university of groningen / faculty of science and engineering

Automatic Drum Transcription Using Template-Initialized Variants of Non-negative Matrix Factorization

Bachelor's Project Thesis

Júlia Vághy, S3994759, j.vaghy@student.rug.nl,
Supervisors: dr. C.P. Lawrence and prof. dr. H. Jaeger

Abstract: Low-quality acoustic drum datasets limit the applicability of automatic drum transcription (ADT) systems; however, this may be circumvented by template initialization in cases where the drummer can provide sound samples of their specific drum kit. To explore performance in these scenarios, the present project assesses the accuracy of different template-initialized non-negative matrix factorization (NMF) variants on the ADT task in the presence of background noise. Performance is evaluated for NMFD, the deconvolutional variant with 2-dimensional spectrotemporal templates, and NMF, the original variant with 1-dimensional spectral templates. Three template adaptivity conditions are considered: adaptive, semi-adaptive, and fixed. Furthermore, the effect of additional noise template components is explored. Performance is evaluated on a synthetic dataset containing 20 drum loops and corresponding instrument samples, each on four noise conditions: none, mild, loud, and extreme. Initialization templates are derived from the drum loop's respective sample sounds on the given noise condition. The results suggest that adaptive NMFD is best suited for the task with F = 0.83 ± 0.13, 0.83 ± 0.1, 0.74 ± 0.11, and 0.69 ± 0.14, on the four noise conditions, respectively. Additional noise template components do not lead to significant performance improvement in either of the conditions.

1 Introduction

The present project explores automatic drum transcription (ADT) for drum-only recordings in the presence of background noise. Such a scenario is especially relevant for home musicians with acoustic drum kits in the context of music production and music education (Wu et al., 2018).

Firstly, producing a professional drum recording using a non-electronic drum kit requires a studio room with suitable acoustics and multiple sophisticated microphones. ADT has the potential to circumvent this need by converting the audio recording from a single microphone into a MIDI file containing the drum loop. The musician can then import the MIDI file into their Digital Audio Workstation (DAW) of choice. Here, they can apply instruments and effects to produce a professional drum recording with a natural feel. Secondly, people learning to play drums could also benefit. If one wants to master a given drum loop, ADT can be used to match what they play against the original loop and provide real-time performance assessment.

However, the quality of today's drum transcription algorithms is not high enough to be used in such scenarios. As pointed out by Wu et al. (2018) and Cartwright & Bello (2018), a major problem in drum transcription is insufficient datasets of annotated real-world drum recordings, which severely limits learning algorithms that require large amounts of training data.

To overcome the lack of data, variants of the non-negative matrix factorization (NMF) algorithm can be initialized with templates derived from the sounds of the user's specific drum kit, as suggested by Battenberg et al. (2012). In application scenarios such as music education and music production, asking the user to provide samples of each drum sound before transcription is a viable option.

Most research on NMF for ADT uses generic templates of bass drum, snare drum, and hi-hat

(Cartwright & Bello, 2018). Thus, there is no conclusion on which NMF variant works best for a broader range of instruments and in scenarios where the templates are initialized using the user's specific drum kit. Moreover, the effect of background noise has not been explored. Therefore, the present project aims to optimize and compare different variants of the NMF algorithm in this scenario.

It is assumed that no instruments other than drums and noise sources are present in the recording and that drum sounds can be superposed, i.e., hitting multiple drums simultaneously. Performance is evaluated in the presence of different types and levels of background noise.

The implementation, the results, and the presentation slides can be found in the GitHub repository github.com/vaghyjuli/nmf_drums, and the data in the Google Drive folder drive.google.com/drive/folders/1FK_LFWgXNtSggJ_s1mbPQEojCsYjLf8f.

1.1 State of the art

ADT research can be partitioned into several subfields, as stated by Wu et al. (2018). The most relevant to the present project are Drum Transcription of Drum-Only Recordings (DTD) and Drum Transcription in the Presence of Melodic Instruments (DTM). Drum transcription in the presence of background noise has not yet been investigated, despite being highly relevant in the previously mentioned application scenarios.

Gillet & Richard (2008) categorize early ADT algorithms into three classes. Firstly, *segment and classify* approaches perform onset detection to segment the signal into events, then apply pattern recognition to the events. Secondly, *separate and detect* approaches first separate the signal into streams such that each stream consists only of activations of a given kit piece, then detect onset candidates in each individual stream. Finally, *match and adapt* approaches use drum templates that are iteratively matched to events and adapted to the detected occurrences (Gillet & Richard, 2008). Later on, Paulus & Klapuri (2009) also modeled temporal connections between events via hidden Markov models (HMMs), and more recently, the field of ADT has been shifting towards deep learning solutions.

The most successful approaches include support vector machines with F-measure F=.83 (Gillet & Richard, 2004), HMMs with F=.82 (Paulus & Klapuri, 2009), probabilistic latent component analysis with F=.7 (Benetos et al., 2014), non-negative matrix factorization (NMF) with F=.89 (Wu & Lerch, 2015), and neural network-based approaches, such as convolutional neural networks with F=.96 (Gajhede et al., 2016) and RNNs with F=.95 (Southall et al., 2017).

Note that the above algorithms were trained and evaluated on the same benchmark datasets, which are limited in a variety of aspects, as explained in Section 1.2. This suggests that algorithms trained on the said datasets would not reach the same accuracies in real life, especially in the context of music education and production.

1.2 The data problem

According to Cartwright & Bello (2018), a severe problem in the field of ADT is that the majority of research has been limited to bass drum, snare drum, and hi-hat, as these are the instruments that have enough coverage in public datasets. The most commonly used datasets are the IDMT-SMT-Drums (Dittmar & Gärtner, 2014) and the ENST (Gillet & Richard, 2006) datasets. Wu et al. (2018) point out that in addition to the insufficient instrument coverage, diversity of playing style is lacking, most data comes from professional studio recordings with homogeneous conditions, and due to the limits of temporal accuracy in human labelling, most transcribed drum loops are overly simple (Wu et al., 2018).

The insufficiency of datasets is a severe problem for the applicability of ADT algorithms. For example, the lack of playing style diversity may be a drawback for algorithms incorporating temporal relations between events such as RNNs or HMMs. Moreover, algorithms trained and optimized using drum data recorded in homogeneous studio conditions may exhibit worse performance in different recording condition, such as a more realistic scenario with background noise. Most importantly, algorithms that need training or generic initialization data to distinguish drum kit pieces are limited to the three instruments with sufficient coverage in the datasets (Wu et al., 2018; Cartwright & Bello, 2018).

However, in application scenarios such as music education and music production, the user can be asked to provide a sample of each drum sound they would like to distinguish in a transcription. Such sound samples can be transformed into templates to initialize variants of the NMF algorithm (Battenberg et al., 2012). NMF with interactive initialization may therefore have the potential to overcome the lack of data and create ADT systems with real-life applicability, which is what the present project aims to explore.

1.3 Short-time Fourier transforms

Before diving into the formal definition of NMF and its variants, let us understand the idea behind short-time Fourier transforms, which gives the basis of the features used in NMF. The description to follow is based on Müller (2021), van Netten et al. (2020), and van Rij (2020).

The analogue acoustic waves created by drumming and background noise sources can be converted to a digital discretized time-domain representation by sampling air pressure at equally spaced points in time via a microphone, yielding a discrete-time signal $\boldsymbol{x} \in \mathbb{R}^L$, where L is the number of sampled sound pressure points. The sampling frequency F_s, given in Hz, denotes the number of samples taken each second. As such, a recording of t seconds yields $L = t \cdot \frac{1}{F_s}$

As explained in Appendix B, one of the first steps of human sound perception is to decompose the acoustic waveform into its constituent frequencies. In the digital representation, mapping from time to frequency domain can be achieved via a discrete Fourier transform (DFT) with K frequency coefficients. For a discrete-time signal $\boldsymbol{x} \in \mathbb{R}^L$, the kth frequency coefficient $\boldsymbol{y}(k) \in \mathbb{C}$ for $k \in [1 : K]$ of the discrete Fourier transform (DFT) $\boldsymbol{y} \in \mathbb{C}^K$ is given by

$$\boldsymbol{y}(k) = \sum_{l=1}^{L} \boldsymbol{x}(l) e^{-j\frac{2\pi}{L}lk}, \tag{1.1}$$

where $\boldsymbol{x}(l)$ is the time domain signal's value at time step $l \in [1 : L]$. The DFT can be efficiently computed from the time domain representation using the fast Fourier transform (FFT) algorithm, popularized by Cooley & Tukey (1965).

However, a pure frequency domain representation is insufficient for signals whose frequency characteristics change over time, such as a drum recording. Time-frequency representation can be obtained using a short-time Fourier transform (STFT). The method takes signal sections of length $\nu \in \mathbb{N}$ and calculates the DFT of each section using FFT. The hop size defines the numbers of time frames between the starting frames of subsequent windows. It is often defined as $\frac{\nu}{2}$. The kth spectral coefficient of the localised DFT starting at time point m is given by

$$\mathbf{X}(n,k) = \sum_{m=0}^{\nu-1} \boldsymbol{x}(n+m) e^{-j\frac{2\pi}{\nu}mk}, \tag{1.2}$$

for $n \in [1 : N]$, $k \in [1 : K]$, where $\nu \in \mathbb{N}$ is the STFT window size, $\mathbf{X} \in \mathbb{C}^{N \times K}$, and $\boldsymbol{x}(n+m) \in \mathbb{R}$ is the time domain signal's value at time step $n+m$. The Nyquist limit $F_{\text{Nyquist}} = \frac{F_s}{2}$ is the maximal frequency that can be measured, therefore, the obtained Fourier coefficients will correspond to the intensities of equally sized frequency bands in the range $[0 : \frac{F_s}{2}]$.

A magnitude STFT $||\mathbf{X}|| \in \mathbb{R}^{N \times K}$ can be obtained by taking magnitudes of the complex entries in $\mathbf{X}$, and thereby a time-frequency representation in the real domain is obtained. In addition, Müller (2021) suggests logarithmically compressing the magnitude STFT in order to accentuate onsets with lower striking velocity. As such, entries in the final matrix representation $\mathbf{Y} \in \mathbb{R}^{K \times N}$ are given by

$$\mathbf{Y}(n,k) = \log(1 + \alpha||\mathbf{X}(n,k)||), \tag{1.3}$$

for $\forall n \in [1 : N], \forall k \in [1 : K]$, where $\alpha \in \mathbb{R}_{>0}$ is a hyperparameter driving the degree of logarithmic compression.

There is a critical trade-off when choosing the window size ν of the STFT. A narrow window leads to good time resolution; however, it prevents the algorithm from identifying low frequencies and may create artifacts in the high-frequency range. On the other hand, a wider window gives poor time resolution and is more computationally expensive but covers a larger range of frequencies.

The obtained (logarithmically compressed) magnitude STFT can be visually represented as a spectrogram, essentially, a heatmap of the entries in $\mathbf{Y} \in \mathbb{R}^{K \times N}$.

1.4 NMF algorithms in ADT

NMF is an originally unsupervised source separation algorithm aimed at decomposing a signal into its constituent sources and their respective time-varying gains. The composite signal is represented by a non-negative matrix $\mathbf{V} \in \mathbb{R}_{\geq 0}^{K \times N}$, and the goal of NMF is to rank factorize the matrix $\mathbf{V}$ into two non-negative matrices: a template matrix $\mathbf{W} \in \mathbb{R}_{\geq 0}^{K \times R}$ and an activation matrix $\mathbf{H} \in \mathbb{R}_{\geq 0}^{R \times N}$. The rank of the factorization $R < K, N$ is predefined and corresponds to the expected number of sources in the composite signal. After factorization, each template matrix column $\mathbf{W}(, r) \in \mathbb{R}_{\geq 0}^{K}$ is expected to contain the template of the rth source for $r \in [1 : R]$, and each activation matrix row $\mathbf{H}(r,) \in \mathbb{R}_{\geq 0}^{N}$ should contain the activation of the rth source on N time steps.

1.4.1 NMF features in ADT

For the drum transcription task, this definition narrows down to decomposing the magnitude STFT of an audio recording of drumming into its constituent instrumental source templates and their respective time-varying activations. In its elementary implementation, NMF is an unsupervised method, however, in ADT, the template matrix $\mathbf{W} \in \mathbb{R}_{\geq 0}^{K \times R}$ is often initialized with generic drum templates and is thereby better described as semi-supervised, as explained in Section 1.4.3.

For the drum transcription task, $\mathbf{V} \in \mathbb{R}^{K \times N}$ is the magnitude STFT derived from the recording using K spectral coefficients and N time windows, as explained in Section 1.3. Then, each entry $\mathbf{V}(k, n) \in \mathbb{R}$ represents the intensity of the kth spectral coefficient for $k \in [1 : K]$ in the nth time window for $n \in [1 : N]$. A spectral band's intensity cannot be less than zero, thus, $\mathbf{V} \in \mathbb{R}_{\geq 0}^{K \times N}$.

In the elementary implementation, the rank of factorization R is defined as the number of drum instruments present in the audio recording. Suppose for now that $\mathbf{V}$ represents a drum-only recordings without background noise. Then, with the ideal factorization, the template matrix $\mathbf{W} \in \mathbb{R}^{K \times R}$ is expected to contain spectral templates for each drum sound as column vectors. As such, $\mathbf{W}(k, r)$ denotes the prototype intensity of the rth source for $r \in [1 : R]$ in the kth spectral band for $k \in [1 : K]$. As negative spectral intensities cannot occur in a magnitude STFT, $\mathbf{W} \in \mathbb{R}_{\geq 0}^{K \times R}$ must by nature be non-negative.

As the audio recording is made up of R sound sources with varying intensity over time, the original time-frequency matrix $\mathbf{V} \in \mathbb{R}_{\geq 0}^{K \times N}$ can be thought of as a superposition of instrument template columns $\mathbf{W}(, r)$ weighted at each time step $n \in [1 : N]$ by their respective time-dependent activation. As such, each row $\mathbf{H}(r,) \in \mathbb{R}^{N}$ of the activation matrix $\mathbf{H} \in \mathbb{R}^{R \times N}$ should represent the rth drum instrument's activation over the N time windows of the original time-frequency matrix $\mathbf{V} \in \mathbb{R}_{\geq 0}^{K \times N}$. The activation of a source cannot be less than zero, thus, $\mathbf{H} \in \mathbb{R}_{\geq 0}^{R \times N}$.

1.4.2 Elementary update rules

NMF algorithms find a template matrix $\mathbf{W} \in \mathbb{R}_{\geq 0}^{K \times R}$ and an activation matrix $\mathbf{H} \in \mathbb{R}_{\geq 0}^{R \times N}$ via gradient descent, such that $\mathbf{V} \approx \hat{\mathbf{V}} = \mathbf{W} \cdot \mathbf{H}$, where $\hat{\mathbf{V}}$ is the low-rank approximation of $\mathbf{V}$. In the elementary unsupervised implementation of NMF, both the template matrix $\mathbf{W}$ and the activation matrix $\mathbf{H}$ are initialized uniformly or randomly (Müller, 2021).

The standard NMF update rules use the cost function $\mathcal{L}$ defined by Lee & Seung (2000) as

$$\mathcal{L}(\mathbf{V}|\hat{\mathbf{V}}) = \sum \left(\mathbf{V} \odot \log \left(\frac{\mathbf{V}}{\hat{\mathbf{V}}} \right) - \mathbf{V} + \hat{\mathbf{V}} \right), \quad (1.4)$$

where $\odot$ denotes the Hadamard (element-wise) product, division is also element-wise, and the sum is computed over all entries of the resulting matrix.

Lee & Seung (2000) suggest that the matrices $\mathbf{W}$ and $\mathbf{H}$ are updated iteratively instead of jointly for the sake of computational efficiency. At a given gradient descent step, the cost function $\mathcal{L}$ exhibits the steepest decrease from the points $\mathbf{H}$ and $\mathbf{W}$ in the direction of the negative gradients $-\nabla\mathcal{L}(\mathbf{H})$ and $-\nabla\mathcal{L}(\mathbf{W})$, respectively. Therefore, the matrices should be updated as

$$\mathbf{H} \leftarrow \mathbf{H} - \gamma_{\mathbf{H}} \nabla\mathcal{L}(\mathbf{H}) \quad (1.5)$$

$$\mathbf{W} \leftarrow \mathbf{W} - \gamma_{\mathbf{W}} \nabla\mathcal{L}(\mathbf{W}), \quad (1.6)$$

where $\gamma_{\mathbf{W}}$ and $\gamma_{\mathbf{H}}$ are the step size parameters.

However, the update rules in Equations 1.5 and 1.6 may result in negative entries in $\mathbf{W}$ and $\mathbf{H}$ which would obstruct correspondence to real source templates and time-dependent activations. To enforce

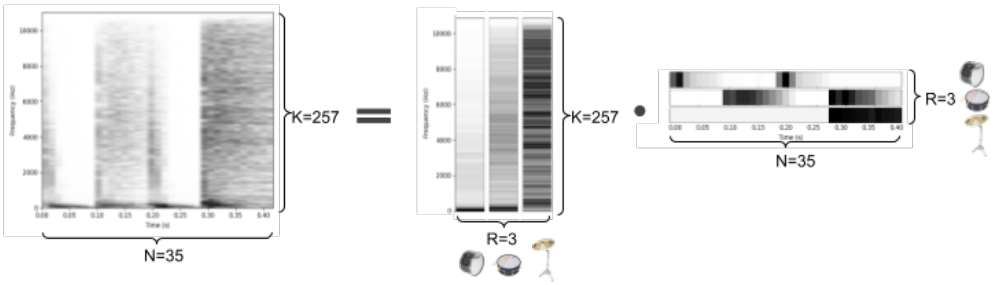

Figure 1.1: Illustration of non-negative matrix factorization. $\hat{\mathbf{V}} = \mathbf{W} \cdot \mathbf{H}$.

non-negativity without the introduction of additional constraints Lee & Seung (2000) propose setting the step size parameters $\gamma_{\mathbf{W}}$ and $\gamma_{\mathbf{H}}$ to

$$\gamma_{\mathbf{H}} := \frac{\mathbf{H}}{\mathbf{W}^\intercal \mathbf{W}\mathbf{H}} \tag{1.7}$$

$$\gamma_{\mathbf{W}} := \frac{\mathbf{W}}{\mathbf{W}\mathbf{H}\mathbf{H}^\intercal}, \tag{1.8}$$

which results in the final update rules

$$\mathbf{W} \leftarrow \mathbf{W} \odot \frac{\frac{\mathbf{V}}{\hat{\mathbf{V}}}\mathbf{H}^\intercal}{\mathbf{J}\mathbf{H}^\intercal} \tag{1.9}$$

$$\mathbf{H} \leftarrow \mathbf{H} \odot \frac{\mathbf{W}^\intercal \frac{\mathbf{V}}{\hat{\mathbf{V}}}}{\mathbf{W}^\intercal \mathbf{J}}, \tag{1.10}$$

where $\odot$ again denotes the Hadamard (element-wise) product, division is also element-wise, and $\mathbf{J} \in \mathbb{R}^{K \times N}$ is a matrix of ones. As the initialization entries of the template matrix $\mathbf{W}$ and the activation matrix $\mathbf{H}$ are non-negative, this property is preserved by the multiplicative update rules.

Multiplicative update rules with a non-decreasing learning rate may raise questions about convergence, however, Lee & Seung (2000) point out that the loss function $\mathcal{L}$ is non-increasing under the update rules.

Additionally, as specified by Müller (2021), one can impose stop criteria such as stopping at an iteration limit $I \in \mathbb{N}$, or if the (element-wise) distance between two subsequently computed matrices is below some threshold $\epsilon \in \mathbb{R}_{>0}$.

1.4.3 Initialization

It is important to note, that having accurate information on the number of sources and the rank of factorization R is crucial to obtain a meaningful solution. Moreover, there are many different factorizations mathematically possible, thus, given a complex enough drum recording, factorization using an uninformed intialization of the template matrix $\mathbf{W}$ is unlikely to lead to a meaningful solution (Müller, 2021). As a result, the field of ADT adopted semi-supervised NMF, whereby the columns of $\mathbf{W}$ are initialized with the generic templates of drum instruments, generally bass drum, snare drum, and hi-hat (Wu et al., 2018). On the other hand, Battenberg et al. (2012) suggest a more flexible solution, initializing with the R sounds of the specific drum kit used in the recording to be transcribed, which is what the present project is concerned with.

As such, R is defined as the number of instruments, and templates are obtained by taking the average magnitude spectra of isolated drum hit sound samples, yielding $\mathbf{W}(, r) \in \mathbb{R}^K$ for $r \in [1 : R]$, where K is the number of spectral bands.

Once $\mathbf{W}$ is initialized, one option is to run the algorithm as explained before, leaving the algorithm in its fully **adaptive** form. Another option is to fix the template matrix $\mathbf{W}$ and only update the activation matrix $\mathbf{H}$, which defines the **fixed** variant (Wu et al., 2018).

1.4.4 Semi-adaptive NMF

To account for the variability of drum sounds and differences from the generic templates, Dittmar & Gärtner (2014) proposed the semi-adaptive NMF (SANMF) method for the DTD task. The method initially pushes the template matrix $\mathbf{W}$ toward the

template sounds and allows more deviation as the iteration limit $I \in \mathbb{N}$ is approached. The SANMF update rule for $\mathbf{W}$ thereby becomes

$$\mathbf{W} \leftarrow (1-\alpha) \cdot \mathbf{W}^{(0)} + \alpha \cdot \mathbf{W} \odot \frac{\frac{\mathbf{V}}{\hat{\mathbf{V}}}\mathbf{H}^\intercal}{\mathbf{J}\mathbf{H}^\intercal}, \qquad (1.11)$$

where $\alpha = (\frac{i}{I})^\beta$, $i \in [1:I]$ is the iteration count, $I \in \mathbb{N}$ is the iteration limit, and $\beta \in \mathbb{R}$ is a hyperparameter governing the rate of divergence from the initial template matrix $\mathbf{W}^{(0)}$ and thereby the degree of adaptivity. The update rules for $\mathbf{H}$ remain as defined in Equation 1.10.

Let us take a closer look at the hyperparameter β. If $\beta = 0$, then $\alpha = (\frac{k}{K})^0 = 1$, thus, the $(1-\alpha) \cdot \mathbf{W}^{(0)}$ term in Equation 1.11 cancels out, which makes the update rule equivalent to that of the adaptive variant, as defined in Equation 1.9. On the other hand, as $\beta \to \infty$, $\alpha \to 0$. As such, the second term of Equation 1.11 cancels out, resulting in no updates, which corresponds to the fixed variant. In essence, the larger the value of the hyperparameter β, the less adaptive the initialized templates are.

Dittmar & Gärtner (2014) found no significant difference between the performance of the semi-adaptive and fixed NMF variants on the DTD task with initialization using generic templates, and that the adaptive NMF variant leads to worse performance. Regardless, the present task may justify the use of semi-adaptivity. In particular, since the templates are mixtures of drum hits and background noise, it might be beneficial to allow the algorithm to filter the spectral components of noise in order to avoid spurious activations.

1.4.5 Partially-fixed NMF

Wu & Lerch (2015) introduced the partially-fixed NMF (PFNMF) variant for the DTM task. This form of the algorithm is intended to also pick up the spectral templates and corresponding activations of harmonic instrumental components with previously unknown spectral characteristics. As such, PFNMF includes Q additional randomly initialised columns in the template matrix $\mathbf{W} \in \mathbb{R}_{\geq 0}^{K \times (R+Q)}$ and corresponding rows in the activation matrix $\mathbf{H} \in \mathbb{R}_{\geq 0}^{(R+Q) \times N}$, which are expected to pick up on harmonic components. The drum template-initialized columns in $\mathbf{W}$ are fixed during gradient descent, the additional randomly initialized harmonic columns remain adaptive as defined in Equation 1.9, and both the randomly initialized harmonic and drum template-initialized components in $\mathbf{H}$ are updated normally using Equation 1.10.

Additional template components may also prove to be useful in picking up background noise, instead of harmonic instrumental components as implemented by Wu & Lerch (2015), therefore, this option is also explored.

1.4.6 NMFD

Non-negative matrix factor deconvolution (NMFD) is the convolutive variant of the NMF algorithm, introduced by Smaragdis (2004). NMFD uses the full spectrotemporal pattern of each drum sound's magnitude STFT as templates. The goal is to model the magnitude spectrogram of the recording $\mathbf{V} \in \mathbb{R}^{K \times N}$ using R spectrotemporal patterns $\mathbf{P}(,r,) \in \mathbb{R}^{K \times T}$, where again, R generally corresponds to the number of drum instruments present in the recording. As such, the method has the power to capture more complex time varying characteristics of drum events.

The set of R patterns can be grouped into a pattern tensor $\mathbf{P} \in \mathbb{R}_{\geq 0}^{K \times R \times T}$. Patterns with less frames in their magnitude STFT are zero-padded to have the same length T as the pattern with maximum length. Then, the convolutive approximation $\hat{\mathbf{V}}$ of the original magnitude STFT $\mathbf{V}$ of the drum loop is given by

$$\hat{\mathbf{V}} = \sum_{t=0}^{T-1} \mathbf{P}(,,t) \cdot \overset{t\rightarrow}{\mathbf{H}}, \qquad (1.12)$$

where $\mathbf{P}(,,t)$ refers to the tth time frame in all patterns, and $\overset{t\rightarrow}{(.)}$ denotes a frame shift operator, as defined by Smaragdis (2004), shifting a matrix by t columns to the right. For example,

$$\mathbf{A} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \end{bmatrix}, \overset{2\rightarrow}{\mathbf{A}} = \begin{bmatrix} 0 & 0 & a & b \\ 0 & 0 & e & f \end{bmatrix}. \qquad (1.13)$$

In the same way, $\overset{\leftarrow t}{(.)}$ would shift a matrix by t columns to the left.

Smaragdis (2004) adapted the multiplicative update rules formulated for NMF by Lee & Seung

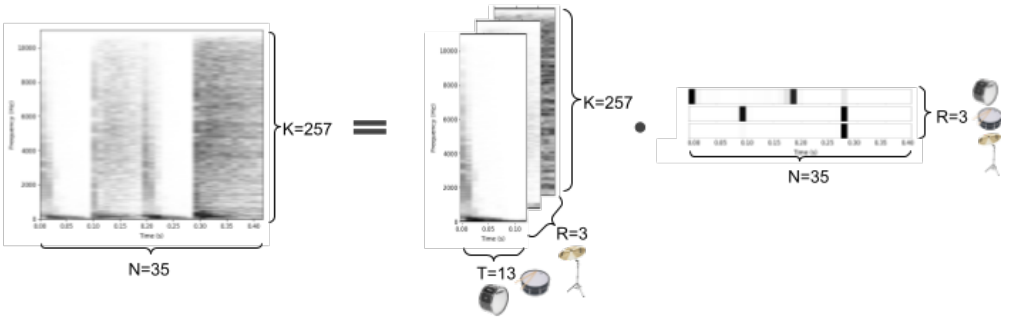

Figure 1.2: Illustration of non-negative matrix factorization deconvolution. $\hat{\mathbf{V}} = \sum_{t=0}^{T-1} \mathbf{P}(,,t) \cdot \overset{t\rightarrow}{\mathbf{H}}$.

(2000) as

$$\mathbf{P}(,,t) \leftarrow \mathbf{P}(,,t) \odot \frac{\frac{\mathbf{V}}{\hat{\mathbf{V}}} \cdot \left(\overset{t\rightarrow}{\mathbf{H}}\right)^{\intercal}}{\mathbf{J} \cdot \left(\overset{t\rightarrow}{\mathbf{H}}\right)^{\intercal}} \tag{1.14}$$

$$\mathbf{H} \leftarrow \mathbf{H} \odot \frac{\mathbf{P}(,,t)^{\intercal} \cdot \left[\frac{\mathbf{V}}{\hat{\mathbf{V}}}\right]^{\leftarrow t}}{\mathbf{P}(,,t)^{\intercal} \cdot \mathbf{J}}, \tag{1.15}$$

for $\forall t \in [1:T]$, where T is the number of frames in the template patterns, $\mathbf{V} \in \mathbb{R}^{K \times N}$ is the drum loop recording's magnitude STFT, $\hat{\mathbf{V}} \in \mathbb{R}^{K \times N}$ is its convolutive approximation as defined in Equation 1.12, $\mathbf{H} \in \mathbb{R}_{\geq 0}^{R \times N}$ is the activation matrix, $\mathbf{P} \in \mathbb{R}_{\geq 0}^{K \times R \times T}$ is the pattern tensor, $\odot$ denotes the Hadamard product, division is element-wise, and $\mathbf{J} \in \mathbb{R}^{K \times N}$ is a matrix of ones.

1.4.7 Conclusion

The following unique features can be extracted from the previously researched template-initialized NMF variants in ADT:

- Template dimensionality (NMF, NMFD),
- Adaptivity of template matrix/pattern tensor (adaptive, semi-adaptive, fixed), and
- Additional (noise) template components.

The present project evaluates all combinations of the above features on the drum transcription task in the presence of background noise with specific template initialization.

2 Methodology

2.1 Data

The goal of data synthesis was to obtain a dataset simulating the scenario of a home musician initializing the templates with their own drum kit then recording a drum loop. As such, data is made up of drum loop recordings, sets of isolated drum hit recordings of the drum instruments present in each drum loop, and expected transcription results. The noising of the data builds on the assumption that instrument sound samples and drum loops are both recorded in the same background noise condition. This entails having the same noise sources, however, spectral characteristics of a given source may vary over time.

MIDI samples For an explanation of the MIDI file format, refer to Appendix C. The drum loop data builds on MIDI snippets from the Groove MIDI dataset created by Gillick et al. (2019). The Groove MIDI dataset contains MIDI drum loops recorded on electronic drum kits, played by 10 different drummers. Thereby, it covers various playing styles and genres, such as rock, funk, jazz, latin, and blues. The striking velocity v varies and is encoded in the MIDI information. The present project uses 20 extracted snippets of two bar length each, approximately two from each drummer while maximizing the number of genres. The key signature of each snippet is $\frac{4}{4}$ with BPM $\in [59:152]$ with mean $\mu_{BPM} = 104.9$. The extracted snippets contain 731 drum strokes in total with striking velocity $v \in [6:127]$ with mean $\mu_v = 73.86$. The Python library mido was used to extract annotations for

each drum loop from the corresponding MIDI files, as explained in Appendix C.

WAV samples Each MIDI snippet was rendered into a WAV file using the Ableton digital audio workstation's Core 505, 606, or 707 drum kits. The number of recordings for each drum kit is approximately equal, and 11 ± 1 drum sounds are included from each kit, giving a total of 33 drum sounds in the dataset. Besides the drum loops, WAV template recordings of isolated drum hits for each of the 33 instruments were extracted with Ableton, using striking velocity $v = 100$. All WAV samples are mono recordings with sample rate $F_s = 22050$ Hz, synthesized with a bit depth of 16 and the triangular dithering settings of Ableton.

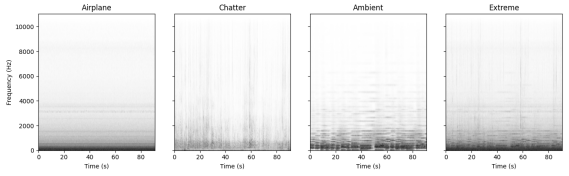

Figure 2.1: Logarithmically compressed magnitude spectrograms of the airplane, chatter, and ambient noise audio files on the loud condition, and their superposition, respectively. The superposition defines the extreme condition.

Noise To simulate realistic recording conditions, the drum loop and template recordings are superposed with noise audio. Three sources of background noise are used with two loudness conditions, namely **mild** and **loud**. The *airplane* background noise simulates mechanical noise and has an almost perfectly time-invariant spectrum. The *chatter* noise simulates people talking in the background, while the *ambient* noise is mild, melodic music without percussive components. The chatter and ambient noise conditions have highly time-variant spectra. Each noise audio file is 90 seconds long. In addition, an **extreme** noise condition is added, which is the superposition of the three noise types on the loud condition. For reference, a **none** noise condition is included which entails no added noise.

The type (airplane, chatter, ambient) and loudness (none, mild, loud, extreme) is invariant in a single transcription experiment. As such, before transcription begins, the drum loop and instrument templates are noised by superposing their original recordings with random snippets of the respective 90-second noise audio file.

Each NMF variant is evaluated on all 20 drum loop recordings in all noise conditions, namely none, mild airplane, mild chatter, mild ambient, loud airplane, loud chatter, loud ambient, and extreme. Performance on the mild and loud conditions is averaged over the airplane, chatter, and ambient conditions.

2.2 Features and template initialization

The discrete time-domain signal $\boldsymbol{x}$ with a length of t second is obtained from each WAV file and is converted to a discrete time-frequency representation using the librosa's (McFee et al., 2022) STFT functionality with window size 512 and hop size 256, yielding $\mathbf{X} \in \mathbb{C}^{K \times N}$, where K $= 257$ and $N = t \cdot \frac{F_s}{\text{hop}} = t \cdot \frac{22050}{256}$. Due to the Nyquist limit of $\frac{F_s}{2}$, the frequency bands cover the frequency range $[0, 11025]$. The window size corresponds to 23.2 ms, and a hop size to 11.6 ms. The latter is ideal because according to Wu et al. (2018), humans identify two sounds as separate if they at least are 8-10 ms apart. A preliminary mock experiment with window sizes $\nu \in 256, 512, 1024, 2048$ and hop sizes $\frac{\nu}{2}$ also suggested that this option leads to the best performance.

The complex STFT $\mathbf{X} \in \mathbb{C}^{K \times N}$ of each drum loop recording is further converted to a logarithmically compressed magnitude STFT $\mathbf{V} = \log(1 + \alpha|\mathbf{X}|) \in \mathbb{R}_{\geq 0}^{K \times N}$ with $\alpha = 10$, as explained in Section 1.3. The same feature transformation is applied to the WAV files of the instrument sample sounds, yielding logarithmically compressed magnitude STFTs $\mathbf{M}_r \in \mathbb{R}^{K \times T_r}$ for each instrument $r \in [1 : R]$ with varying length T_r.

The two-dimensional NMFD templates for a given from loop are obtained by zero-padding each instrument template $\mathbf{M}_r$ to have length $T = \max(\{T_r \text{ for } \forall r \in [1 : R]\})$. This yields a set of templates $\{\mathbf{P}_r \in \mathbb{R}^{K \times T} \text{ for } \forall r \in [1 : R]\}$, which are then grouped to form the initialized pattern tensor $\mathbf{P}$, as explained in Section 1.4.6.

The one-dimensional NMF templates are obtained by averaging the spectral coefficient intensities of each $\mathbf{M}_r \in \mathbb{R}^{K \times T_r}$ over the T_r time windows, thereby obtaining a one-dimensional NMF

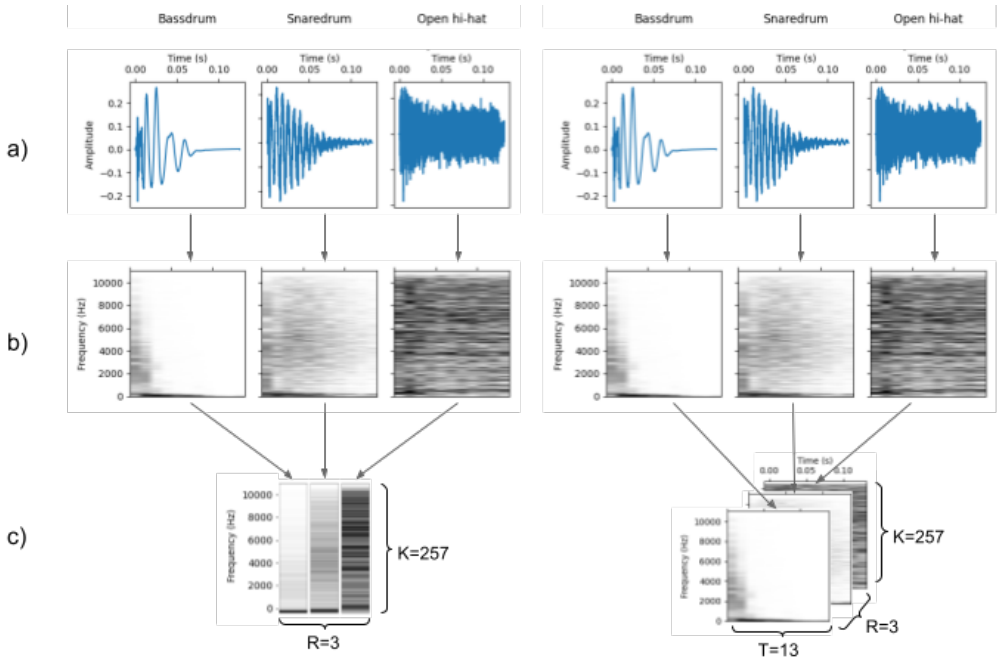

Figure 2.2: Template initialization. a) Waveforms of the sample sounds bassdrum, snare drum, and open hi-hat, respectively. b) Logarithmically compressed magnitude spectrograms derived from the waveforms. c) Initialized NMF template matrix (left) and NMFD pattern tensor (right).

template vector $\boldsymbol{w}_r \in \mathbb{R}^K$ for each of the $r \in [1:R]$ instruments. As such, the kth spectral coefficient in the template vector $\boldsymbol{w}_r(k)$ for $\forall k \in [1:K]$ is given by

$$\boldsymbol{w}_r(k) = \frac{\sum_{t=1}^{T_r} \mathbf{M}_r(k,t)}{T_r}.$$

Then, each column $\mathbf{W}(,r) \in \mathbb{R}^K$ of the template matrix $\mathbf{W} \in \mathbb{R}^{K \times R}$ is initialized with $\boldsymbol{w}_r$.

2.3 NMF variants

A preliminary mock experiment showed no significant difference between performance using uniform and random initialization of the activation matrix $\mathbf{H}$ and the additional noise components in the template matrix $\mathbf{W}$, thus, uniform initialization was chosen arbitrarily.

The tested NMF variants are combinations of the unique features that can be identified in previous research, as explained in Section 1.4.7. The update equations are listed below.

As suggested by Müller, 2021, gradient descent stops at the iteration limit $I = 1000$ for NMF and $I = 50$ for NMFD, or when the maximal element-wise difference between two subsequently computed matrices is smaller than $\epsilon = 0.001$. As such,

$$\big(\max(\mathbf{H}' - \mathbf{H}) \leq \epsilon \wedge \max(\mathbf{W}' - \mathbf{W}) \leq \epsilon\big) \implies \text{stop} \tag{2.1}$$

for the NMF variants, and

$$\big(\max(\mathbf{H}' - \mathbf{H}) \leq \epsilon \wedge \max(\mathbf{P}' - \mathbf{P}) \leq \epsilon\big) \implies \text{stop} \tag{2.2}$$

for the NMFD variants.

2.3.1 Update rules

Adaptive NMF

The adaptive NMF implementation builds on the FMP notebook created by Müller & Zalkow (2019).

$$\mathbf{W} \leftarrow \mathbf{W} \odot \frac{\frac{\mathbf{V}}{\hat{\mathbf{V}}}\mathbf{H}^\intercal}{\mathbf{J}\mathbf{H}^\intercal} \tag{2.3}$$

$$\mathbf{H} \leftarrow \mathbf{H} \odot \frac{\mathbf{W}^\intercal\frac{\mathbf{V}}{\hat{\mathbf{V}}}}{\mathbf{W}^\intercal\mathbf{J}} \tag{2.4}$$

Semi-adaptive NMF

$$\mathbf{W} \leftarrow (1-\alpha)\cdot\mathbf{W}^{(0)} + \alpha\cdot\mathbf{W}\odot\frac{\frac{\mathbf{V}}{\bar{\mathbf{V}}}\mathbf{H}^{\intercal}}{\mathbf{J}\mathbf{H}^{\intercal}}, \tag{2.5}$$

where $\alpha = (\frac{i}{I})^{\beta}$, $i \in [1:I]$ is the iteration count, $I \in \mathbb{N}$ is the iteration limit $I = 1000$, and $\beta \in [1...6]$ is a hyper-parameter governing the rate of divergence from the initial template matrix $\mathbf{W}^{(0)}$.

$$\mathbf{H} \leftarrow \mathbf{H}\odot\frac{\mathbf{W}^{\intercal}\frac{\mathbf{V}}{\bar{\mathbf{V}}}}{\mathbf{W}^{\intercal}\mathbf{J}} \tag{2.6}$$

Fixed NMF

$$\mathbf{H} \leftarrow \mathbf{H}\odot\frac{\mathbf{W}^{\intercal}\frac{\mathbf{V}}{\bar{\mathbf{V}}}}{\mathbf{W}^{\intercal}\mathbf{J}} \tag{2.7}$$

Adaptive NMFD

The adaptive NMFD implementation builds on the NMF Toolbox created by López-Serrano et al. (2019).

$$\mathbf{P}(,,t) \leftarrow \mathbf{P}(,,t)\odot\frac{\frac{\mathbf{V}}{\bar{\mathbf{V}}}\cdot\left(\overset{t\rightarrow}{\mathbf{H}}\right)^{\intercal}}{\mathbf{J}\cdot\left(\overset{t\rightarrow}{\mathbf{H}}\right)^{\intercal}} \tag{2.8}$$

$$\mathbf{H} \leftarrow \mathbf{H}\odot\frac{\mathbf{P}(,,t)^{\intercal}\cdot\left[\overset{\leftarrow t}{\frac{\mathbf{V}}{\bar{\mathbf{V}}}}\right]}{\mathbf{P}(,,t)^{\intercal}\cdot\mathbf{J}} \tag{2.9}$$

Semi-adaptive NMFD

$$\mathbf{P}(,,t) \leftarrow (1-\alpha)\cdot\mathbf{P}(,,t)^{(0)} + \alpha\cdot\mathbf{P}(,,t)\odot\frac{\frac{\mathbf{V}}{\bar{\mathbf{V}}}\cdot\left(\overset{t\rightarrow}{\mathbf{H}}\right)^{\intercal}}{\mathbf{J}\cdot\left(\overset{t\rightarrow}{\mathbf{H}}\right)^{\intercal}}, \tag{2.10}$$

where $\alpha = (\frac{i}{I})^{\beta}$, $i \in [1:I]$ is the iteration count, $I \in \mathbb{N}$ is the iteration limit $I = 50$, and $\beta \in [1...6]$ is a hyper-parameter governing the rate of divergence from the initial pattern tensor $\mathbf{P}^{(0)}$.

$$\mathbf{H} \leftarrow \mathbf{H}\odot\frac{\mathbf{P}(,,t)^{\intercal}\cdot\left[\overset{\leftarrow t}{\frac{\mathbf{V}}{\bar{\mathbf{V}}}}\right]}{\mathbf{P}(,,t)^{\intercal}\cdot\mathbf{J}} \tag{2.11}$$

Fixed NMFD

$$\mathbf{H} \leftarrow \mathbf{H}\odot\frac{\mathbf{P}(,,t)^{\intercal}\cdot\left[\overset{\leftarrow t}{\frac{\mathbf{V}}{\bar{\mathbf{V}}}}\right]}{\mathbf{P}(,,t)^{\intercal}\cdot\mathbf{J}} \tag{2.12}$$

Added template components

The number of added uniformly initialized noise template components Q was considered in the range $Q \in [0,1,2,3,4,5]$. The noise template columns in $\mathbf{W}$ and patterns in $\mathbf{P}$ were intialized uniformly and were adaptive during gradient descent, while the drum components were adaptive, semi-adaptive, or fixed, based on the condition evaluated in the experiment. The noise activation rows in $\mathbf{H}$ were intialized and updated in the same way as those corresponding to drum templates.

2.4 Onset detection

After the source separation step, gains in the activation matrix $\mathbf{H}$ were converted to onsets for each instrument, as explained below.

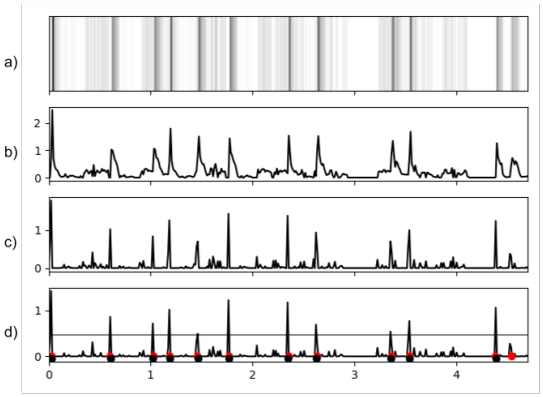

Figure 2.3: Processing a) a row of the activation matrix $\mathbf{H}(r,)$ for instrument onset detection. b) Local energy function $\boldsymbol{e}$. c) Energy-based novelty function $\Delta_{\boldsymbol{e}}$. d) Enhanced novelty function $\tilde{\Delta}_{\boldsymbol{e}}$. The grey line denotes the onset threshold θ, red dots indicate real onsets, and black dots indicate the set of onsets O labelled by the algorithm.

Let us take the rth row of the activation matrix $\mathbf{H}(r,) \in \mathbb{R}^{N}_{\geq 0}$, where N is the number time windows in the recording's magnitude STFT, and $r \in [1, R]$ is the index of the instrument to which the row's activation corresponds. $\mathbf{H}(r,)$ can be thought of as a local energy function $\boldsymbol{e}$ for the respective instrument, from which an energy-based novelty function is derived, as explained by Müller (2021).

Firstly, the derivative of the energy function is calculated, which in the discrete case can be defined as the distance between two subsequent energy values $\boldsymbol{e}(n+1) - \boldsymbol{e}(n)$, where $n \in [1, N]$ is the time step. As onset detection is concerned only with increases in energy, half wave rectification is applied, defined as

$$||a||_{\geq 0} = \begin{cases} a, & \text{if } a \geq 0 \\ 0 & \text{if } a < 0. \end{cases} \tag{2.13}$$

As such, the energy-based novelty function is given by

$$\Delta_{\boldsymbol{e}}(n) = ||\boldsymbol{e}(n+1) - \boldsymbol{e}(n)||_{\geq 0}. \tag{2.14}$$

Additionally, small fluctuations are suppressed using a local average function $\boldsymbol{\mu}$, defined as

$$\boldsymbol{\mu}(n) = \frac{1}{2M+1} \sum_{m=-M}^{M} \Delta_{\boldsymbol{e}}(n+m), \tag{2.15}$$

where $M \in \mathbb{N}$ is the averaging window size. The present implementation uses $M = 3$. The local average is then subtracted from the original novelty function and the result is half-wave rectified. As such, an enhanced novelty function $\tilde{\Delta}_E$ is obtained, where peaks only show if they exceed the novelty function's local average (Müller, 2021).

$$\tilde{\Delta}_{\boldsymbol{e}} = ||\Delta_{\boldsymbol{e}}(n) - \boldsymbol{\mu}(n)||_{\geq 0} \tag{2.16}$$

Finally, a peak picking algorithm is applied to the enhanced novelty function, which yields a set of onset candidates $P = \{p_1, p_2, ...\}$.

It proved to be beneficial to select peaks only beyond a certain height due to additional fluctuations not filtered out by the local average function. Thus, the final set of onsets O is given by

$$O = \left\{ p_i \text{ if } p_i > \frac{\max(P)}{\theta} \text{ for } \forall p_i \in P \right\}, \tag{2.17}$$

where $\max(P)$ is the maximum peak height, and the onset threshold $\theta \in \mathbb{R}_{>0}$ can be thought of as a hyperparameter balancing precision and recall. Based on preliminary mock experiments, NMF variants were implemented using $\theta = 3$, and NMFD variants using $\theta = 6$.

2.5 Evaluation

According to Wu et al. (2018), precision, recall, and the F-measure are commonly used for evaluating the performance of automatic drum transcription algorithms. The precision shows the ratio of correctly detected onsets to all detected onsets, the recall shows the ratio of correctly detected onsets to all annotated onsets, and the F-measure balances both concerns. The present analysis focuses on the F-measure.

Wu et al. (2018) point out that humans identify two sounds as separate if they at least are 8-10 ms apart, thus, transcription inaccuracies up to 20 ms are acceptable. However, as human annotated data sets are transcribed less accurately, ADT performance more often evaluated on wider tolerance windows, i.e., 50-100 ms (Wu et al., 2018). The present project adopts the most common 50 ms window.

The F-measure is defined based on true positives, false positives, true negatives, and false negatives, however, true negatives are not used in ADT. True positives (TP) denote detected onsets that fall within the tolerance window of an annotated drum event, while false positives (FP) denote detected onsets that do not. As the tolerance window is quite narrow, it is unlikely that more onsets would be detected within the tolerance window of a single annotated event. However, in case this occurs, the closest one will be classified as TP and others as FP. False negatives (FN) denote an annotated drum event with no detected onset within its tolerance window. Then, the F-measure can be calculated as

$$\mathrm{F} = \frac{2 \cdot \mathrm{TP}}{2 \cdot \mathrm{TP} + \mathrm{FP} + \mathrm{FN}}. \tag{2.18}$$

To assess the performance of a given NMF variant on a noise condition, the F-measure is computed for each of the 20 drum loops with added noise, then averaged to obtain an overall performance measure. For the mild and loud noise conditions, performance is averaged over the airplane, chatter, and ambient conditions. As such, four F-measures are obtained for each variant, one for each noise condition (none, mild, loud, extreme).

3 Results

3.1 Adaptivity

Recall from Section 1.4.4 that the larger the value of the hyperparameter β is in Equation 1.11, the

less adaptive the initialized templates are. $\beta = 0$ corresponds to the adaptive condition, while $\beta \to \infty$ leads to the fixed condition.

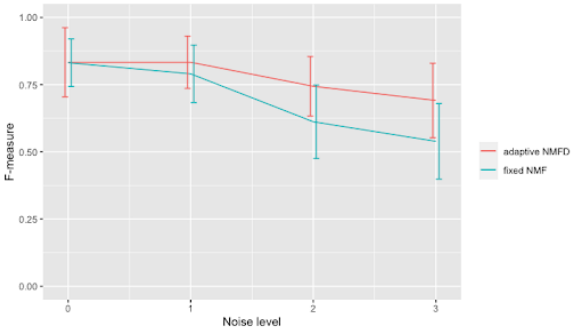

Figure 3.1: Average F-measure of the best performing NMF and NMFD variants: fixed NMF (blue) and adaptive NMFD (red), respectively, as a function of noise level. The performance shown here is without added noise components ($Q = 0$).

NMF On the none and mild noise conditions, NMF variants exhibit a clear tendency towards better performance as the hyperparameter β increases, as shown in Figure 3.2 and Table D.1. The fixed variant with F $= 0.832 \pm 0.089$, 0.79 ± 0.107, 0.612 ± 0.137, 0.539 ± 0.141, in order of increasing noise level, outperforms the adaptive and semi-adaptive variants on the none and mild conditions, however, the semi-adaptive NMF performs slightly better than the fixed variant in the loud and extreme conditions. The adaptive variant performs the worst in all noise conditions with F $= 0.684 \pm 0.101$, 0.645 ± 0.147, 0.508 ± 0.121, 0.476 ± 0.095, in order of increasing noise level.

NMFD does not show large performance variation as a function of adaptivity, as shown in Figure 3.2 and Table D.2. The adaptive NMFD variant with F $= 0.833 \pm 0.129$, 0.833 ± 0.097, 0.744 ± 0.111, 0.691 ± 0.139, on the four noise conditions in order of increasing noise level, performs slightly better than the other NMFD variants in the mild and loud conditions. On the none condition, the fixed variant achieves a higher score, namely F $= 0.836 \pm 0.109$, and in the extreme condition, the adaptive variant is slightly outperformed by three semi-adaptive variants, the maximum being F $= 0.701 \pm 0.105$ with $\beta = 5$, however, these differences seem negligible.

3.2 Template dimensionality

Figure 3.1 shows the average F-measures of the fixed NMF and the adaptive NMFD variants as a function of noise level. Even though the two variants exhibit approximately the same performance on the none noise condition, F $= 0.832 \pm 0.089$ for fixed NMF and 0.833 ± 0.129 for adaptive NMFD, their performance diverges significantly with the increase in noise level. Additionally, the worst performing NMFD variants outperform the best performing NMF variants on every noise condition, as it can be observed in Tables D.1 and D.2.

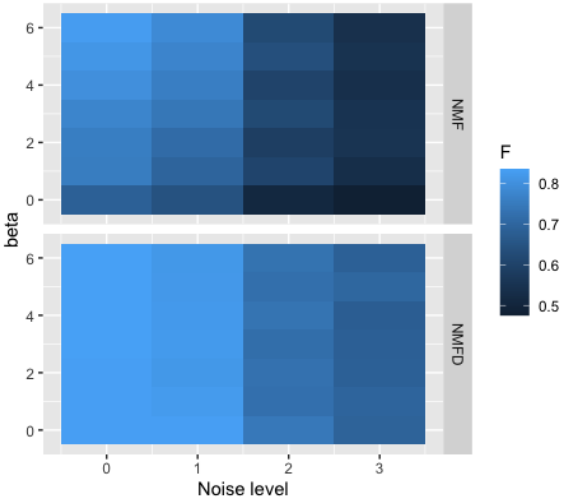

Figure 3.2: Heatmap of the average F-measures of NMF (upper) and NMFD (lower) as a function of the degree of adaptivity and noise level. $\beta = 0$ corresponds to the adaptive condition, $\beta \in [1 : 6]$ to the semi-adaptive condition, and the fixed condition ($\beta \to \infty$) is not included. The performance shown here is without added noise components ($Q = 0$).

3.3 Added template components

As shown in Figure 3.3, the number of added noise template components $Q \in [0 : 5]$ does not lead to significant performance differences for adaptive and fixed NMF and NMFD. The same holds for the semi-adaptive variants, as shown in Figure 3.4.

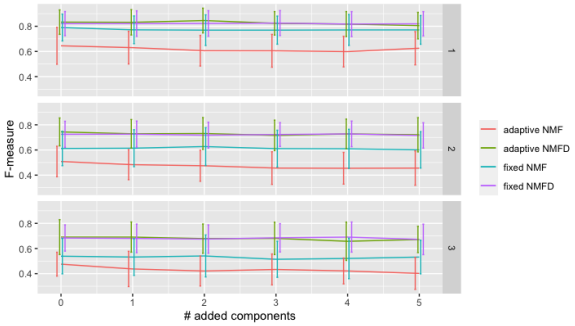

Figure 3.3: Average F-measure of adaptive NMF (red), adaptive NMFD (green), fixed NMF (cyan), and fixed NMFD (violet) as a function of the number of additional template components $Q \in [0:5]$. The noise conditions mild (1), loud (2), and extreme (3) are shown.

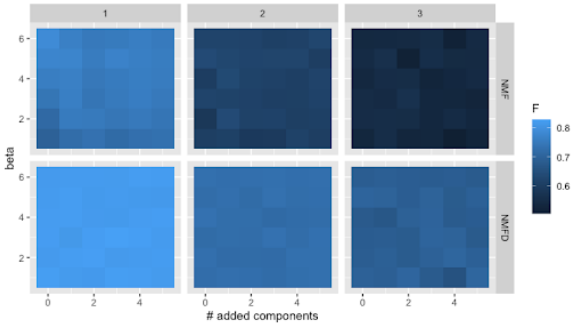

Figure 3.4: Heatmap of the average F-measures of semi-adaptive NMF (upper) and NMFD (lower) as a function of adaptivity ($\beta \in [1:6]$) and the number of additional template components $Q \in [0:5]$. The noise conditions mild (1), loud(2), and extreme (3) are shown.

4 Discussion

4.1 Adaptivity

NMF performs best in its fixed form without noise and with mild noise, and its semi-adaptive form leads to the best performance on louder conditions. NMFD performance is not subject to as much variance as a function of adaptivity, however, its adaptive form performs slightly better overall.

Dittmar & Gärtner (2014) found that semi-adaptivity for NMF did not improve performance compared to the fixed variant on the DTD task with generic template initialization. However, applied to the ADT task with specific template initialization and transcription in the presence of background noise, semi-adaptivity for NMF appears to be beneficial on the louder noise levels. The results are homogeneous in that adaptive NMF performs worse than the other two variants.

Semi-adaptivity for NMFD has not yet been investigated, however, it appears to have comparable performance to the adaptive and fixed variants and does not lead to a significant improvement.

4.2 Template dimensionality

The complexity of two-dimensional NMFD patterns seems to result in more robustness against noise, in comparison to one-dimensional NMF templates. Fixed NMF and adaptive NMFD start at a comparable performance on the none noise condition with $F = 0.832 \pm 0.089$ and $F = 0.833 \pm 0.129$, respectively. However, fixed NMF performance degrades significantly as the noise level increases, to $F = 0.79 \pm 0.107$, 0.612 ± 0.137, 0.539 ± 0.141. On the other hand, adaptive NMFD attains the same performance with $F = 0.833 \pm 0.097$ in the mild noise condition, and degrades only to $F = 0.744 \pm 0.111$ and $F = 0.691 \pm 0.139$ on the loud and extreme conditions, respectively, as shown in Figure 3.1.

Recall that the initialization of templates happens in the presence of background noise. NMF template columns are single dimensional and represent the average spectra of the isolated instrument sample with noise. If a milder drum instrument sample, such as a bass drum hit is superposed by a noise segment with large spectral magnitudes, then it may dominate the template and thereby the computed activation corresponding to the template. On the other hand, NMFD patterns are two-dimensional, also encoding the temporal variation of drum spectra and the added noise spectra. As the added noise is less likely to occur in the same spectrotemporal form during the drum loop recording, the adaptive NMFD variant is more likely to filter it out and thereby converge to the spectrotemporal patterns corresponding to the actual drum sound.

NMFD is also meant to capture highly time-variant sounds better than NMF (Wu et al., 2018), however, the performance of fixed NMF and adaptive NMFD are approximately the same on the none noise condition, therefore, the advantage of

NMFD seems to result rather from its robustness against noise.

4.3 Added template components

Although Wu & Lerch (2015) found that introducing additional uninitialised template components led to a robust system for the DTM task, the number of additional noise template components $Q \in [0:5]$ did not appear to make a difference in the present task. Thus, $Q = 0$ can be used for the sake of computational efficiency.

4.4 Limitations

The focus of selecting drum loops for the dataset was to include a broader range of instruments, genres, and playing styles than it is customary in NMF research, in order to evaluate the variants in a more general and realistic scenario. However, the dataset is quite limited in its size, containing only 20 drum loops that are being evaluated on each noise condition. Since the present study is exploratory in its nature, this appears sufficient, however, evaluation could benefit from a larger dataset. Moreover, a synthetic dataset only allows for the variation of striking velocity but not striking position. Striking position may introduce additional spectrotemporal variation, as explained in Appendix B, which is not accounted for in the present evaluation. Additionally, the faithfulness of simulating a home musician's recording scenario could be enhanced by adding different types of reverberation conditions to the dataset.

4.5 Future directions

A truly usable ADT system would require near-perfect accuracy, however, all of the evaluated NMF variants failed to produce results that approach this objective, even in the condition without noise. This seems to suggest that the way forward in ADT is not with NMF variants. Given the performance of neural network-based systems (see Section 1.1) on the generic datasets, it appears more productive to focus on creating more sophisticated datasets that would allow neural network-based systems to generalize to a wider range of drum instruments and more realistic recording conditions.

5 Conclusion

The present study explored the performance of different NMF variants in a simulated ADT scenario of interactive initialization and recording in the presence of background noise, a setting especially relevant in music education and music production with rudimentary equipment. NMF performed best in its fixed form without noise and with mild noise, and in its semi-adaptive form with louder conditions. NMFD performance was not subject to as much variation as a function of adaptivity, however, its adaptive form performed slightly better overall. Added noise template components did not make a significant difference. As for the template dimensionality, NMFD was clearly better suited for the task, due to the robustness given by the two-dimensional templates. As such, adaptive NMFD without added noise components appears to be most well-suited for the task with $\mathrm{F} = 0.833 \pm 0.129$, 0.833 ± 0.097, 0.744 ± 0.111, 0.691 ± 0.139, on the four noise conditions in order of increasing noise level. As this performance is insufficient for real-life application, a more productive near-future direction for the field of ADT appears to be resolving the data problem in order to create more general neural network-based ADT systems.

References

Battenberg, E., Huang, V., & Wessel, D. (2012). Live drum separation using probabilistic spectral clustering based on the Itakura-Saito divergence. In *Proceedings of the AES 45th conference on time-frequency processing in audio, Helsinki, Finland.*

Benetos, E., Ewert, S., & Weyde, T. (2014). Automatic transcription of pitched and unpitched sounds from polyphonic music. In *Proceedings of 2014 IEEE international conference on acoustics, speech and signal processing (ICASSP)* (pp. 3107–3111).

Cartwright, M., & Bello, J. P. (2018). Increasing drum transcription vocabulary using data synthesis. In *Proceedings of the international conference on digital audio effects (DAFx)* (pp. 72–79).

Cooley, J. W., & Tukey, J. W. (1965). An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, *19*(90), 297–301.

Dittmar, C., & Gärtner, D. (2014). Real-time transcription and separation of drum recordings based on nmf decomposition. In *Proceedings of the international conference on digital audio effects (DAFx)* (pp. 187–194).

Gajhede, N., Beck, O., & Purwins, H. (2016). Convolutional neural networks with batch normalization for classifying hi-hat, snare, and bass percussion sound samples. In *Proceedings of the 2016 audio mostly conference* (pp. 111–115).

Gillet, O., & Richard, G. (2004). Automatic transcription of drum loops. In *Proceedings of 2004 IEEE international conference on acoustics, speech, and signal processing* (Vol. 4, pp. 269–272).

Gillet, O., & Richard, G. (2006). Enst-drums: an extensive audio-visual database for drum signals processing. In *Proceedings of the international society for music information retrieval conference (ISMIR).*

Gillet, O., & Richard, G. (2008). Transcription and separation of drum signals from polyphonic music. *IEEE Transactions on Audio, Speech, and Language Processing*, *16*(3), 529–540.

Gillick, J., Roberts, A., Engel, J., Eck, D., & Bamman, D. (2019). Learning to groove with inverse sequence transformations. In *Proceedings of the international conference on machine learning (ICML).*

Lee, D., & Seung, H. S. (2000). Algorithms for non-negative matrix factorization. *Advances in neural information processing systems*, *13*.

López-Serrano, P., Dittmar, C., Özer, Y., & Müller, M. (2019). NMF toolbox: Music processing applications of nonnegative matrix factorization. In *Proceedings of the international conference on digital audio effects (DAFx).* Birmingham, UK.

McFee, B., Metsai, A., McVicar, M., Balke, S., Thomé, C., Raffel, C., ... Kim, T. (2022, June). *librosa/librosa: 0.9.2.* Zenodo. doi: 10.5281/zenodo.6759664

Müller, M. (2021). *Fundamentals of music processing: Using Python and Jupyter notebooks.* Springer Nature.

Müller, M., & Zalkow, F. (2019). FMP notebooks: Educational material for teaching and learning fundamentals of music processing. In *Proceedings of the international conference on music information retrieval (ISMIR).* Delft, The Netherlands.

Paulus, J., & Klapuri, A. (2009). Drum sound detection in polyphonic music with hidden markov models. *EURASIP Journal on Audio, Speech, and Music Processing*, *2009*, 1–9.

Smaragdis, P. (2004). Non-negative matrix factor deconvolution; extraction of multiple sound sources from monophonic inputs. In *Proceedings of the international conference on independent component analysis and signal separation* (pp. 494–499).

Southall, C., Stables, R., & Hockman, J. (2017). *Automatic drum transcription for polyphonic recordings using soft attention mechanisms and convolutional neural networks.* International Society of Music Information Retrieval.

van Netten, S., Maathuis, H., & Visser, J. (2020). *Lecture notes in signals and systems.* Faculty of Science and Engineering, University of Groningen.

van Rij, J. (2020). *Lecture notes in language and speech technology.* Faculty of Science and Engineering, University of Groningen.

Wu, C.-W., Dittmar, C., Southall, C., Vogl, R., Widmer, G., Hockman, J., ... Lerch, A. (2018). A review of automatic drum transcription. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *26*(9), 1457–1483.

Wu, C.-W., & Lerch, A. (2015). Drum transcription using partially fixed non-negative matrix factorization with template adaptation. In *Proceedings of the international society for music information retrieval conference (ISMIR)* (pp. 257–263).

A Notation

- $\mathbf{X}$ = matrix
- $\boldsymbol{x}$ = vector
- X = constant, set
- x = variable
- $\mathbf{X}(i,)$ = ith row of matrix $\mathbf{X}$.
- $\mathbf{X}(,i)$ = ith column of matrix $\mathbf{X}$.
- $\mathbf{X}(i,j)$ = ith element of the jth column of matrix $\mathbf{X}$.
- $\mathbf{X}^\intercal$ = transpose of matrix $\mathbf{X}$.
- $\mathbf{X} \cdot \mathbf{Y}$ = dot product of matrices $\mathbf{X}$ and $\mathbf{Y}$.
- $\boldsymbol{x}(i)$ = ith element in vector $\boldsymbol{x}$.
- $[1:X]$ = integer interval $[1, 2, ..., X-1, X]$.

B The nature of drum sounds

When a mechanical system is set in motion with an initial input and allowed to vibrate freely, it vibrates at one or more of its natural frequencies, which depend on the system's structure, material, and boundary conditions. If it is not isolated, the vibrating system gradually gives off its energy to its environment until vibration damps down completely.

Vibrating objects generate pressure waves in air. This is because the object's motion in one direction compresses the air directly in front of it, while its motion in the other direction decompresses the air. Molecules are always in motion, constantly exchanging energy, thereby making such compression and decompression points propagate through air. This results in a pressure wave propagating the frequencies of the vibrating body (van Rij, 2020).

When a pressure wave hits the eardrum, forced vibration occurs; thereby, the eardrum picks up the vibration, which gets transmitted through other structures to the fluid in the cochlea. The waves flowing through the cochlea induce waves in the basilar membrane. Each part of the basilar membrane has different resonant frequencies; thus, different incoming frequencies lead to resonant behavior at different locations. On top of the basilar membrane, the organ of Corti contains receptors called hair cells, which pick up the vibrations. If the amplitude at a given point is large enough - that is, when resonance occurs - the movement of the hair cells opens ion channels, which causes the release of neurotransmitters and the firing of an action potential, carrying information of the respective frequency band. Thereby, the inner ear acts as a filter bank with bandpass filters, translating mechanical vibration to electrical impulses corresponding to the vibration's constituent frequencies. Corresponding action potentials are finally carried to the brain through the auditory nerve, where the time-frequency information can be further processed (van Rij, 2020).

The two most essential qualities of sound humans perceive are pitch and loudness. Higher frequencies denote higher pitch, while loudness has to do with the amplitude of pressure waves. Humans can hear frequencies in the range of 20 to 20,000 Hz, and can distinguish sounds that are at least 20 ms apart (Wu et al., 2018).

Instruments in a drum kit can be classified as membranophones, and idiophones. Membranophones consist of an elastic membrane, stretched over one end of a cylindrical body. Some membranophones have additional components, such as an additional resonating membrane stretched over the other end of the body. Or, in case of the snare drum, wires stretched accross the lower membrane. On the other hand, idiophones, such as the cymbal are metallic bodies that vibrate as a whole (Wu et al., 2018).

So how do humans discern which drum is playing at a given moment? Each drum instrument has a distinct structure, material, and boundary conditions, which makes it resonate with certain characteristic frequencies. For the case of membranophones, the membrane tension and body shape are especially important. The human brain classifies sounds based on these characteristic frequencies and their temporal variation. Loudness does not matter for the classification of a drum sound. Nevertheless, it contributes to a listening experience; thus, the transcription algorithm should recognize it in an ideal scenario. This, however, is beyond the scope of the present project.

According to Wu et al. (2018), there are several particularities of drum sounds that distinguish automatic drum transcription from other music information retrieval problems. Right after the *onset*, drum sounds exhibit an initial transient with broadband, noise-like spectrum, called *attack*, followed by a slow *decay* of a few hundred milliseconds. While there is little variability in the sound pattern produced by striking a specific drum, the attack often has different spectral characteristics than the decay, and some drums exhibit even more complex time-varying patterns. Additionally, different striking positions and velocities can introduce some variation, especially if the tightness of the membrane is not uniform.

Tonal components in drums have an inharmonic structure, meaning that the overtones depart from being integer multiples of the fundamental frequency. Thus, many algorithms for pitched instrument transcription, such as fundamental frequency estimation, are not applicable in ADT (Wu et al., 2018).

C MIDI

As explained by Müller (2021), *beats* are the fundamental temporal units that Western musical pieces build on. Beats define the overall duration of notes, given as a fraction with regard to a whole note $\frac{1}{B}$, where $B \in \mathbb{Z}_{>0}$. A number $N \in \mathbb{Z}_{>0} \leq B$ of beats are contained in a larger structure called a *bar* or *measure*. The temporal structure of a piece of music is indicated by the *key signature*, defined as $\frac{N}{B}$.

The duration of a beat is defined by the *tempo*, indicated in beats per minute (BPM). For instance, if a beat is defined as a quarter note, that is, $\frac{1}{B} = \frac{1}{4}$ and the tempo is 120 BPM, then, a quarter note takes $\frac{60}{120} \cdot \frac{1}{4} = 0.125$ seconds.

The MIDI standard encodes music in a digital format that computers can parse. A MIDI file contains meta-messages that are relevant for parsing and a list of MIDI messages. If transmitted to an electronic instrument or applied to a virtual instrument in a DAW, the messages trigger sounds. Note that MIDI does not carry information about sound directly; it only encodes performance information, i.e., with what velocity and pitch, and when given instruments were played. For the case of drums, pitch does not play a role. Instead, a drum kit occupies a single channel in the MIDI file and each kit piece has a corresponding note.

The MIDI messages m of interest to the present project indicate the start and end of notes, respectively, accompanied by a note number, a velocity value, a channel number, and a time value. As such,

$$\text{type}(m) \in \{\text{"note-on"}, \text{"note-off"}\}.$$

In the midi message m_i, the note number $n_i \in [0, 127]$ corresponds to the musical pitches C0 to G#9 for pitched instruments, while it denotes individual drum instruments if the channel represents a drum kit. The velocity $v_i \in [0, 127]$ defines the sound intensity and has different interpretations depending on the instrument. For drum instruments, it corresponds to the striking velocity of the drum. The channel number $c_i \in [0, 15]$ denotes the MIDI channel assigned to the instrument or kit. The time value $\Delta k_i \in \mathbb{R}_{>0}$ defines the time of execution, given in ticks, for message m_i relative to m_{i-1}.

To use a MIDI file for performance evaluation, it must be converted to a set of labels, namely, timestamps for each instrument, given in seconds. Given

the tick duration D, the absolute time position in seconds t_i of m_i can be calculated as

$$t_i = (\sum_{n=0}^{i} \Delta k_n) \cdot D. \tag{C.1}$$

Thereby, we obtain tuples (m_i, t_i). For each instrument r, onsets are given by t_i where note(m_i) = r and type(m_i) = "note-on".

D Results

Table D.1: Average F-measures for NMF without additional template components (Q=0), as a function of adaptivity and noise level.

	None	Mild
Adaptive	0.684 ± 0.101	0.645 ± 0.147
Semi $\beta = 1$	0.756 ± 0.112	0.696 ± 0.125
Semi $\beta = 2$	0.757 ± 0.114	0.713 ± 0.107
Semi $\beta = 3$	0.775 ± 0.117	0.74 ± 0.108
Semi $\beta = 4$	0.799 ± 0.12	0.758 ± 0.113
Semi $\beta = 5$	0.813 ± 0.11	0.774 ± 0.107
Semi $\beta = 6$	0.829 ± 0.096	0.787 ± 0.106
Fixed	0.832 ± 0.089	0.79 ± 0.107
	Loud	Extreme
Adaptive	0.508 ± 0.121	0.476 ± 0.095
Semi $\beta = 1$	0.605 ± 0.116	0.53 ± 0.135
Semi $\beta = 2$	0.585 ± 0.128	0.549 ± 0.128
Semi $\beta = 3$	0.619 ± 0.127	0.548 ± 0.141
Semi $\beta = 4$	0.602 ± 0.127	0.535 ± 0.136
Semi $\beta = 5$	0.637 ± 0.133	0.547 ± 0.162
Semi $\beta = 6$	0.619 ± 0.131	0.537 ± 0.151
Fixed	0.612 ± 0.137	0.539 ± 0.141

Table D.2: Average F-measures for NMFD without additional template components (Q=0), as a function of adaptivity and noise level.

	None	Mild
Adaptive	0.833 ± 0.129	0.833 ± 0.097
Semi $\beta = 1$	0.834 ± 0.11	0.825 ± 0.095
Semi $\beta = 2$	0.834 ± 0.11	0.818 ± 0.097
Semi $\beta = 3$	0.835 ± 0.11	0.822 ± 0.095
Semi $\beta = 4$	0.835 ± 0.11	0.821 ± 0.099
Semi $\beta = 5$	0.835 ± 0.11	0.819 ± 0.101
Semi $\beta = 6$	0.835 ± 0.11	0.817 ± 0.097
Fixed	0.836 ± 0.109	0.821 ± 0.096
	Loud	Extreme
Adaptive	0.744 ± 0.111	0.691 ± 0.139
Semi $\beta = 1$	0.724 ± 0.108	0.696 ± 0.11
Semi $\beta = 2$	0.727 ± 0.107	0.685 ± 0.108
Semi $\beta = 3$	0.72 ± 0.102	0.682 ± 0.122
Semi $\beta = 4$	0.732 ± 0.094	0.676 ± 0.103
Semi $\beta = 5$	0.722 ± 0.104	0.701 ± 0.105
Semi $\beta = 6$	0.73 ± 0.107	0.685 ± 0.111
Fixed	0.725 ± 0.104	0.683 ± 0.104