LARGE SCALE POLYPHONIC MUSIC TRANSCRIPTION USING RANDOMIZED MATRIX DECOMPOSITIONS

İsmail Arı, Umut Şimşekli, Ali Taylan Cemgil, Lale Akarun

Bogazici University, Department of Computer Engineering, TURKEY

ABSTRACT

Polyphonic music transcription is a fundamental problem in computer music and over the last decade many sophisticated and application-specific methods have been proposed for its solution. However, most techniques cannot make fully use of all the available training data efficiently and do not scale well beyond a certain size. In this study, we develop an approach based on matrix factorization that can easily handle very large training corpora encountered in real applications. We evaluate and compare four different techniques that are based on randomized approaches to SVD and CUR decompositions. We demonstrate that by only retaining the relevant parts of the training data via matrix skeletonization based on CUR decomposition, we maintain comparable transcription performance with only 2% of the training data. The method seems to compete with the state-of-the-art techniques in the literature. Furthermore, it is very efficient in terms of time and space complexities, can work even in real time without compromising the success rate.

Index Terms— Polyphonic music transcription, CUR decomposition, singular value decomposition, matrix skeletonization, randomized matrix decompositions

1. INTRODUCTION

Automatic music transcription is one of the fundamental problems studied in the field of audio processing. Given polyphonic music, the aim is to recognize the notes and the time interval in which they are active. The methods developed to solve this problem find place in various areas such as phonetics, speech processing, and music information retrieval [1]. Transcription is closely related to pitch detection and tracking, and considerable amount of research has been devoted to this topic. The methods in the literature can be roughly divided into two groups as algorithmic and model-based approaches [1]. In the last few years, model-based methods on matrix factorizations have become very popular [1, 2, 3].

İ. A. and L. A. are supported by TÜBİTAK grant 108E161. U. Ş. and A. T. C. are supported by TÜBİTAK grant 110E292. U. Ş. is also supported by TÜBİTAK BİDEB 2211 grant.

It is still not very clear how humans recognize musical notes in polyphonic textures. Experience suggests that human listeners become more successful with training in recognizing musical constructs. Inspired partially by this idea, Smaragdis has demonstrated that it is possible to perform polyphonic pitch tracking successfully via a linear model that tries to approximate the observed musical data as a superposition of previously recorded monophonic musical data [4]: $\mathbf{X} \approx \mathbf{DW}$ where $\mathbf{X}$ is the observed spectrogram, $\mathbf{D}$ is the dictionary matrix obtained from the training data, and $\mathbf{W}$ contains the corresponding weights.

This approach is expensive since it makes use of a potentially very large dictionary matrix $\mathbf{D}$. One should express the dictionary matrix using fewer dimensions in order to perform a faster transcription with a smaller computation and memory requirement.

The basic approach to reduce the dimensionality of $\mathbf{D}$ relies on Singular Value Decomposition (SVD) where $\mathbf{D} = \mathbf{A\Sigma B}^\intercal$ [5]. $\mathbf{A}$ and $\mathbf{B}$ are the orthonormal matrices holding the left and right singular vectors, respectively, and the diagonal matrix $\mathbf{\Sigma}$ includes the singular values in descending order. The k-rank reduced SVD approximates $\mathbf{D}$ by using only the first k singular vectors and singular values and it gives the best low-rank approximation of the matrix in terms of Frobenius (or l_2) norm of the reconstruction error. Unfortunately, SVD cannot be computed by conventional techniques when the data matrix is very big. There are randomized techniques to solve a reduced SVD that approximates the data matrix well in practice within a tolerable error bound [6, 7].

SVD gives the best results when reconstruction error is in question, but it is mostly criticized in terms of reification issues where the singular vectors may not represent *physical reality* [8]. Besides, the matrix which is reconstructed by using reduced SVD might contain negative values even though the original matrix is non-negative as in the case of spectrogram data.

The dictionary matrix can also be expressed as a collection of its individual rows and columns, which already represent physical identities. This decomposition is known as CUR decomposition in the literature. It attracted considerable attention in the last few years [6, 7, 9]. In this case, $\mathbf{D} \approx \mathbf{CUR}$ where $\mathbf{C}$ and $\mathbf{R}$ represent columns and rows of the matrix, respectively, and $\mathbf{U}$ is computed as a link matrix

between them. CUR decomposition can fully represent the matrix by selecting columns/rows that span the whole column/row space of $\mathbf{D}$ as well as approximating it with fewer dimensions. Halko *et al.* [6] and Mahoney [7] present an extensive summary of the current deterministic and randomized CUR decomposition techniques with relative and additive error bound analysis for the reconstruction. When compared to SVD, CUR leads to a more interpretable decomposition since its basis vectors are individual rows and columns [7] with physical correspondence. Furthermore, for the case of sparse data as in a spectrogram of polyphonic music samples, CUR maintains the sparsity whereas SVD may give a dense decomposition.

In this study, we investigate fast and efficient methods for polyphonic piano transcription based on linear decompositions of spectrogram data, notably [4]. We develop various ways to handle big amount of data efficiently within the same algorithmic framework. In particular, we employ randomized approaches to SVD and CUR computations to significantly reduce the size of the dictionary without compromising the success rate of the full solution.

2. POLYPHONIC MUSIC TRANSCRIPTION

Before discussing the methods, let us start by clarifying the problem definition where we will consider it as a machine learning problem.

Let $\mathbf{D}_i$, with elements $D_i(f, \tau_i)$, denote the magnitude spectrogram of monophonic music recordings belonging to I different notes. Here, $i = 1, \ldots, I$ is the note index, $f = 1, \ldots, F$ is the frequency index, and $\tau_i = 1, \ldots, N_i$ is the time index where F is the number of frequency bins and N_i is the number of columns in $\mathbf{D}_i$. Since we are interested only in the pitch of the data, we remove the effect of sound intensity by normalizing each vector such that its elements sum up to 1. We also remove the samples below some loudness level. We obtain the training data by concatenating all training vectors, $\mathbf{D} = [\, \mathbf{D}_1 \, \mathbf{D}_2 \, \ldots \, \mathbf{D}_I \,]$ where there are a total number of $N = \sum_{i=1}^{I} N_i$ training samples. Test data are composed of polyphonic recordings. We have considered piano samples in this paper, but the method is also valid for the generic case where multiple instruments are played together. Let $\mathbf{X}$, with values $X(f, t)$, be the spectrogram of the test data where $t = 1, \ldots, T$ and T is the number of time frames.

2.1. Linear model approach

In this study, we use a basic linear model that describes the relationship between the observed and the training data where the observed spectrum is expressed as a superposition of the training vectors:

$$\mathbf{X} \approx \hat{\mathbf{X}} = \mathbf{D}\mathbf{W} \tag{1}$$

The aim is to find the weight matrix $\mathbf{W}$ which minimizes $\mathcal{D}\left[\mathbf{X} \| \mathbf{D}\mathbf{W}\right]$ where $\mathcal{D}[\cdot\|\cdot]$ is a properly selected cost function. We choose KL divergence for as the cost function. Note that the model is identical to the NMF model whose update rule is well known [10, 2]. We start with random initialization of $\mathbf{W}$, and continue with the following step until convergence:

$$\mathbf{W} \leftarrow \mathbf{W} \odot \left(\frac{\mathbf{D}^\intercal \frac{\mathbf{X}}{(\mathbf{D}\mathbf{W})}}{\mathbf{D}^\intercal \mathbb{1}} \right) \tag{2}$$

The $\odot$ symbol stands for the Hadamard product implying element-wise multiplication and $\mathbf{D}^\intercal$ is the transpose of $\mathbf{D}$. The division operation is also done element-wise. $\mathbb{1}$ is a matrix of all ones of the same size with $\mathbf{X}$. Active notes are observed to have significantly higher weights than the inactive ones and they are selected by thresholding. In order to obtain a single weight for a particular note, the corresponding weights are summed up since there might be multiple weights corresponding to the same note. Additionally, each weight value is smoothed by applying a median filter to remove sharp jumps.

The computational bottleneck of the algorithm is seen to be in the matrix-matrix multiplications involving $\mathbf{D}$ and $\mathbf{D}^\intercal$ in Eq. (2). The sizes of $\mathbf{X}$, $\mathbf{D}$, and $\mathbf{W}$ are $F \times T$, $F \times N$ and $N \times T$, respectively. So, the running time is dominated by the matrix multiplication operation which is in order of $O(FNT)$. Space complexity is $O(FN)$ since we store all elements in $\mathbf{D}$. Note that Eq. (2) can work in parallel on the columns of $\mathbf{W}$ since each column of $\mathbf{W}$ can be computed independently. So, the time complexity can be reduced to $O(FN)$.

2.2. Improving efficiency via SVD

The dictionary matrix may involve millions of columns and/or rows. Besides, it may not fit in RAM in real applications. Our aim is to make the method efficient in terms of both time and space complexity. The key point is to decompose $\mathbf{D}$ and use its decomposition instead of taking whole of it. A rank k approximation of a matrix can be obtained best by the reduced SVD [5]:

$$\underset{\tilde{\mathbf{D}},\ \mathrm{rank}(\tilde{\mathbf{D}}) \leq k}{\arg\min} \|\mathbf{D} - \tilde{\mathbf{D}}\|_F = \mathbf{A}_k \mathbf{\Sigma}_k \mathbf{B}_k^\intercal \tag{3}$$

where k, $\mathbf{A}_k$, $\mathbf{\Sigma}_k$ and $\mathbf{B}_k$ stand for the rank, the orthonormal matrix involving the left singular vectors, the diagonal matrix involving the k largest singular values in descending order, and the orthonormal matrix involving the right singular vectors, respectively.

For this problem, it is convenient to store $\mathbf{\Sigma}_k \mathbf{B}_k^\intercal$ as $\tilde{\mathbf{B}}_k^\intercal$ to avoid redundant computations. Using this decomposition,

Eq. (2) is reorganized as follows:

$$\mathbf{W} \leftarrow \mathbf{W} \odot \left(\frac{\tilde{\mathbf{B}}_k \left(\mathbf{A}_k^\intercal \frac{\mathbf{X}}{(\mathbf{A}_k(\tilde{\mathbf{B}}_k^\intercal \mathbf{W}))} \right)}{\tilde{\mathbf{B}}_k(\mathbf{A}_k^\intercal \mathbb{1})} \right) \tag{4}$$

It is well-known that SVD itself is an expensive operation. The full solution of SVD is of order $O(\min\{FN^2, F^2N\})$ [5]. The reduced solution is also expensive when conventional techniques are used. To tackle this problem, we use the randomized SVD technique of Halko *et.al.* [6] which serves as a good example of randomized matrix factorization algorithms [6, 7]. This method considers $\mathbf{D}$ as a transformation matrix, generates random points and transforms them by $\mathbf{D}$, and it is based on the orthogonalization of the range of $\mathbf{D}$. Reduced SVD of a matrix of size $F \times N$ at rank-k takes $O\left((F+N)\,k\right)$ time. Note that reduced SVD is only done once at the training phase to obtain $\mathbf{A}_k$ and $\tilde{\mathbf{B}}_k$ and they are fixed at the test phase. The time complexity is $O\left((F+N)\,kT\right)$ when the operations are done as given in Eq. (4). Space complexity is $O\left((F+N)\,k\right)$ which is the total number of elements in matrices $\mathbf{A}_k$ and $\tilde{\mathbf{B}}_k$. Note that a similar approach is used in our recent work [11].

2.3. Improving efficiency via CUR

Another approach is to approximate the full matrix using its own rows and columns. This decomposition is known as CUR decomposition in the literature. We approximate the dictionary matrix as $\mathbf{D} \approx \mathbf{CUR}$ where $\mathbf{C}$, and $\mathbf{R}$ represent actual columns and rows of the matrix, respectively, and $\mathbf{U}$ is a linking matrix.

Let k_{col} and k_{row} be the number of the selected columns and rows, respectively. Then, the dimensions for $\mathbf{C}$, $\mathbf{U}$ and $\mathbf{R}$ will be $F \times k_{\text{col}}$, $k_{\text{col}} \times k_{\text{row}}$ and $k_{\text{row}} \times N$, respectively.

We first compute the probability of selecting the i^{th} row as proposed in [8] as follows:

$$\rho_i = \frac{1}{k} \sum_{j=1}^{k} a_{ij}^2\,, \quad i = 1, \ldots, F \tag{5}$$

where a_{ij} is the $(i,j)^{\text{th}}$ entry of $\mathbf{A}_k$, the matrix containing the left singular vectors. Similarly, the probability of selecting the j^{th} column is computed as follows:

$$\pi_j = \frac{1}{k} \sum_{i=1}^{k} b_{ji}^2\,, \quad j = 1, \ldots, N \tag{6}$$

where b_{ji} is the $(j,i)^{\text{th}}$ entry of $\mathbf{B}_k$, the matrix containing the right singular vectors.

Afterwards, k_{col} columns and k_{row} rows are randomly selected using a multinomial distribution with the computed probabilities. The relative error is ensured to be very small with a high probability as proved in [8] when sufficient number of dimensions are selected. Since k_{row} is relatively small compared to k_{col} in our case, let us define $\tilde{\mathbf{C}} = \mathbf{CU}$ and reorganize the update rule in Eq. (2) as follows:

$$\mathbf{W} \leftarrow \mathbf{W} \odot \left(\frac{\mathbf{R}^\intercal \left(\tilde{\mathbf{C}}^\intercal \frac{\mathbf{X}}{(\tilde{\mathbf{C}}(\tilde{\mathbf{R}}\mathbf{W}))} \right)}{\mathbf{R}^\intercal(\tilde{\mathbf{C}}^\intercal \mathbb{1})} \right) \tag{7}$$

The time complexity of the algorithm will be of order $O((F+N)\min\{k_{\text{row}}, k_{\text{col}}\}T)$. The minimum implies that one can merge $\mathbf{U}$ and $\mathbf{R}$ if k_{row} is larger than k_{col}. The memory requirement will be of order $O((F+N)\min\{k_{\text{row}}, k_{\text{col}}\})$ since we store both matrices.

2.4. Improving efficiency via selection of important data

The dictionary matrix might be a huge matrix possibly involving many redundant information in its columns and rows. So, instead of using low rank approximations of the full matrix, we extract only the important information. Firstly, we use only the selected columns of the spectrogram, that is, we only use $\mathbf{C}$:

$$\mathbf{W} \leftarrow \mathbf{W} \odot \left(\frac{\mathbf{C}^\intercal \frac{\mathbf{X}}{(\mathbf{CW})}}{\mathbf{C}^\intercal \mathbb{1}} \right) \tag{8}$$

Let us mention that Lee and Choi follow a similar approach to improve NMF in [9]. We go further and observe that there is no necessity to hold all of the rows which correspond to the frequency bands since the notes can be detected without hearing the higher harmonics. With this information, we skeletonize the dictionary matrix and keep only the points at the intersection of the selected rows and columns. Let $\check{\mathbf{D}}$ be the skeleton of $\mathbf{D}$, and let $\check{\mathbf{X}}$ involve the selected rows of the observed data. Then, the formula becomes:

$$\mathbf{W} \leftarrow \mathbf{W} \odot \left(\frac{\check{\mathbf{D}}^\intercal \frac{\check{\mathbf{X}}}{(\check{\mathbf{D}}\mathbf{W})}}{\check{\mathbf{D}}^\intercal \mathbb{1}} \right) \tag{9}$$

where the sizes of $\mathbf{W}$ and $\mathbb{1}$ are selected appropriately.

The time complexities will be of order $O(Fk_{\text{col}}T)$ and $O(k_{\text{col}}k_{\text{row}}T)$ for C-based and skeleton approaches, respectively. Since we discard most of the data and keep only the important part, the computational gain is very high in practice. The number of rows and columns that represent the variety in the data well enough may be hundreds of times less than the original dimensions of the full data. For a realistic case, if we have $F = 1025$, $N \approx 10^5$, and we choose $k_{\text{row}} = 400$ and $k_{\text{col}} = 4000$, the algorithm becomes nearly 75 times faster and we use only less than 2% of the data. This method can work in real time and handle big amount of data which can not be handled by conventional methods. Time and space complexities for the discussed methods are summarized in Table 1.

Table 1. Time and space complexities in Big-O notation

	Time	Space
Full linear model	$F\,N\,T$	$F\,N$
SVD-based	$(F+N)\,k\,T$	$(F+N)\,k$
CUR-based	$(F+N)\min\{k_{\text{row}},k_{\text{col}}\}T$	$(F+N)\min\{k_{\text{row}},k_{\text{col}}\}$
C-based	$F\,k_{\text{col}}\,T$	$F\,k_{\text{col}}$
Skeleton	$k_{\text{row}}\,k_{\text{col}}\,T$	$k_{\text{row}}\,k_{\text{col}}$

3. EXPERIMENTS AND RESULTS

In order to evaluate our methods, we have conducted several experiments. In our experiments, we have used the MAPS (MIDI Aligned Piano Sounds) data set [12]. The training set is obtained using 440 monophonic piano sounds of 44.1 kHz sampling rate where we represent the audio by its magnitude spectrogram which is computed via DFT. The spectrogram is formed using overlapping Hanning windows of dimension 2048 with a window shift 512. While constructing $D_i(f,\tau_i)$, we simply concatenated the spectra corresponding to the i^{th} note where $i = 1,\ldots,88$. About one third of the training data is removed due to low loudness and the obtained final dictionary matrix is 1025×115600 and is around 860 MB.

A test set is formed by choosing random sections from five different polyphonic pieces. The histogram of the polyphony orders (the number of notes playing simultaneously at a time instance) in the test data are given in Fig. 1 where it can be observed that the test data are predominantly polyphonic.

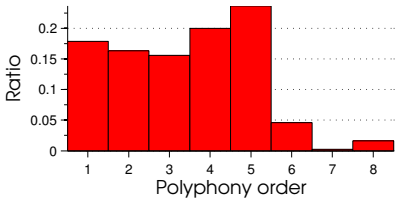

Fig. 1. Histogram of polyphony order: The polyphony in the test data is expressed mainly by the first 6 orders.

In order to evaluate the performance of the methods we used precision (fraction of retrieved instances that are relevant), recall (fraction of relevant instances that are retrieved) and f-measure $= \frac{2\times\text{precision}\times\text{recall}}{\text{precision}+\text{recall}}$ metrics.

We start with the full solution given in Eq. (2) and obtain an f-measure of 78.07%. This performance competes with the leading results obtained on MAPS (81% [12], 77% [1]). It should be noted that the method is a basic machine learning method and does not employ any advanced field-related signal processing techniques.

As all our methods are based on SVD, as the next step, we decompose $\mathbf{D}$ via randomized SVD. We observe that 98% of the total variance can be covered by using only the first 51 singular vectors. That is, there seems to be a high correlation in the data and thus, SVD is suitable for the problem.

As the first factorized method, the SVD-based approach given in Eq. (4) is applied with $k = 50$, 100, and 200 to see the effect of the used dimensions on the result. It is shown in Fig. 2 that we get better results with more dimensions. We get an f-measure value of 78.14% when we have 200 dimensions where the complexities are nearly 5 times better than the full solution.

Next, we compute the CUR decomposition of the dictionary. $k = 51$ dimensions are used in the computations of row selection probabilities ρ_i and column selection probabilities π_j. In this way, we aim to avoid selecting columns and rows leading to residual noise. We select 400 rows (frequency bands) and 4000 columns (samples), and we apply the CUR-based approach given in Eq. (7). As shown in Fig. 2, the results seem to be similar to the SVD-based approach.

Finally, we select only the important parts of $\mathbf{D}$ with the computed indices of the CUR decomposition above. The results for the C-based and skeleton-based approaches are also given in Fig. 2. We only use less than 2% of the data after skeletonization, the speed is nearly 75 times faster, and we get an f-measure of 77.17% which is very close to the result of the full solution.

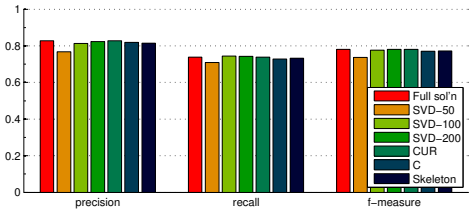

Fig. 2. Results obtained on the test set per approach: It is seen that the result is better preserved by using more dimensions in SVD-based approach where using 200 dimensions is enough to get the original results. CUR seems to be an alternative to SVD. C-based approach and skeletonizing the matrix give promising results.

In addition to the overall results, we give the results for each polyphony order in Fig. 3 for the skeleton-based approach. We got similar results for the other approaches, so we do not provide separate plots for each. As can be seen, recall rate is perfect in the monophonic case but the precision is low. Using a higher threshold on the weights may lead to selection of fewer notes as active and this can improve precision with a trade-off in recall rate if one is more interested in precision. The values used here are obtained by optimizing the threshold on a verification set which excludes the test data.

As post-processing, we apply a median filter on each row of $\mathbf{W}$ before thresholding. Filtering leads to an increase around 3-6% in f-measure for all approaches.

We also analyze the effect of the selected number of columns and rows on the results and give it in Fig. 4 for CUR-based and skeleton-based approaches. As it is seen,

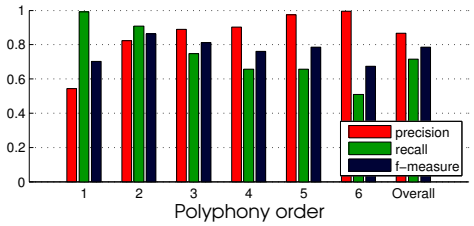

Fig. 3. Precision, recall and f-measures for each polyphony order: The figure shows the results for the skeleton approach. F-measure seems to be more than 65% for each polyphony order.

f-measure becomes stabler after a few hundreds of rows and a few thousands of columns. So, there is no need to keep all of the data since it holds lots of repetitive information. Note that the results are obtained on a separate validation test excluding the test samples. The fluctuations in the plots stem from randomization which is used for selection.

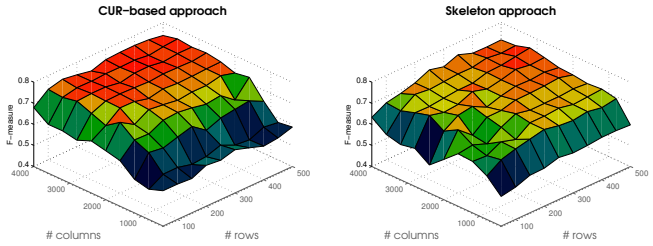

Fig. 4. f-measures versus the number of columns (samples) and rows (frequency bins) in CUR-based approach (left) and skeleton approach (right): It is seen that only a few hundred frequency bins and a few thousand samples are necessary to keep success ratio of the algorithm.

4. CONCLUSIONS

We have addressed the problem of polyphonic music transcription and discussed that the conventional methods are inefficient to handle big data. We show that even a standard matrix factorization model is prohibitive in real applications where a huge amount of training data is used. The update rules are made efficient by the use of randomized matrix decompositions. Without compromising the success rate, time and space complexities are improved such that the proposed method can work in real time. A high f-measure value ($\sim$78%) is obtained on polyphonic recordings by using only a few hundred frequency bins and sample columns out of a huge dictionary matrix. We get at least 65% f-measure on each polyphony order which shows that the proposed method is stable and consistent. The reported results show that the method can compete with the leading methods in the literature. With abundance of data in different applications, randomized matrix decompositions are likely to get more attention in the future and we have illustrated a relevant application of this in the context of music transcription.

5. REFERENCES

[1] Anssi Klapuri and Manuel Davy, *Signal Processing Methods for Music Transcription*, Springer, 2006.

[2] P. Smaragdis and J. C. Brown, "Non-negative matrix factorization for polyphonic music transcription," in *IEEE WASPAA*, 2003, pp. 177–180.

[3] P. H. Peeling and Godsill S. J. Cemgil, A. T., "Generative spectrogram factorization models for polyphonic piano transcription," *IEEE TASLP*, vol. 18, no. 3, pp. 519–527, 2010.

[4] P. Smaragdis, "Polyphonic pitch tracking by example," in *IEEE WASPAA*, 2011, pp. 125–128.

[5] G. H. Golub and C. F. Van Loan, *Matrix computations*, The Johns Hopkins University Press, 3rd edition, 1996.

[6] N. Halko, P. G. Martinsson, and J. A. Tropp, "Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions," *SIAM Review*, 2011.

[7] M. W. Mahoney, "Randomized Algorithms for Matrices and Data," *Foundations and Trends in Machine Learning*, pp. 123–234, 2011.

[8] M. W. Mahoney and P. Drineas, "CUR matrix decompositions for improved data analysis," *Proc. of the National Acad. of Sci.*, vol. 106, no. 3, pp. 697–702, 2009.

[9] H. Lee and S. Choi, "CUR+NMF for learning spectral features from large data matrix," in *IEEE Int'l Joint Conf. on Neural Networks*, 2008, pp. 1592–1597.

[10] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, pp. 788–791, 1999.

[11] İ. Arı, U. Şimşekli, A. T. Cemgil, and L. Akarun, "SVD-based Polyphonic Music Transcription," in *IEEE Signal Proc. and Communications Applications Conf.*, 2012.

[12] V. Emiya, R. Badeau, and B. David, "Multipitch Estimation of Piano Sounds Using a New Probabilistic Spectral Smoothness Principle," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1643–1654, 2010.