Homework Assignment No. 2 - Timbre Transfer

Flavio De Lellis

January 12, 2025

Contents

1 Introduction

The timbre of a sound is a complex auditory attribute that distinguishes different sounds having the same pitch and loudness. It encompasses various physical properties of the sound, such as its spectrum, envelope, and harmonic content, which contribute to its unique identity. Timbre transfer is the process of altering the timbre of a target audio signal to make it resemble the timbre of a source signal while retaining the original pitch and rhythm of the target.

Modern techniques for timbre transfer often employ advanced neural networks, but in this assignment, the task is achieved using signal processing methods, specifically Nonnegative Matrix Factorization (NMF). By factorizing the magnitude spectrum of audio signals, NMF enables the extraction and manipulation of timbre-related features. The transfer is accomplished by fixing the source spectrum as a template and learning an activation matrix that adapts the source characteristics to the target.

2 Question 1

The first analysis in Question 1 aims to understand the **spectral content** of the source and target audio signals. Below, we describe the steps undertaken:

2.1 Sound Selection

The analysis begins by selecting specific audio files:

- **Source**: `Bees_Buzzing.mp3`, representing the timbre to be learned.
- **Target**: `Jingle_Bells_Boogie.wav`, representing the audio where the timbre will be applied.

2.2 Signal Loading

The waveforms of the source and target are loaded using the `librosa.load()` function:

- **Uniform Sampling Rate**: Signals are resampled to a common rate of $F_s = 22050$ Hz, ensuring consistency during processing.
- **Standardized Format**: Both signals are converted into numerical arrays for computational analysis.

2.3 Transition to Time-Frequency Domain

The time-domain waveforms are transformed into the time-frequency domain using the Short-Time Fourier Transform (STFT). This step is essential for capturing both the temporal and spectral properties of the audio.

Parameters for STFT:

- $N_{\text{length}} = 4096$
- $H_{\text{size}} = 1024$

We'll see further that these parameters are crucial in our scope as we'll also tune them.

2.4 Magnitude Spectrum Extraction and Plotting

The magnitude spectrum is extracted from the complex-valued STFT matrices:

- **Purpose**: Focuses on the energy distribution across frequencies, ignoring phase information. The magnitude spectrum serves as the key input for Nonnegative Matrix Factorization (NMF) in later steps.

Visual inspection is conducted by plotting the magnitude spectra:

- **Spectrograms**: Display the energy distribution in both time and frequency domains, using logarithmic scaling on the frequency axis for better representation of harmonic structures.
  We've used the function **librosa.display.specshow** to plot the STFT of source and target.

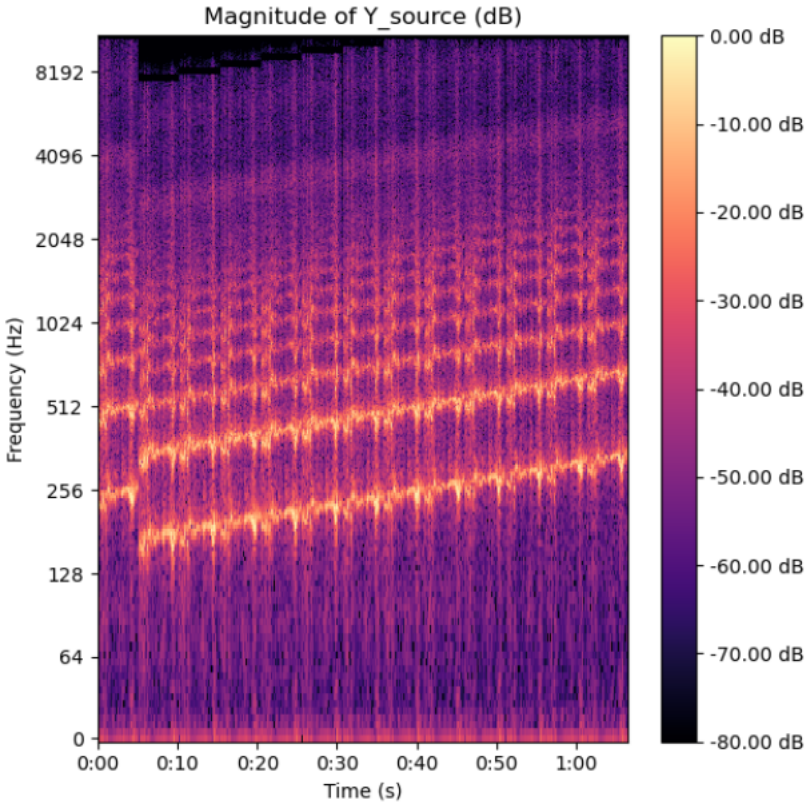

Figure 1: STFT of source **Beez Buzzing**

The source spectogram tells us that the energy appears to be distributed across a wide range of frequencies, with most of it concentrated below 2000

Hz and above 128 Hz. Higher harmonics fade out gradually, indicating that the signal has diminishing energy in higher frequency ranges.
The Bees Buzzing sound has a rich, harmonic spectrum with subtle noise components. This makes it suitable as a timbre source, as its distinct frequency structure can add a unique texture to the target signal.

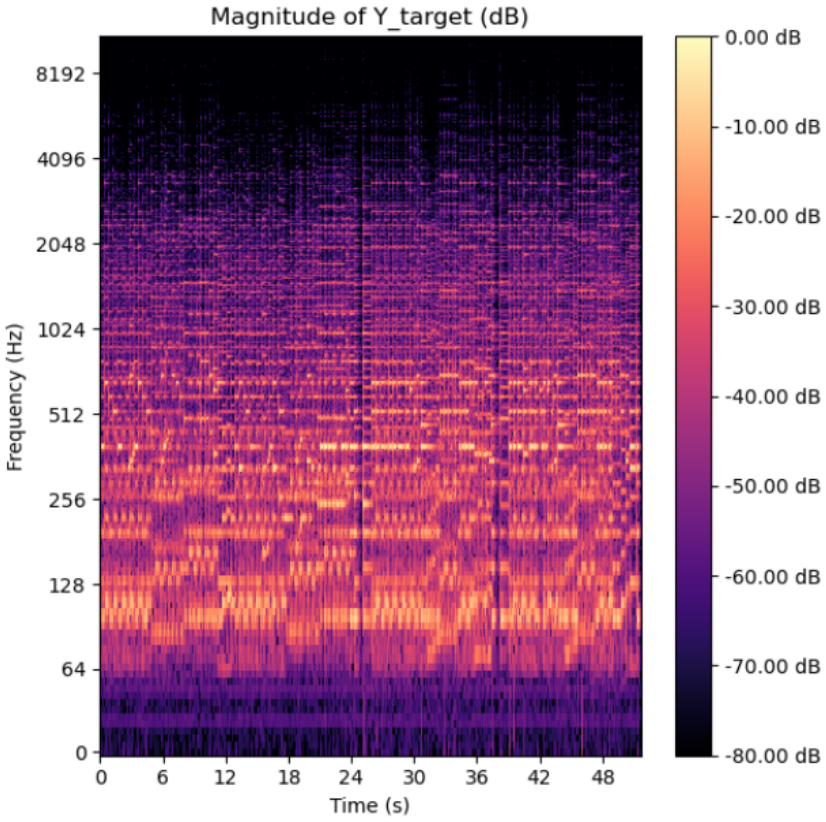

Figure 2: STFT of target **Jingle Bells Boogie**

The Jingle Bells Boogie track spectrogram suggests a blend of harmonic and percussive components, with varying intensities and patterns that contribute to its dynamic and evolving musical structure. Overall, the track demonstrates a vibrant and energetic sound profile.

2.4.1 Time Resolution

Time resolution refers to the ability to detect changes in a signal over time. It depends on the hop size H and the sampling frequency F_s, and can be expressed as:

$$\text{Time Resolution} = \frac{H}{F_s}$$

A smaller hop size H results in better time resolution, allowing for more precise tracking of rapid changes in the signal. However, this comes at the cost of reduced frequency resolution.

2.4.2 Frequency Resolution

Frequency resolution measures the ability to distinguish between different frequency components in a signal. It is determined by the window length N and the sampling frequency F_s, given by:

$$\text{Frequency Resolution} = \frac{F_s}{N}$$

A larger window length N improves frequency resolution, enabling finer differentiation of closely spaced frequencies. However, this reduces time resolution, causing changes in the signal to appear smeared over time.

2.4.3 The Time-Frequency Trade-Off

Time and frequency resolution are inversely related. This trade-off means that improving one resolution inherently worsens the other. Practical applications often require balancing these resolutions based on the nature of the signal:

- **Fast transient sounds** (e.g., percussive events): Use shorter windows for better time resolution.
- **Harmonic or tonal sounds** (e.g., sustained notes): Use longer windows for better frequency resolution.

3 Question 2: Nonnegative Matrix Factorization

Nonnegative Matrix Factorization (NMF) is a powerful dimensionality reduction and feature extraction technique. It decomposes a nonnegative data matrix V into two smaller nonnegative matrices W and H, such that:

$$V \approx WH$$

Where:

- V: The original data matrix (our audio spectrogram).
- W: A matrix whose columns represent basis components or **templates**.
- H: A matrix whose rows represent coefficients or **activations**.

Let's go deeper into each step.

3.1 Matrix Initialization

Matrix initialization prepares the key matrices W_0 (templates) and H_0 (activations) for the Nonnegative Matrix Factorization (NMF) process.

W0: The template matrix W_0 is initialized using the normalized magnitude spectrogram of the source audio, Y_{source}. Each column of W_0 represents the frequency characteristics (templates) of the source audio. Normalization ensures that the templates are scaled appropriately:

H0: The activation matrix H_0 is initialized with random nonnegative values. Each row corresponds to the time-varying activations for a specific template in W_0. These random values are iteratively refined during NMF to match the target audio's characteristics.

Xs: The normalized STFT of the source audio, is prepared similarly to W_0. This ensures consistency during reconstruction.

3.2 Performing NMF

Then we can exploit the given function with the parameters given by the assignment. After the NMF process, W remains unchanged as the spectral templates from the source, and H contains the learned activations that match the structure of the target audio under the influence of the source timbre

3.3 Frobenius Error

In this project, the Frobenius error between the original magnitude spectrogram Y_{target} and the approximated spectrogram Y_{approx} is:

$$\text{Frobenius Error} = \|Y_{\text{target}} - Y_{\text{approx}}\|_F$$

This error measures the root of the sum of the squared differences between the elements of Y_{target} and Y_{approx}. It is also sometimes referred to as the 2-norm error, especially when the difference matrix is treated as a flattened vector.

The 2-norm error between the original target magnitude Y_{target} and the approximated magnitude Y_{approx} is reported as **5005.38**. This value represents the difference between the actual and reconstructed spectrograms after performing NMF.

In the context of this project:

Magnitude of the Value:
A 2-norm error of 5005.38 is large, based on the scale and energy of the spectrogram Y_{target}.
This error reflects the limitations of using a simplified method like NMF for timbre transfer.

- **Influencing Factors:** The error is influenced by:

- The fixed W, which may not perfectly represent the target's spectral characteristics.
- The constraints applied to H, such as sparsity and smoothness, which may limit its ability to fully adapt to the target.
- **The number of iterations (50)**, which might be insufficient for convergence to a lower error.

3.4 Activation Matrix (H)

The activation matrix determine how strongly each template (column of W) is "activated" or contributes at a given time frame.

3.4.1 Initial Activation Matrix (H_0)

The plot of H_0, the initial activation matrix, reveals the following characteristics:

The matrix exhibits a **random structure**, as the values are initialized using a uniform random distribution. This lack of discernible patterns or structure is expected, as H_0 is generated using the `np.random.rand` function. It serves as a neutral starting point for the NMF process. At this stage, H_0 is **independent of the data**.

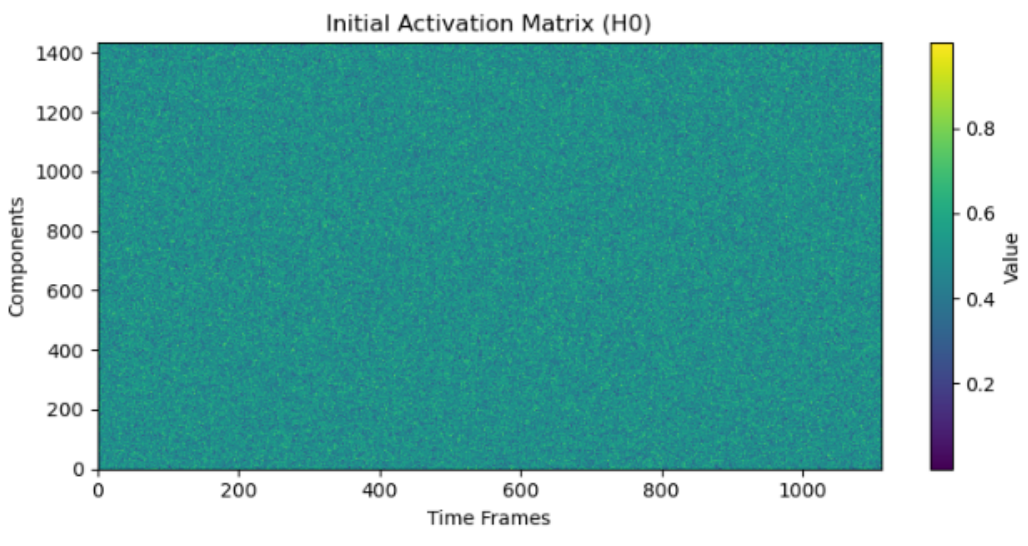

Figure 3: H0 plot

3.4.2 Activation Matrix (H)

The plot of the learned activation matrix (H) reveals the time-varying structure extracted through the NMF process. Unlike the initial random matrix (H_0), the learned H exhibits sparse and structured activations, with specific components becoming active at certain time frames. These activations represent how the source templates (W) influence the target audio, highlighting dominant frequencies over time. These lines are not really visible, I suggest to zoom in to spot them, in the next step we're going to make it more visible anyway.

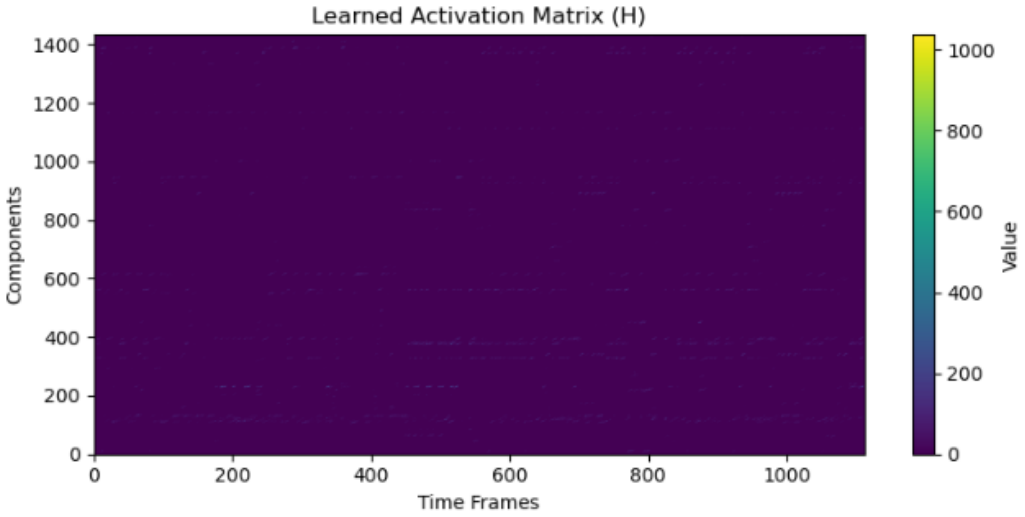

Figure 4: H0 plot

3.4.3 Compression in NMF for Visualization

Non-negative matrix factorization (NMF) often produces matrices with a wide dynamic range, making it challenging to interpret small-scale patterns in visualizations. Compression techniques are applied to **enhance visibility** and interpretability of matrices such as the template matrix W, the activation matrix H, and the reconstructed approximation WH.

The primary goal of compression is reveal finer details in the matrices.

3.4.4 Common Compression Methods

We chose to apply a power-law compression, but we could have used logarithmic one as well.

- **Logarithmic Compression**: A logarithmic transformation is applied as:

  $$X_{\text{compressed}} = \log(1 + \gamma X)$$

  where γ is a scaling factor. Logarithmic compression emphasizes small values while diminishing the impact of large values.

- **Power-Law Compression**: A power-law transformation is used as:

  $$X_{\text{compressed}} = X^{\gamma}$$

  where $0 < \gamma < 1$ controls the compression intensity. Power-law compression is smooth and particularly effective for visualizing NMF results.

3.4.5 Implementation in Visualization

To visualize the NMF matrices, compression can be applied to W, H, and WH as follows:

$$W_{\text{compressed}} = W^{\gamma}, \quad H_{\text{compressed}} = H^{\gamma}, \quad (WH)_{\text{compressed}} = (W \cdot H)^{\gamma}$$

The frequency axis can be constrained to a range of interest (e.g., 0–2000 Hz), and the time axis can be scaled using the hop size and sampling frequency. Below is an example of visualizing the matrices:

- The **template matrix** W: Displays fixed spectral patterns.
- The **activation matrix** H: Indicates time-dependent activations of the spectral patterns.
- The **reconstructed matrix** WH: Provides the approximate magnitude STFT of the original signal.
- The **original matrix** V: The actual magnitude STFT of the target signal.

3.4.6 Choosing Compression Factors

As we can see from the visualize nmf function in the notebook, a small value of gamma like 0.1 enhances a lot the details that we want to spot on the graph. Generally, values among the range (0,1) help us to better visualize the components of the matrix, while values higher then 1 have the opposed effect. In the notebook we can see a plot of the compressed functions starting from a low value of gamma, reaching a factor of gamma = 2 in the end.

At first we show the compressed H learned matrix, then the other compressed signals of the visualize nmf function

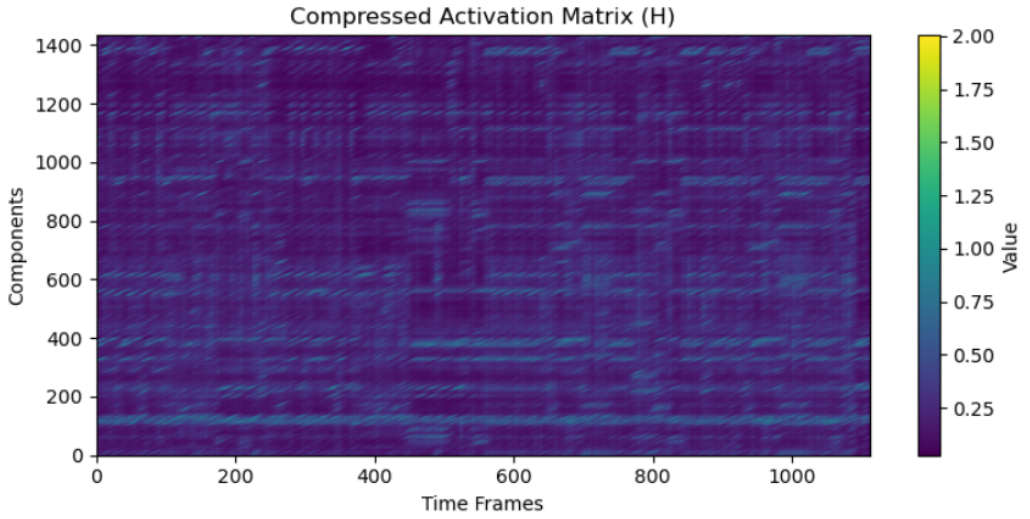

Figure 5: Compressed activation matrix

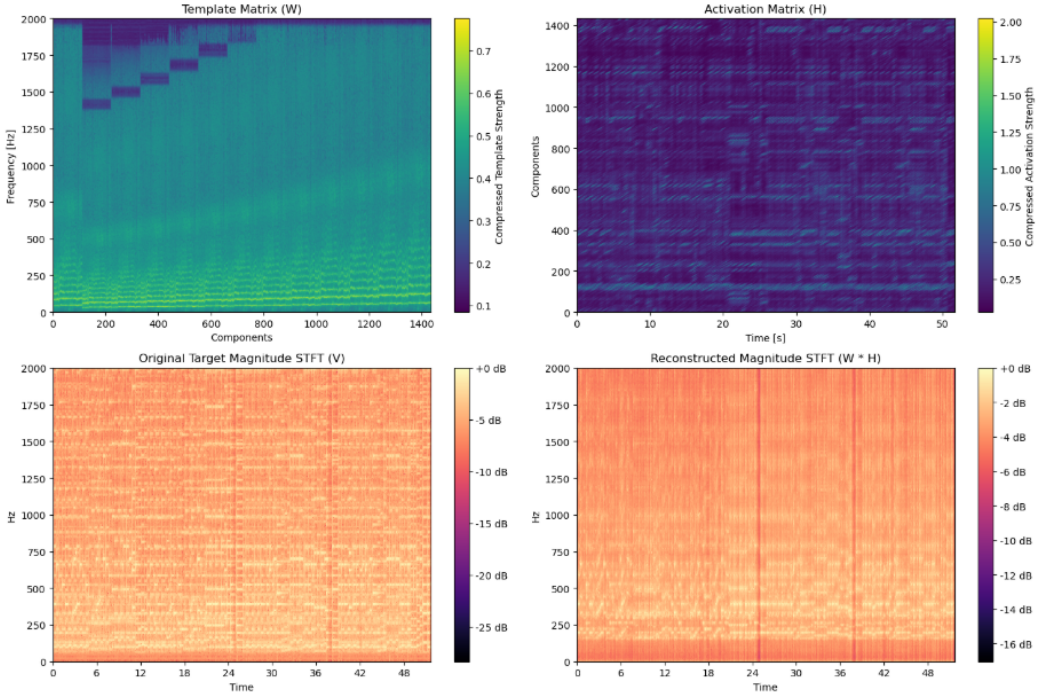

Figure 6: Compressed signals with a 0.1 gamma factor

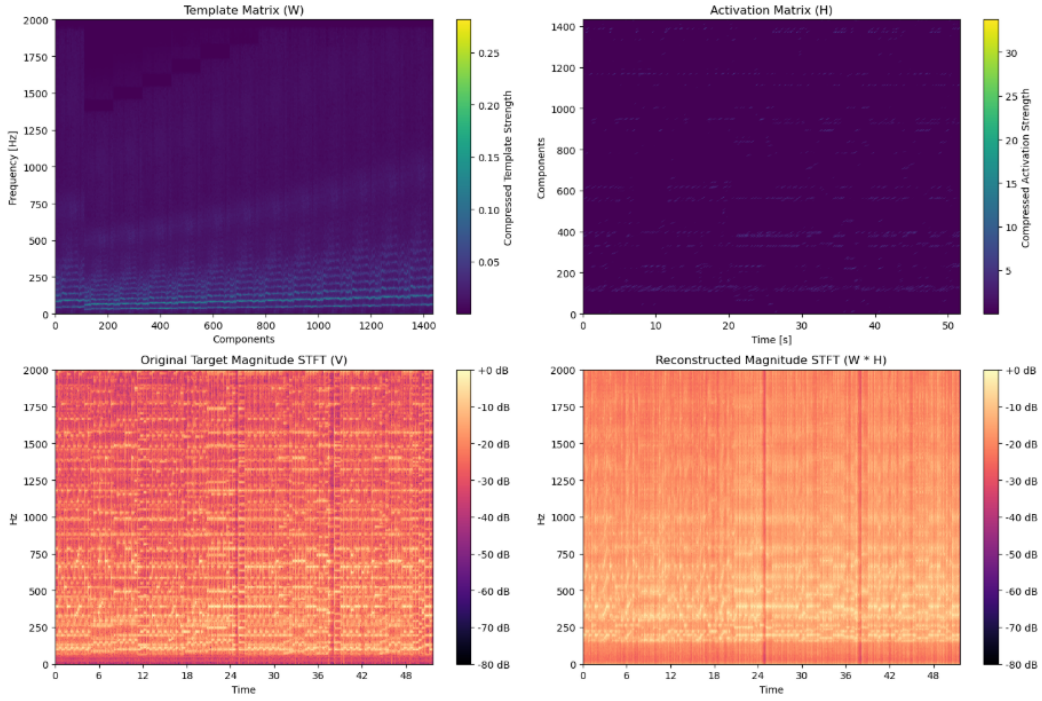

Figure 7: Compressed signals with a 0.5 gamma factor

4 Question 3: Timbre Transfer

4.1 Reconstruction of Waveforms Using ISTFT and Griffin-Lim

To synthesize the time-domain waveforms after the timbre transfer, two methods were employed: the Inverse Short-Time Fourier Transform (ISTFT) and the Griffin-Lim algorithm. Below, we detail the steps and analyze the results for both methods.

4.1.1 Replacing Magnitude with Complex Values

The learned activation matrix H was combined with the normalized magnitude spectrum of the source X_s to form the complex-valued spectrogram:

$$Y_{\text{transfer}} = X_s H, \tag{1}$$

where Y_{transfer} encodes the source timbre with the target's time-frequency structure.

4.1.2 Re-synthesis Using ISTFT

The ISTFT method reconstructs the time-domain signal from Y_{transfer} by directly applying the inverse transformation: This method uses the phase from the source STFT and does not refine it to match the target, potentially leading to artifacts.

4.1.3 Re-synthesis Using Griffin-Lim

The Griffin-Lim algorithm iteratively refines the phase to minimize reconstruction error and produce a more natural sound. The magnitude $|Y_{\text{transfer}}|$ is used, and the algorithm updates the phase iteratively: The method ensures a more accurate reconstruction but at a higher computational cost.

4.1.4 Phase Analysis

The first 10 seconds of the reconstructed signals were analyzed by computing their phase using the FFT:

- **ISTFT Phase:** The phase plot showed irregularities due to mismatched phase information.
- **Griffin-Lim Phase:** The iterative phase refinement resulted in smoother and more consistent phase behavior.

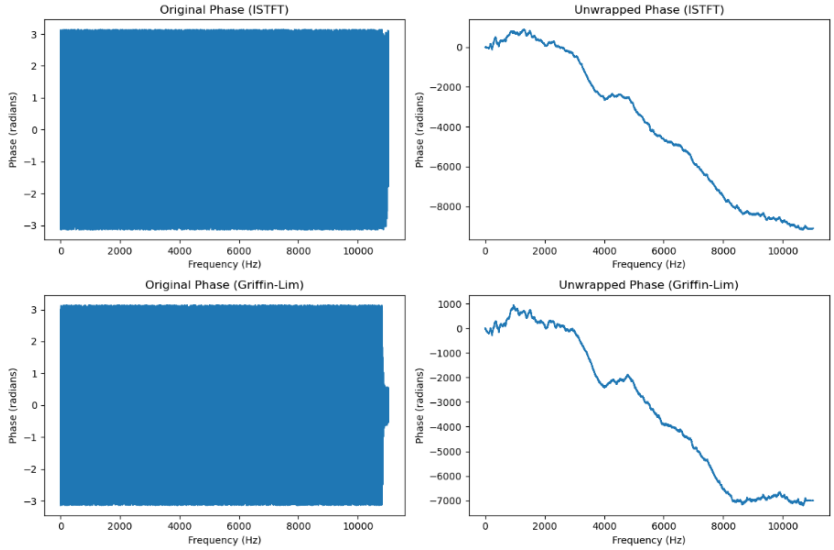

Figure 8: Unwrapped phase of istft and GL signals

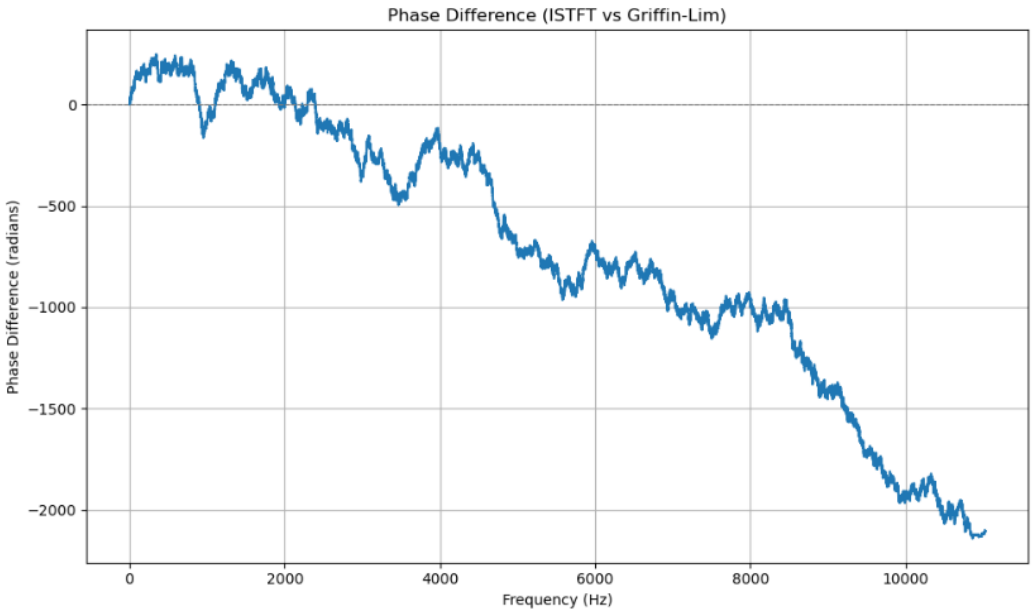

Figure 9: Difference between the two phases

This graph confirms how the two types of techniques treat phase, the approach is different and that's why we get this kind of result of a non-zero curve.

4.1.5 Timbre transfer considerations

Listening to the results, what we can say is that the timbre transfer is correctly implemented in both istft and Griffin-Lim cases. The effect lacks in detail, expecially in the istft case, while in Griffin-Lim we obtain a more detailed rhytmic behaviour. In the Griffin-Lim one audio seems to be more reactive to rapid variation of the melody.

5 Question 4: Implementazione della Funzione Generale

5.1 Function for Timbre Transfer

To simplify and automate the timbre transfer process, a Python function named `timbre_transfer` was implemented. This function integrates the key steps of the algorithm, including STFT computation, NMF factorization, and

waveform reconstruction, as described in previous sections. Below, we detail its workflow and functionality.

5.1.1 Function Description

The function `timbre_transfer` takes as input the target waveform t, the source waveform s, the sampling frequency fs, and several parameters for STFT and re-synthesis. Its purpose is to transfer the timbre of the source audio to the target audio and output the reconstructed waveform.

5.1.2 Steps of the Algorithm

1. **Compute STFTs:** The Short-Time Fourier Transform (STFT) of both the target (X_{target}) and source (X_{source}) waveforms is computed using `librosa.stft`. The magnitude spectrograms Y_{target} and Y_{source} are extracted.

2. **Normalize Source Templates:** The source magnitude Y_{source} is normalized to initialize the template matrix W_0:

$$W_0 = \frac{Y_{\text{source}}}{\sum Y_{\text{source}} + \varepsilon}, \tag{2}$$

   where ε is a small constant to avoid division by zero. Similarly, the normalized source STFT X_s is computed for later use in reconstruction.

3. **Initialize Activation Matrix:** The activation matrix H_0 is initialized with random values.

4. **Perform NMF:** Nonnegative Matrix Factorization (NMF) is applied to the target magnitude Y_{target}, using the pre-initialized W_0 and H_0. The source templates W_0 are fixed during the factorization, and constraints are applied to ensure a meaningful decomposition.

5. **Approximate Target Magnitude:** The target spectrogram is approximated using the learned activation matrix H and the normalized source templates X_s:

$$Y_{\text{transfer}} = X_s H. \tag{3}$$

6. **Reconstruct Waveform:** The time-domain waveform is reconstructed from Y_{transfer} using either:

   - **Griffin-Lim Algorithm:** Iteratively refines the phase to minimize reconstruction error.

- **ISTFT:** Uses the source phase without refinement, leading to faster but less accurate results.

7. **Visualization (Optional):** If the `plot` option is enabled, the function visualizes the NMF results, including the template matrix W, activation matrix H, and the spectrograms.

5.1.3 Function Output

The function returns the reconstructed audio waveform y, which incorporates the source timbre applied to the target audio. This implementation allows for efficient experimentation with different parameters and methods, enabling a detailed analysis of the timbre transfer process.

6 Question 5: Processing All Source-Target Combinations

In Question 5, the `timbre_transfer` function was applied to all combinations of source and target audio files. Below, we describe the process and address the related questions.

6.0.1 Explanation of the Code

- **Audio Files:** Three source audio files (`Bees Buzzing`, `Wind Blowing`, `Chainsaw Sawing`) and four target audio files (`Jingle Bells Boogie`, `Have Yourself`, `Blue Christmas`, `White Christmas`) were used.
- **Loop Structure:** The script iterates over all combinations of sources and targets.
- **Loading Audio:** Each source and target audio file is loaded using `librosa.load`, with a sampling rate of 22,050 Hz.
- **Timbre Transfer:** For each source-target pair, the `timbre_transfer` function is applied with Griffin-Lim re-synthesis (`resynth='gf'`) and plotting disabled (`plot=False`).
- **Saving Results:** The output waveform is saved in the `audio/results` directory with filenames in the format `<target>_<source>_transferred.wav`.

6.0.2 Analysis and Answers to Questions

1. **Is the algorithm able to transfer the timbre of the ssource to the target for each source and target?** Yes, the algorithm successfully transferred the timbre for all combinations. However, the quality varied depending on the source-target pair, as some combinations yielded more perceptually meaningful results.
2. **What are the best results?** The best results were observed when the harmonic content of the source aligned well with the structure of the target. For example, `Bees Buzzing` performed well with `Jingle Bells Boogie` due to their shared temporal dynamics.
3. **Why do some target-source pairs provide worse results?** Mismatches in spectral content and dynamics led to poorer results. For instance, `Chainsaw Sawing` with `Have Yourself` resulted in unnatural-sounding audio due to the high-frequency noise dominating the target's smoother harmonic structure.

6.1 Timbre Transfer Across Multiple Sources and Targets

6.1.1 Purpose

The goal is to apply the `timbre_transfer` function to all combinations of source and target audio files, systematically transferring the timbre of each source file to each target file. The results are saved as new audio files.

6.1.2 N length and Hop size tuning

To optimize the performance of the timbre transfer algorithm, we systematically tune the hop size (H) and the window length (N) used in the Short-Time Fourier Transform (STFT). This is what is done by the function **compute frobenius error** in the code.

The best parameters found in our research were N length = **2048** and Hop size = **2048**

Now we exploit this result to tune other parameters of the nmf function

6.1.3 Most Important Parameter: `fix_W`

- **Definition:** Controls whether the template matrix W is fixed (predefined using the source spectrogram) or learned during the NMF process.
- **Impact:**
  - When `fix_W=True`, the spectral patterns of the source are directly imposed on the target audio, ensuring faithful timbre transfer.
  - When `fix_W=False`, the spectral templates are learned from the target audio, increasing flexibility but diluting the source timbre.
- **Conclusion:** Fixing W is critical for effective timbre transfer as it ensures that the source's spectral characteristics dominate the factorization. In fact in the notebook we have plotted only one example in which we set fix W parameter to false, as a result we get a completely different W plot.

6.1.4 Supporting Parameters

- `cont_polyphony`: Controls the level of polyphony allowed in the target audio. Higher values allow more spectral components to overlap, capturing harmonically rich signals, while lower values result in cleaner separation.
- `cont_length`: Constrains the temporal continuity of activations in the H matrix. Higher values enforce smoother transitions over time, suitable for sustained sounds, while lower values capture rapid changes.
- `cont_grid`: Specifies the granularity of constraints applied to the H matrix. Higher values enforce stricter constraints, simplifying the factorization.
- `cont_sparsen`: Enforces sparsity on the H matrix, encouraging each activation to focus on fewer templates at a time. Sparse activations improve interpretability and focus.

So these parameters are highly dependent on the type of track we're analyzing, their effect is not as influent as the one of the fix W parameter.

6.1.5 Optimal Configuration

The configuration that minimizes the Frobenius error often balances constraints and flexibility. For most source-target pairs:

- Fixing W (`fix_W=True`) yields the most faithful timbre transfer.
- Moderate values for `cont_polyphony`, `cont_length`, and `cont_sparsen` strike a balance between capturing spectral detail and maintaining temporal smoothness.

6.2 Timbre Comparison Using MFCCs and DTW

Doing some reasearch I've found that in this implementation, we can compute the similarity between two audio signals by analyzing some of their features.

6.2.1 Timbre Representation Using MFCCs

- **Mel-Frequency Cepstral Coefficients (MFCCs):** MFCCs are features that represent the timbre of a sound by capturing its spectral envelope. Timbre is influenced by factors such as harmonic content and the distribution of energy across different frequencies.
- **Why MFCCs?**
  - MFCCs mimic the human auditory system's sensitivity to different frequency bands, making them ideal for perceptual audio comparisons.
  - They reduce the dimensionality of the spectrogram while retaining essential information about the sound's timbre.

6.2.2 Comparing MFCCs Using DTW

- **Dynamic Time Warping (DTW):** DTW is used to align the MFCC features of two audio signals. It accounts for temporal distortions by allowing non-linear alignment of time steps, which compensates for variations in timing or rhythm between the signals.
- **Cumulative Cost as Similarity Score:**
  - DTW produces a *distance matrix* that represents the cumulative cost of aligning the MFCCs of one signal to the other.
  - The final cumulative cost (`distance_matrix[-1, -1]`) serves as the similarity score:
    - **Lower cost:** Indicates greater similarity, as fewer adjustments were needed to align the timbral profiles.
    - **Higher cost:** Suggests more significant differences in timbre or structure between the signals.

6.2.3 Example Usage

In the example, we compared an audio file to itself:

- The two inputs are identical (`"audio/results/Jingle_Bells_Boogie_Bees_Buzzing_`
- Since they are the same file, the similarity score is really close to zero, indicating no difference in their timbres. Then we compared Jingle Bells Boogie to its Beez Buzzing version: as a result we get

an high value (close to 79), meaning that the two audio files have different timbre.