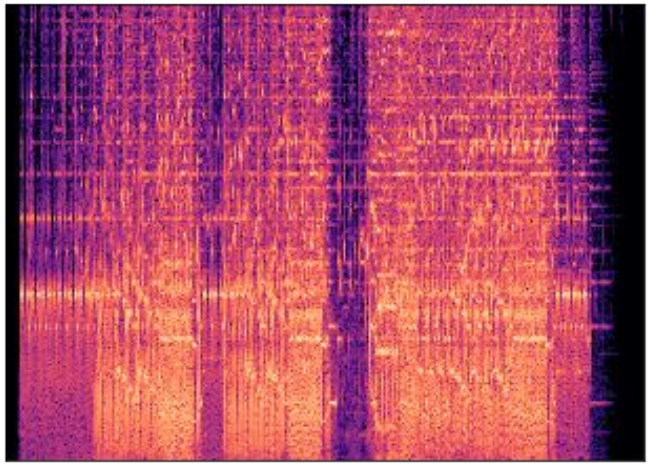


Une cqt de la HCQT



Représentation saillante

Exemples - Résultat musique



Une cqt de la HCQT

Représentation saillante

Cover detection

Mal défini : qu'est-ce qu'une reprise/cover ?

On pose des hypothèses de départ.

Même mélodie ? Originale Cover

Même paroles ? Originale Cover

"Feeling" semblables ? Originale Cover

(procès perdu par Georges Harisson)

Dépend du use case final. Ici : méthode qui se base sur la mélodie et donc la F0.

Cover detection avec représentation F0

Cover Detection Using Dominant Melody Embeddings , Doras, ISMIR 2019

Utiliser la représentation saillante F0 pour détecter des cover.

On veut comparer une oeuvre à toutes les oeuvres de notre dataset.

Besoin d'une représentation compacte: **embeddings.**

L'idée est de construire un nouvel espace où la distance euclidienne représente la distance entre deux oeuvres.

Cover detection - Embeddings

Représentation saillante F0

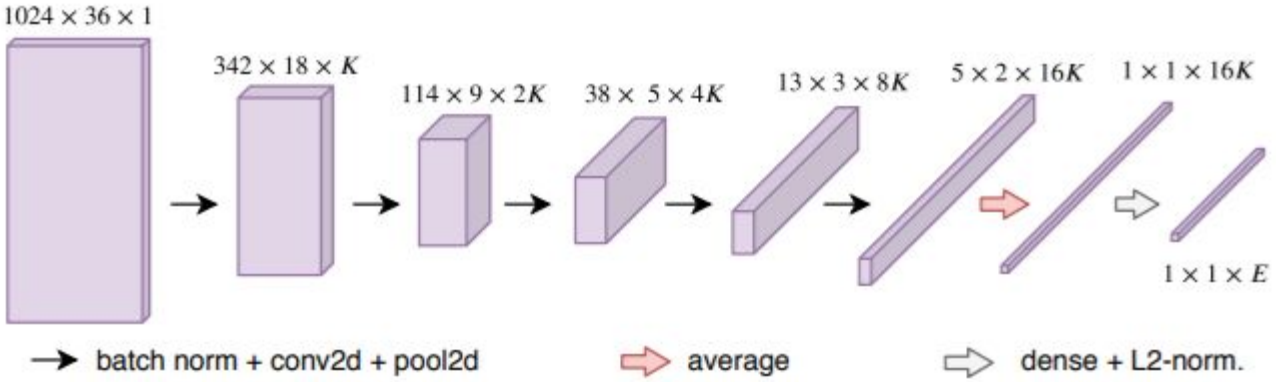


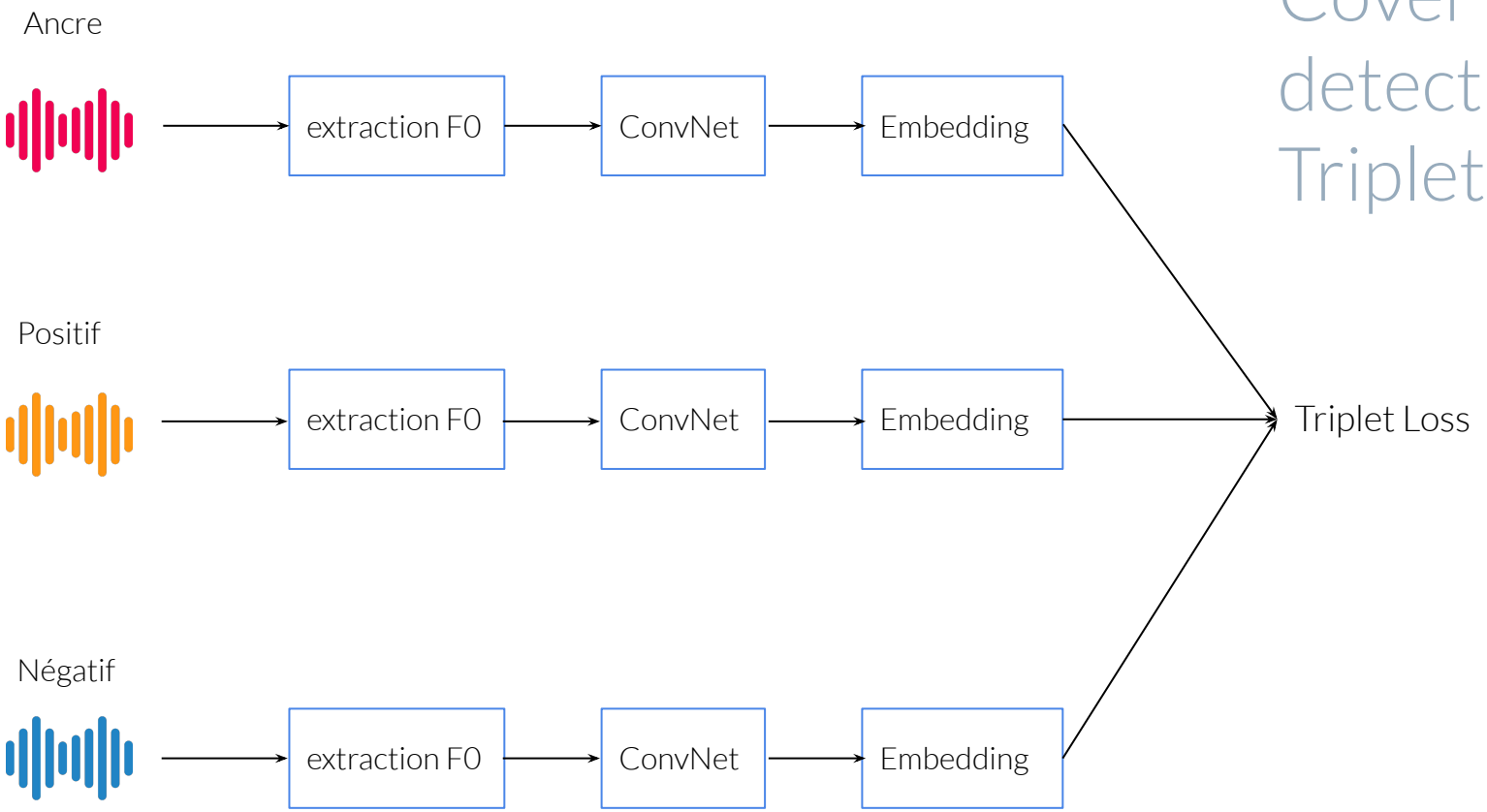
Figure 1: Convolutional model (time on the first dimension, frequency on the second dimension).

Embeddings

Figure tirée iré de l'article Doras, ISMIR 2019

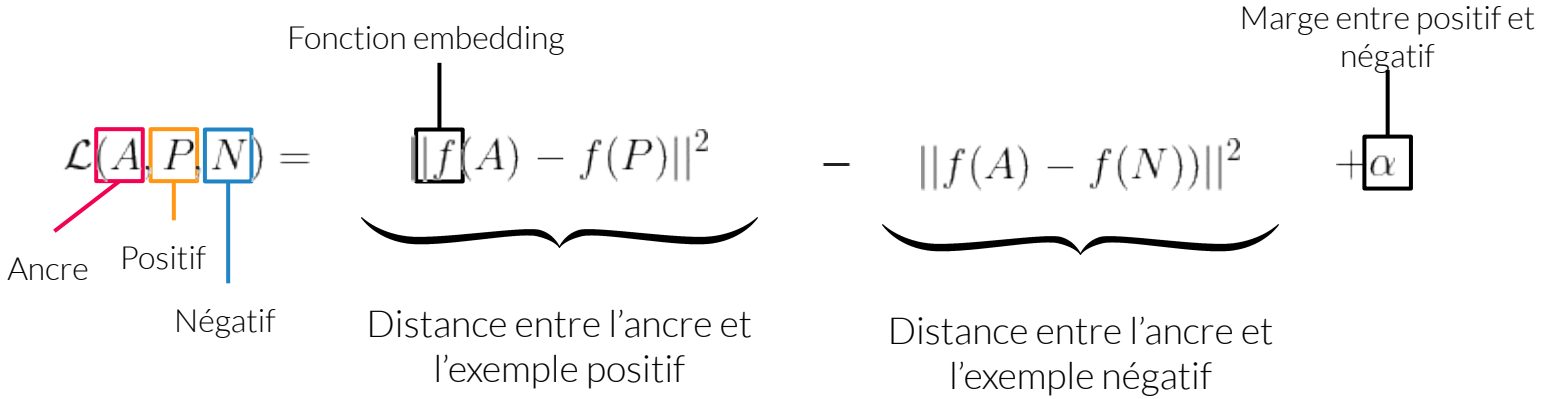
Le réseaux de neurones convolutif apprend un nouvel espace où projeter les données de F0. La distance euclidienne a une signification sémantique grâce a la triplet loss.

Cover detection - Triplet Loss



Cover detection - Triplet Loss

Pour un triplet d'exemples :

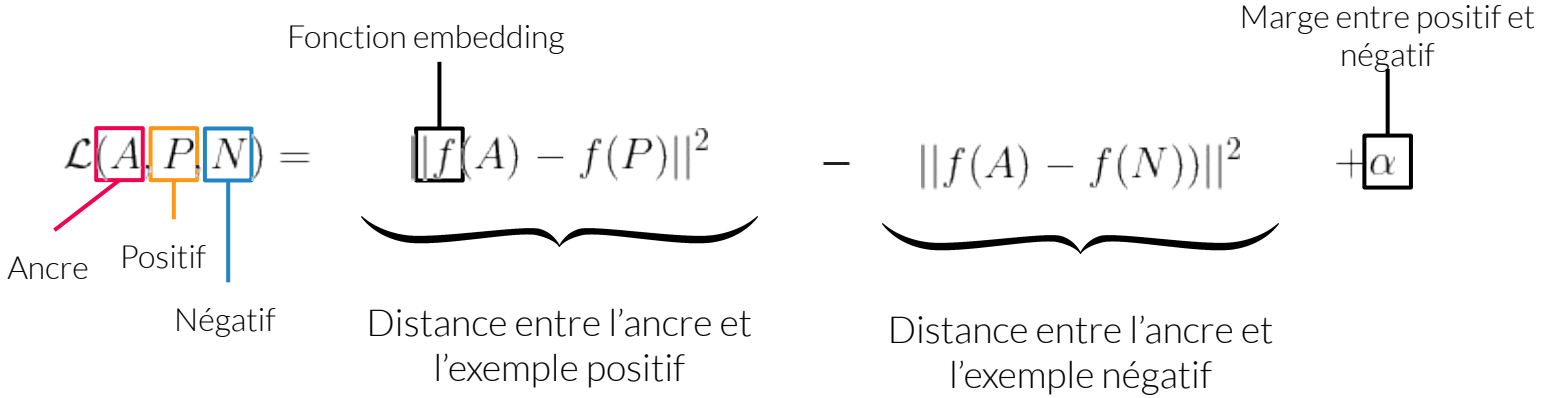


En minimisant cette fonction de coût :

- On minimise le premier terme → on minimise la distance entre l'ancre et le positif
- On maximise le second terme → on maximise la distance entre l'ancre et le négatif

Cover detection - Triplet Loss

Pour un triplet d'exemples :



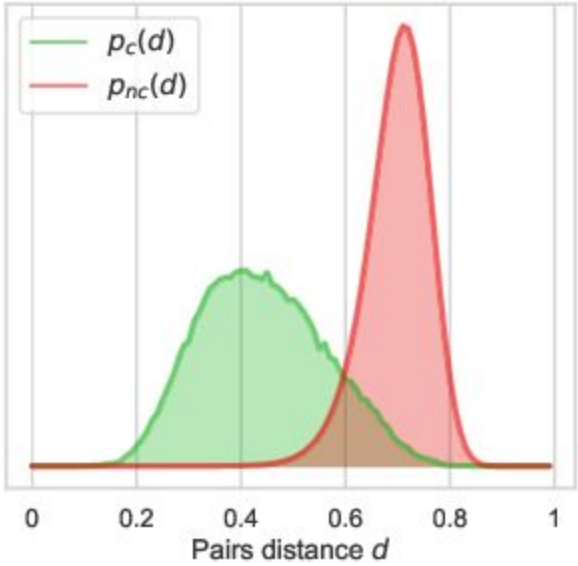
Sur toute la base de données : $\mathcal{J} = \sum_{i=1}^{M} \mathcal{L}\left(A^{(i)}, P^{(i)}, N^{(i)}\right)$

Résultats

On calcule toutes les distances entre les oeuvres originales et les covers et les non covers de notre base.

$p_c(d)$ distance entre covers

$p_{nc}(d)$ distance entre non covers



Bibliographie

- Jansson, Andreas et al. “Singing Voice Separation with Deep U-Net Convolutional Networks.” ISMIR (2017).
- Stoller, Daniel et al. “Wave-U-Net: A Multi-Scale Neural Network for End-to-End Audio Source Separation.” ISMIR (2018).

- Bittner, Rachel et al. “Deep Salience Representations For F0 Estimation In Polyphonic Music”, ISMIR 2017
- Doras, Guillaume, Peeters, Geoffroy, “Cover Detection Using Dominant Melody Embeddings” , ISMIR 2019