Low-rank Decompositions of Musical STFT Tensors

Nicholas Richardson

December 15, 2022

1 Introduction

Many tools exist for analyzing audio data, but one of the most widely used is the short-time Fourier transform (STFT) either on its own, or in conjunction with classical or modern approaches [1]. Another set of tools, tensor decompositions, have proven useful in compressing, approximating, and synthesizing high order data across applications [2]. So we ask how tensor decompositions may be used on STFTs to examine such signals. The main problem we are interested in is how to analyze audio recordings of musical instruments and extract important information about the notes and rhythms played. One way we may do this is to consider low-rank decompositions where each term in the factorization corresponds to a different feature, similar to [3]. We may expect these STFTs to have some low-rank structure since single instrument recordings typically contain a sequence of notes, where each note has a fixed set of frequencies and overall amplitude [4].

This report begins with necessary background information such as signal processing, the short-time Fourier transform, and non-negative canonical polyadic tensor decomposition in Section 2. The two main experiments and results are discussed in Section 3. Lastly, since the project is more exploratory in nature, we conclude with a summary of the findings and further work to consider in Section 4.

2 Background

2.1 Signal Processing

The main problem revolves around working with audio signals. For our purposes, we use the Waveform Audio File (WAV) format, which can be treated as a single vector $y \in \mathbb{R}^N$. We think of the entries $y_n = y(t_n)$ as samples from a continuous real function $y : [0, T] \longrightarrow \mathbb{R}$ where the time points are evenly spaced on some domain $[0, T]$. This format also stores the sampling rate $f = T/N$ samples per second which can be used to recover the time points t_n.

When working with audio, it is common practice in musical recordings to peak-normalize the vectors by dividing by the maximum absolute value entry so that $\|y\|_\infty = \max_n |y_n| = 1$ [5]. This limits the range of values between -1 and 1 so different songs can be processed at the same volume. This also ensures accurate playback since most audio engines "clip" any values outside $[-1, 1]$ by replacing them with a -1 or 1.

2.2 Short-time Fourier Transform

To better analysis the signals, the (discrete) Short-time Fourier Transform (STFT) is used [6]. This transforms a vector $y \in \mathbb{R}^N$ into the matrix $\mathbf{STFT}(y) = Y \in \mathbb{C}^{K,L}$ according to

$$Y_{k,l} = \sum_{n=0}^{N-1} y_{n+1} w_{l-n-1} e^{-\frac{i2\pi}{N} kn}.$$

This can be thought of as performing a Fourier Transform along a short time interval given by a sliding window w. The k, l entry of Y then gives information about the frequency k localized at the time point t_{n_l}.

In practice, L may only be a fraction of N (not all time points t_l are considered) where the window hops from being centred at t_{n_l} to $t_{n_{l+1}}$ with a hop size $h = n_{l+1} - n_l$.

The window we use for the experiments in Section 3 is the Hann window which is well suited for musical signals [7]. It is defined as $w_n = \sin^2(\pi n/N)$, where the window is zero for $n < 0$ and $n \geq W$. We use a "width" $W = 40$ and hop size $h = W/2 - 1 = 19$ samples in Section 3. This choice of width is used to balance the trade off between good frequency resolution (large widths) and good time resolution (small widths) according to the uncertainty principle [6].

The magnitude of the STFT entries gives the amplitude of a frequency at a particular point in time and ignores the complex phase. For many audio applications, this is sufficient to analysis the signal, since humans can detect changes in amplitude but not in phases in general. For this reason, we give a name to the entry-wise magnitude of the STFT, $|Y|$, and will refer to this as the (magnitude) spectrogram. Note that other definitions are commonly used such as $|Y|^2$ or $\log|Y|$ [7].

Now given a STFT $Y \in \mathbb{C}^{K,L}$, it is also possible to recover the time domain signal $y = \mathbf{STFT}^{-1}(Y)$ similar to the inverse of the Fourier transform. But this inversion cannot be done directly on a spectrogram $|Y| = S \in \mathbb{R}^{K,L}$. To go from S to y, we must add some complex phase to the entries $Y_{kl} = S_{kl}\exp(i\Theta_{kl})$. There are sophisticated methods for estimating what this phase should be for a given spectrogram as explored in [8], but one simple estimation is to reuse the the complex angles of the STFT $\Theta = \angle Y$. On complex numbers $z = re^{i\theta} \in \mathbb{C}$, we have $\angle z = \theta \in \mathbb{R}$. On a complex matrix $Z \in \mathbb{C}^{m\times n}$, $\angle Z \in \mathbb{R}^{m\times n}$ acts element-wise. In the case we have the full spectrogram $S = |Y|$, this does recover the signal y exactly. The advantage of decoupling the magnitude from the phase is that this allows us to easily process and work with a real-valued and non-negative matrix S. And if we make modifications or approximations of the spectrogram $\tilde{S}$, such as the low-rank decomposition performed in Section 3, we will still be able to invert it back to a standard time domain signal.

2.3 Non-negative Canonical Polyadic Decomposition

The tensor decomposition algorithm we use is Xu and Yin's block coordinate descent method [9] applied to the non-negative canonical polyadic decomposition (NNCP) problem. Their method can be applied to a tensor of any order, but for our problem we assume a 3-order tensor $T \in {\mathbb{R}_{\geq 0}}^{n_1\times n_2\times n_3}$ is given. The method also requires an assumed tensor rank $r \in \mathbb{N}$ of T.

The problem can be stated as finding a decomposition

$$T = \sum_{l=1}^{r} a^{(l)} \otimes b^{(l)} \otimes c^{(l)}$$

of T where $a^{(l)} \in \mathbb{R}^{n_1}$, $b^{(l)} \in \mathbb{R}^{n_2}$, $c^{(l)} \in \mathbb{R}^{n_3}$. Here, we wish to find the a, b, c's which minimize the squared error of the entries. Writing $A \in {\mathbb{R}_{\geq 0}}^{n_1\times r}$ where $A_{il} = a_i^{(l)}$ and similarly for B, C, we summarize the problem as the following:

$$\begin{aligned}
&\min_{A,B,C} \frac{1}{2}\|T - A\circ B\circ C\|_F^2 \\
&= \min_{A,B,C} \frac{1}{2}\sum_{ijk}\left(T_{ijk} - \sum_l A_{il}B_{jl}C_{kl}\right)^2 \\
&= \min_{A,B,C} F(A,B,C).
\end{aligned}$$

To solve this, we iterate over the factors A, B, C, keeping the other two fixed, with the following update scheme:

$$A_{n+1} = \arg\min_A \langle \nabla_A F_n, A - \hat{A}_n\rangle + \frac{\lambda_n^{(A)}}{2}\|A - \hat{A}_n\|_F^2. \tag{1}$$

The first term of (1) encourages the new step to go in the direction of the negative gradient, and the second term ensures the new step is not too far from the old one. This has the closed form solution

$$A_{n+1} = \max(0, \hat{A}_n - \nabla_A F_n / \lambda_n^{(A)}). \tag{2}$$

Note in (2) that we update A_{n+1} using $\hat{A}_n = A_n + \omega_n^{(A)}(A_n - A_{n-1})$ rather than the previous step A_n. This allows for some momentum in the update to prevent getting stuck in small local minimums.

To fully describe the method, we must pick $\lambda_n^{(A)}$ and $\omega_n^{(A)}$. These are selected to optimize the decent at each iteration, and the full details of their derivation can be seen in [9]. We use the inverse step size

$$\lambda_n^{(A)} = \left\| (M_n^{(A)})^\top M_n^{(A)} \right\|_2, \quad M_n^{(A)} = B_n * C_n$$

and momentum

$$\omega_n^{(A)} = \min\left(\frac{t_{n-1} - 1}{t_n}, \frac{1}{2}\sqrt{\frac{\lambda_{n-1}^{(A)}}{\lambda_n^{(A)}}} \right)$$

where the variable t_n is updated as $t_{n+1} = \frac{1}{2}\left(1 + \sqrt{1 + 4t_n^2}\right)$. The $*$ symbol is used to denote the Khatri-Rao product or the column-wise Kronecker product

$$A * B = [\text{Vec}\,(a^{(1)} \otimes b^{(1)}), \ldots, \text{Vec}\,(a^{(r)} \otimes b^{(r)})] \in \mathbb{R}^{n_1 n_2 \times r}$$

for matricies $A \in \mathbb{R}^{n_1 \times r}$ and $B \in \mathbb{R}^{n_2 \times r}$. Lastly, we initialize A_0, B_0, and C_0 with independent standard normal distributed entries, and $t_0 = 1$.

3 Experiments

3.1 Data & Software

The songs used in the experiment are taken from the MUSDB18HQ song dataset [10]. This dataset contains 150 full songs in WAV format, in addition to the isolated bass, drums, vocals, and other instruments. To reduce the size of the problem, only the first 27 songs in the dataset are considered.

The experiments were coded using the Julia Language, and the implementation of the NNCP algorithm from the TensorDecompositions package is used for the tensor decomposition [11]. The full code is available on GitHub.[1]

3.2 2-order Tensor

The first experiment conducted can be treated as a diagnostic run to ensure everything is set up correctly, and we can interpret the results correctly. We first consider the bass track to the first song "Night Owl" by A Classic Education. Our tensor $T \in \mathbb{R}^{2049 \times 106 \times 1}$ is constructed by setting $T_{i,j,1} = |Y_{i,j}| = |\mathbf{STFT}(y)_{i,j}|$ where 5 seconds of the song is used at the input signal y. We start the song at the 10 second mark to skip over any initial silence in the intro before the bass enters. Figure 1 displays the song y in the usual time domain, and its spectrogram $|Y|$.

We perform rank-1 NNCP decomposition $T = a \otimes b \otimes c$ to recover the important frequencies a, amplitude b, and (in this case) a redundant scalar c. This is essentially a non-negative rank-1 matrix factorization. The reason for appending a trivial a dimension is only to reuse the same code for the following experiment in Section 3.3.

In Figure 2, we see the average harmonic frequencies of the bass are extracted in a. This is similar to what the magnitude of a standard Fourier transform would yield. The factor b represents what can be thought of as the time varying "envelope" or instantaneous amplitude of the notes being played [12]. Although there exists classical ways of recovering this envelope, this non-negative rank-1 decomposition on the song has recovered the frequencies and envelope jointly.

[1] https://github.com/njericha/TensorDecompSTFT

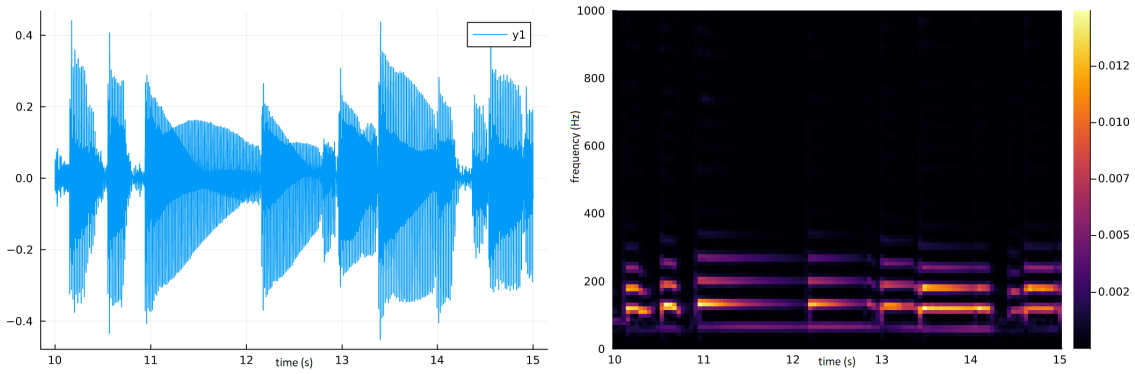

Figure 1: (Left) The input bass signal y. Here the time is given as seconds from the start of the song. (Right) Spectrogram $|Y|$ of the input bass signal.

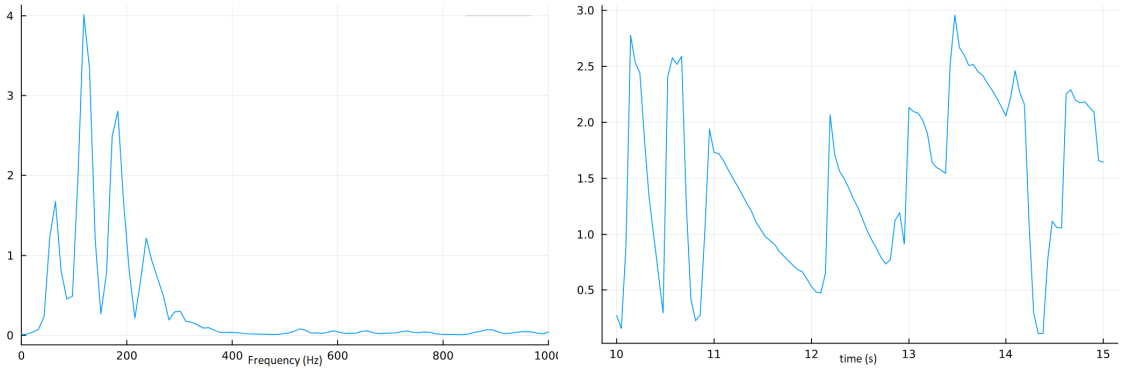

Figure 2: The learned factors a (left) and b (right) in the tensor decomposition of $T = a \otimes b \otimes c$. The first factor corresponds to the average amplitude of that frequency, and the second corresponds to the relative amplitude of the signal at that time.

3.3 3-order Tensor

Similar to the first experiment, we construct our tensor $T \in \mathbb{R}^{2049\times106\times27}$ where $T_{i,j,k} = |Y^k_{i,j}|$ and y^k is the bass part of the k^{th} song in the dataset. Again, we only use 5 second clips of the songs starting at the 10 second mark to skip over any intro silence before the bass parts enter. This time, we perform rank-11 NNCP decomposition $T = A \circ B \circ C = \sum_{l=1}^{11} a^{(l)} \otimes b^{(l)} \otimes c^{(l)}$ to recover the important frequencies A, their rhythms B, and presence in each song C. We also peak-normalize each song y^k is to ensure all songs are at the same volume and the low-rank factors don't favour approximating one song over another.

We justify our assumption that the tensor is well approximated by a rank-11 tensor by considering the musical nature of the recordings. Western music commonly uses the same 11 notes[2] and their octaves. The different octaves of each note do have distinct frequency signatures, but they differ by the addition or removal of one or two frequencies in a typically bass guitar range. So for our purposes, we expect this to give a close enough approximation.

The resulting factors $a^{(l)}$, $b^{(l)}$, and $c^{(l)}$ after performing NNCP decomposition are displayed in Figure 3. Amazingly, the algorithm identifies commonly used notes across the 27 songs in the factors $a^{(l)}$. The note's rhythms are captured in the factors $b^{(l)}$, and the factors $c^{(l)}$ highlight which song uses the note $a^{(l)}$.

[2]I acknowledge I have made an error in this assumption. There are in fact 12 common notes used in western music: A, $A^\sharp/B^\flat$, B, C, $C^\sharp/D^\flat$, D, $D^\sharp/E^\flat$, E, F, $F^\sharp/G^\flat$, G, and $G^\sharp/A^\flat$. The slashes are used to indicate enharmonically equivalent notes. Rather than redoing the code and figures, the idea and conclusions from the results still follow, with the understanding that we may be missing a note if all 12 notes be present in the 27 songs examined.

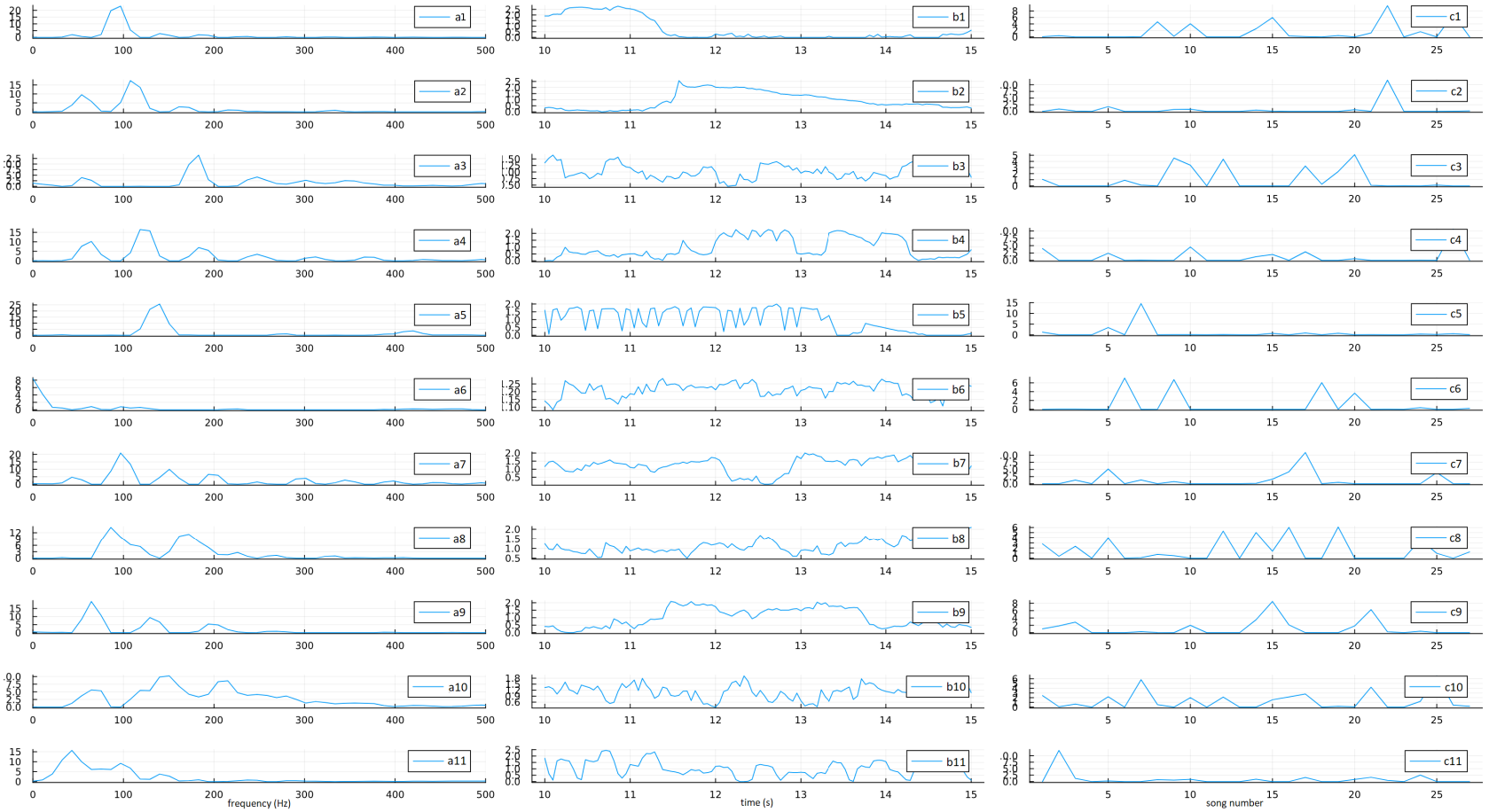

Figure 3: (Left to right) Frequency factors $a^{(l)}$, their time envelopes $b^{(l)}$, and song activation $c^{(l)}$ from the experiment performed in Section 3.3.

Comparing this experiment to the previous one, the rank-1 decomposition experiment from Section 3.2 performs something similar to a single Fourier transform and envelope extraction. This rank-11 decomposition however simultaneously performs a feature identification, and Fourier-like transform and envelope extraction of each note. Additionally, each feature is labelled in such a way that we know which songs use each note.

To investigate this further with an example, we will consider the $k = 22^{\text{th}}$ song in the dataset. Looking at the $c^{(l)}$ factors in Figure 3, we see the $l = 1$ and $l = 2$ terms both have a large spike at the 22^{th} entries $(c^{(1)})_{22}$ and $(c^{(2)})_{22}$. The rest of the $(c^{(l)})_2 2$ entries are approximately 0. This suggests the song $k = 22$ is made up of the two notes $a^{(1)}$ and $a^{(2)}$ with the rhythms $b^{(1)}$ and $b^{(2)}$. Finding the maximum peaks of $a^{(1)}$ and $a^{(2)}$, we can identify these notes as F and $A^\sharp$ with frequencies 90 Hz and 110 Hz.

We can then recover the song $k = 22$ by the following procedure. First, take the tensor product the frequency and time factors to obtain a spectrogram for each note $S^{(l)} = a^{(l)} \otimes b^{(l)}$. Then we can add the complex phase from the original song to get our STFT matrix $Y^{(l)} = S^{(l)} e^{i \angle Y^k}$. Finally, we take the inverse STFT to recover our time domain signals of each note $y^{(l)} = \mathbf{STFT}^{-1}(Y^{(l)})$. The resulting peak-normalized signals are plotted in Figure 4. Our low-rank approximation of this song is then the sum of these two term $\tilde{y}^k = y^{(1)} + y^{(2)}$. This approximation is compared with the original song in Figure 5. We can see in this figure how close these signals are. The audio of these two signals can be heard in the `slideshow.ipynb` file on GitHub.

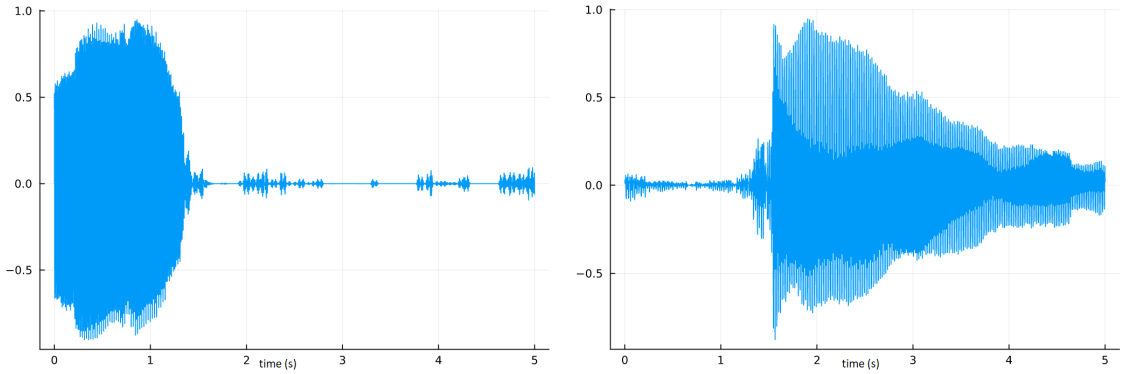

Figure 4: The recovered time domain signals $y^{(l)} = \mathbf{STFT}^{-1}\left((a^{(l)} \otimes b^{(l)}) e^{i \angle Y^{22}}\right)$, peak-normalized, from the first two terms of the rank-11 decomposition of T for $l = 1$ (left) and $l = 2$ (right).

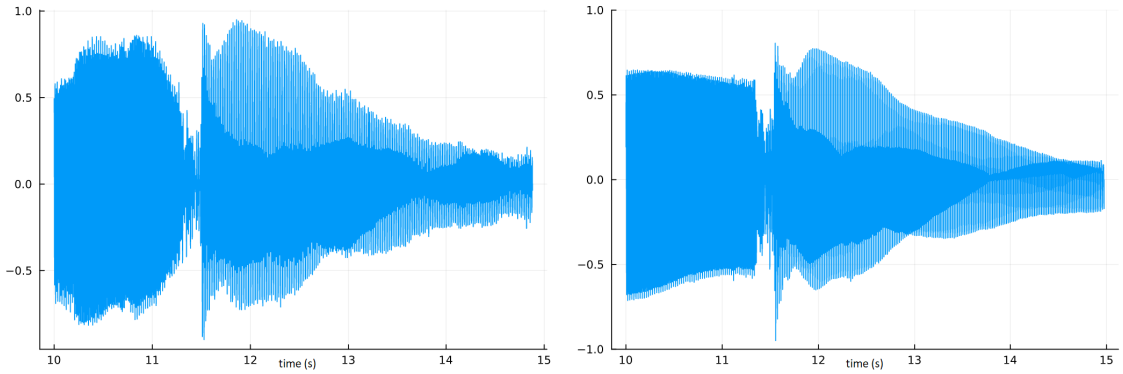

Figure 5: Comparison between the reconstructed song $\tilde{y}^k$ (left) and the original song y^k (right) for the $k = 22$ song in the dataset. The reconstruction is created by summing the first two rank-1 terms as displayed in Figure 4.

4 Conclusion

4.1 Summary

For this project, we tested a non-negative low-rank tensor factorization method on musical data. Specifically, we found the Non-negative Canonical Polyadic Decomposition method proposed by Xu and Yin successfully approximates the spectrograms of bass guitar song clips. Additionally, NNCP extracts the notes and their rhythms for the songs, and finds which songs use which notes.

4.2 Further Work

Some interesting areas that are worth exploring further include the following ideas. Different instruments such as drums can be tested to see how well this low-rank model can identify common drum sounds and rhythms. The main function we are minimizing is the sum of squared differences on the entries directly, but we could consider using a log-magnitude spectrum to more heavily weight the higher frequencies, which is typically orders of magnitude smaller in amplitude. In the second experiment, only the first two low rank terms are examined, so it would be worth going through each term to verify if all 11 terms are in fact different musical notes. We also rely on using the complex phase from the original signal to reconstruct each low rank term to playable audio. A more sophisticated phase estimator could be used to better reconstruct these terms. Finally, a great test of this decomposition method would be on full length songs rather than short 5-second clips, and on the full dataset available.

5 Acknowledgements

I would like to acknowledge this work was done as part of the MATH 605D course at the University of British Columbia and thank the teaching assistant Mathew Faltyn and professor Elina Robeva for all their work in the course this term. It was a pleasure to learn about the exciting field of tensor decompositions and everything in the course really inspired this project.

6 References

[1] Hendrik Purwins et al. "Deep Learning for Audio Signal Processing". In: *IEEE Journal of Selected Topics in Signal Processing* 13.2 (May 2019), pp. 206–219. ISSN: 1941-0484. DOI: 10.1109/JSTSP.2019.2908700.

[2] Tamara G. Kolda and Brett W. Bader. "Tensor Decompositions and Applications". In: *SIAM Review* 51.3 (Aug. 6, 2009), pp. 455–500. ISSN: 0036-1445, 1095-7200. DOI: 10.1137/07070111X.

[3] Laetitia Gauvin, André Panisson, and Ciro Cattuto. "Detecting the Community Structure and Activity Patterns of Temporal Networks: A Non-Negative Tensor Factorization Approach". In: *PLOS ONE* 9.1 (Jan. 31, 2014), e86028. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0086028.

[4] Valentin Emiya, Ronan Hamon, and Caroline Chaux. "Being Low-Rank in the Time-Frequency Plane". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Apr. 2018, pp. 4659–4663. DOI: 10.1109/ICASSP.2018.8462423.

[5] Marc Le Brun. "Digital waveshaping synthesis". In: *Journal of the Audio Engineering Society* 27.4 (1979), pp. 250–266.

[6] Karlheinz Gröchenig. *Foundations of Time-Frequency Analysis*. Red. by John J. Benedetto. Applied and Numerical Harmonic Analysis. Boston, MA: Birkhäuser, 2001. ISBN: 978-1-4612-6568-9 978-1-4612-0003-1. DOI: 10.1007/978-1-4612-0003-1.

[7] Meinard Müller. "Fourier Analysis of Signals". In: *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. Ed. by Meinard Müller. Cham: Springer International Publishing, 2015, pp. 39–114. ISBN: 978-3-319-21945-5. DOI: 10.1007/978-3-319-21945-5_2.

[8] D. Griffin and Jae Lim. "Signal estimation from modified short-time Fourier transform". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 32.2 (1984), pp. 236–243. DOI: 10.1109/TASSP.1984.1164317.

[9] Yangyang Xu and Wotao Yin. "A Block Coordinate Descent Method for Regularized Multiconvex Optimization with Applications to Nonnegative Tensor Factorization and Completion". In: *SIAM Journal on Imaging Sciences* 6.3 (2013), pp. 1758–1789. DOI: 10.1137/120887795.

[10] Zafar Rafii et al. *MUSDB18 - a Corpus for Music Separation*. Version 1.0.0. Zenodo, Dec. 17, 2017. DOI: 10.5281/ZENODO.1117372.

[11] Jeff Bezanson et al. "Julia: A Fresh Approach to Numerical Computing". In: *SIAM Review* 59.1 (Jan. 2017), pp. 65–98. ISSN: 0036-1445. DOI: 10.1137/141000671.

[12] "Envelope of a Bandpass Signal". In: *Software Receiver Design: Build Your Own Digital Communication System in Five Easy Steps*. Ed. by Andrew G. Klein, Jr Johnson C. Richard, and William A. Sethares. Cambridge: Cambridge University Press, 2011, pp. 416–420. ISBN: 978-1-139-00522-7. DOI: 10.1017/CBO9781139005227.026.