APP2

RECONNAISSANCE DE MUSIQUES

Rayane D
Sacha K

Adam C
Sacha M

Méthodes

Méthode pour identifier les notes:

- Appliquer au signal un filtre passe-bande sur les fréquences des notes recherchées
- Faire la TFTC du signal afin de récupérer le spectrogramme du signal
- Sectionner le spectrogramme pour identifier chaque note jouée à chaque instant
- Déterminer l’amplitude max sur chaque tranche et assigner à une note

Méthode pour identifier les instruments:

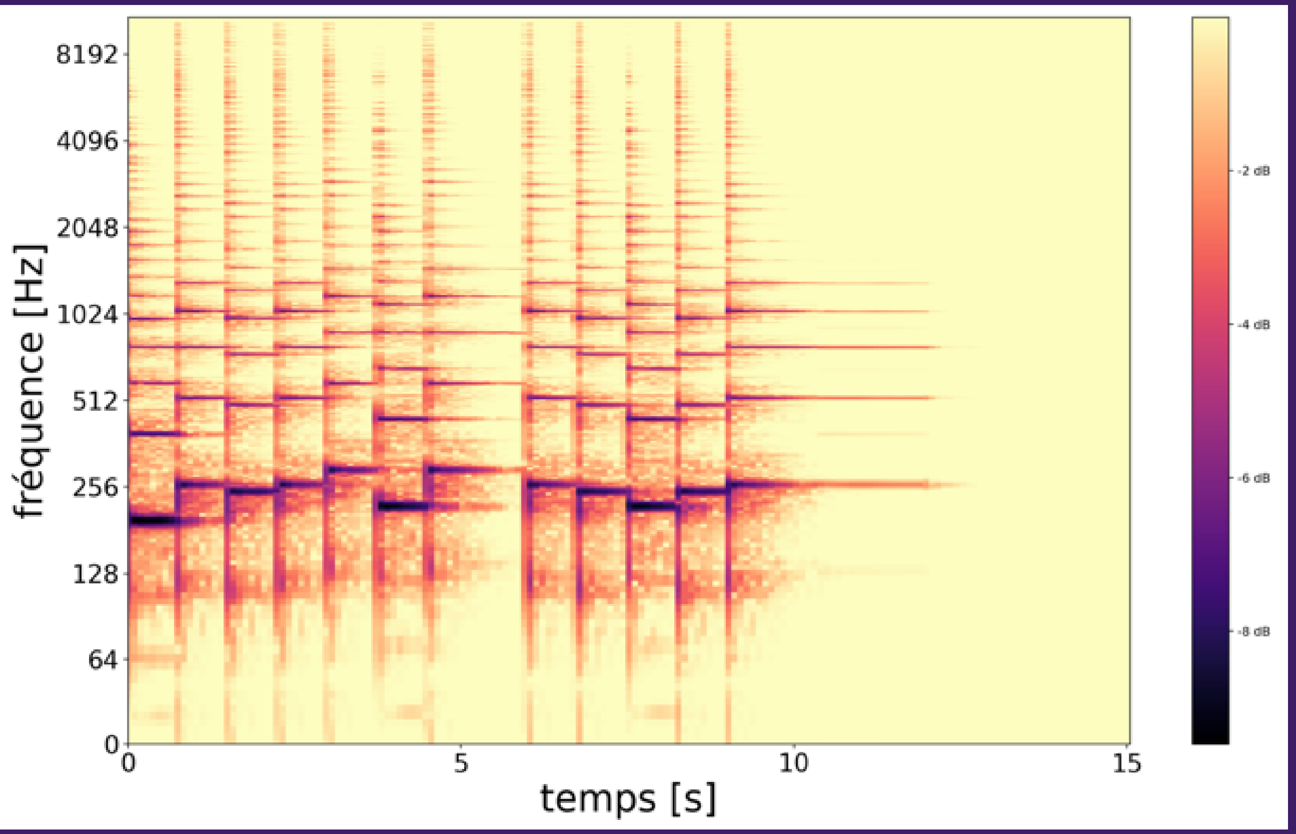
- Découper le signal en plusieurs segments
- Compter le nombre de fréquence ayant une énergie significative
- Si ce nombre dépasse un certain seuil, c’est une percussion
- Sinon c’est un Instrument à corde

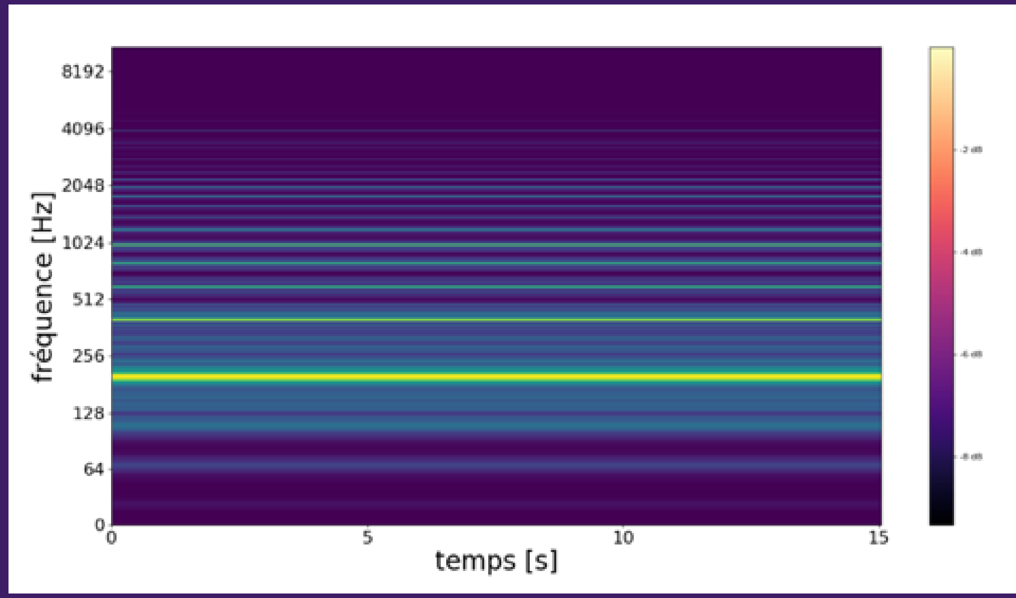
```python
#on renvoi un tableau qui donne les intervalles de changement de note
def instants_changements(spectrogram, y_ref, threshold, Fe, nperseg):
    row_index = int(y_ref * nperseg / Fe)
    row = spectrogram [row_index, :]
    diff = np.diff(row)
    change_points = np.where(diff > threshold) [0]
    t_max = len(row)/Fe/2*nperseg
    change_times = change_points /Fe * nperseg/2
    change_times = np.insert (change_times, 0 , 0)
    change_times = np.append(change_times, t_max)
    return change_times
```

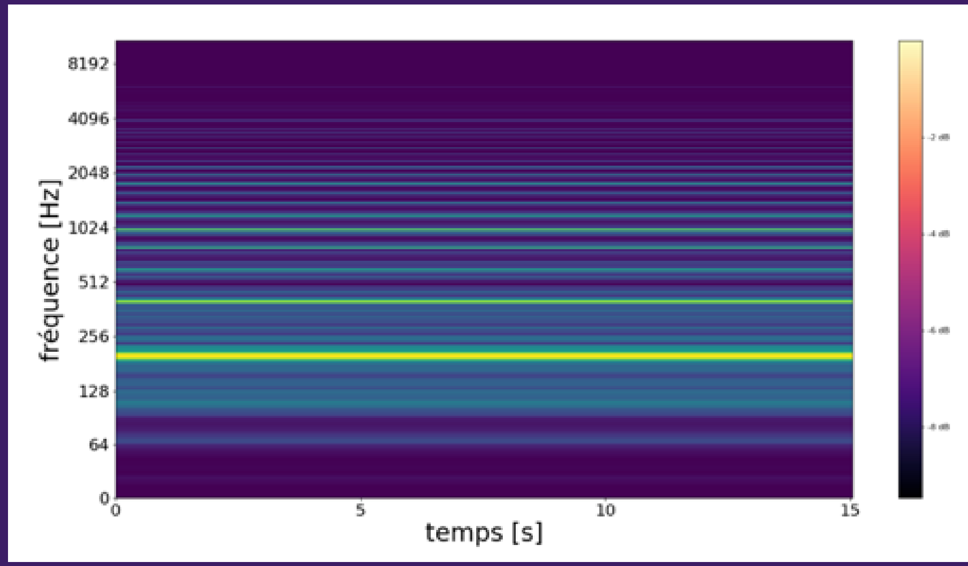
```python
def tableau_des_tranches(spectrogram, y_ref, threshold, Fe, nperseg):
    change_times = instants_changements(spectrogram, y_ref, threshold, Fe, nperseg)
    tranches = []
    tranches_f=[]
    for i in range(len(change_times)-1):
        debut = int(change_times[i])
        fin = int(change_times[i+1])
        tranches.append(spectrogram[:,debut:fin])
    for j in tranches:
        if j.size>0:
            tranches_f.append(j)
    return tranches_f
```

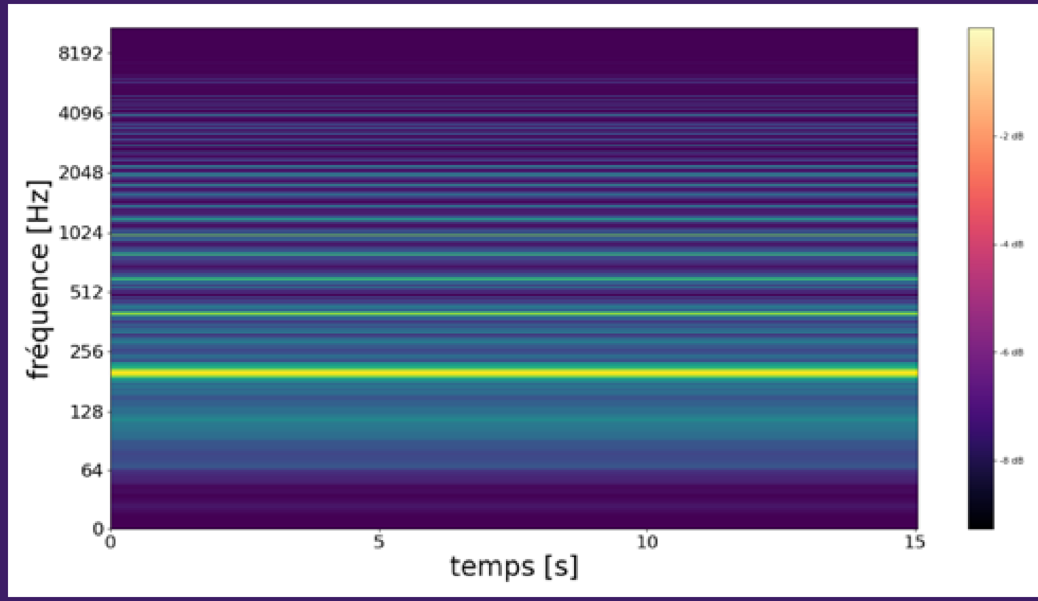
```python
def find_widest_peak(spectrogram):
    # Aplatir le spectrogramme en une seule dimension
    flattened_spectrogram = np.ravel(spectrogram)

    # Trouver les pics
    peaks, _ = find_peaks(flattened_spectrogram)

    # Calculer la largeur de chaque pic
    widths, _, _, _ = peak_widths(flattened_spectrogram, peaks)

    # Trouver l'indice du pic le plus large
    widest_peak_index = np.argmax(widths)

    # Retourner l'indice du pic le plus large et sa largeur
    return peaks[widest_peak_index], widths[widest_peak_index]
```

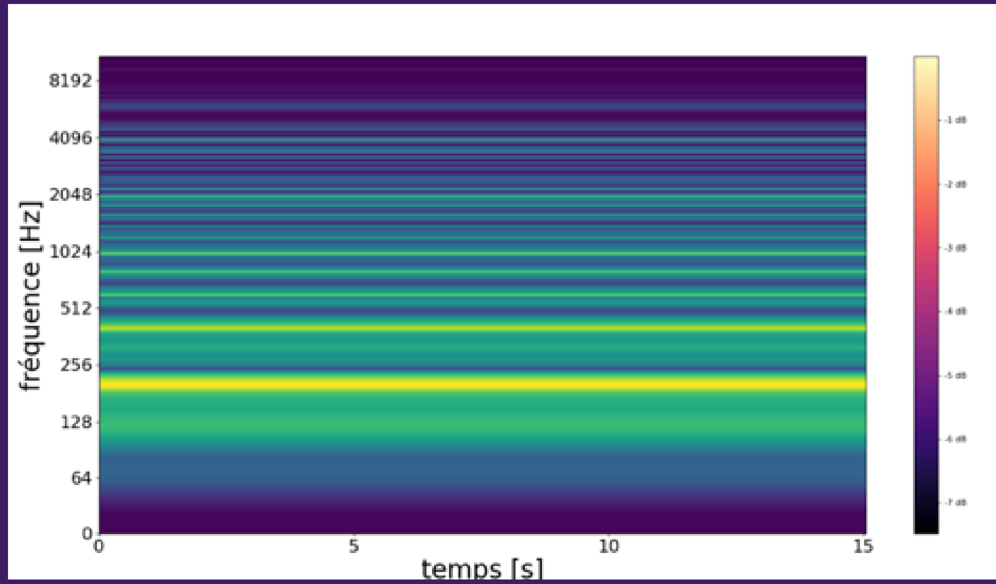
```python
def detect_instrument_in_segment(segment, sr):

    # Compter les fréquences significatives
    significant_freqs_count = count_significant_frequencies(segment, sr)

    # Estimer si l'instrument est à cordes ou une percussion basé sur le
    if significant_freqs_count < 7500:  # Seuil à ajuster selon les besoi
        instrument_type = "Cordes"
    else:
        instrument_type = "Percussion"

    return instrument_type
```


8192
4096
2048
1024
512
256
128
64
0
fréquence [Hz]
0
5
10
15
temps [s]
-2 dB
-4 dB
-6 dB
-8 dB


8192
4096
2048
1024
512
256
128
64
0
fréquence [Hz]
0
5
10
15
temps [s]


8192
4096
2048
1024
512
256
128
64
0
fréquence [Hz]
0
5
10
15
temps [s]


8192
4096
2048
1024
512
256
128
64
0
fréquence [Hz]
0
5
10
15
temps [s]


8192
4096
2048
1024
512
256
128
64
0
fréquence [Hz]
0
5
10
15
temps [s]

Conclusion et Perspectives d’amélioration