NTNU
Norwegian University of
Science and Technology

Master's thesis

2020

Frode van der Meeren

NTNU
Norwegian University of
Science and Technology
Faculty of Humanities
Department of Music

Master's thesis

Frode van der Meeren

# Effects of Restrictive Classification on Audio Source Separation Using Non-Negative Matrix Factorization

May 2020

NTNU
Norwegian University of
Science and Technology

# Effects of Restrictive Classification on Audio Source Separation Using Non-Negative Matrix Factorization

**Frode van der Meeren**

Master of Muisc Technology
Submission date: May 2020
Supervisor: Øyvind Brandtsegg

Norwegian University of Science and Technology
Department of Music

## Table of Contents

## Introduction

In everyday life, the human mind perceives sound, picks it apart, and interpret it into unique sounds from separate sources. This ability has through history been fundamental for survival, and through evolution it has evolved into a complex system involving several intricate mechanisms and a wide range of means to differentiate and separate sound from various individual sources. The mind has thus shown that source separation of sources is possible, something which implies that it might theoretically be a possible task for a computer to do as well.

Humans and many other animals have ears as the main sensory organ for sound. Soundwaves in air are converted to mechanical waves which in turn is mechanically converted to frequency-amplitude information sent to the brain. The brain performs source recognition, source separation, in addition to identifying the direction of the source. Another feature of the brain is also the ability to learn new sounds, and associate audio features with other impressions. Sounds are in this way put into context and perceived depending on both earlier experiences and the current situation.

For at least 40 000 years, humans have had a desire to combine sounds to produce music. While traditionally as a form for performance art, the rise of new technologies in modern times have drastically changed the way music is produced and listened to. One of the most important and influential music-related technologies in the last 100 years, has been the ability to record and reproduce audio. This technology has enabled the objectification of music and sounds, which in turn has unlocked a wide range of tools and techniques for working with audio. It enables tools for processing, shaping, and analyzing audio at any time after the time of recording. It is desirable to have clear separated sources when using sounds in audio processing.

Computers are machines that enable algorithmic processing of information. This allows for automatic processing of data, and combined with recording technology and sampling, audio can be processed as digital signals. The benefit of this, is that the computer can be used as a tool for retrieving information from and process the audio signal mathematically.

Traditionally, computers are tied to running rigid programs, based on a set of fixed instructions, and dealing with discrete numbers. These properties make computers very suited for accurate mathematic and logic operations, but it poses greater challenges for abstract tasks. While the human mind is perfectly fit for making sense of abstract sensory input, a computer will quickly face great difficulties dealing with the unknown. Source separation of an arbitrary soundscape also pose these problems.

As early as the 1950s it was theorized that computers might be programmed to behave in ways which seems cognitive, behaving like an intelligent agent when presented unknown input. This idea is one of the fundaments of the Artificial Intelligence (AI) field of research. An early approach for intelligent agents was to implement algorithmic solutions specifically tailored to solving single tasks, but as computers became more powerful new techniques like machine learning and weighted networks have become more and more popular.

Since computers over the last few decades have achieved greater and greater performance, new ways of approaching the idea of computerized source separation of audio have emerged. One of the most promising techniques today uses Non-negative Matrix Factorization (NnMF), which is a machine-learning algorithm capable of generalizing patterns. By feeding this algorithm sound data, it will learn general patterns in a model, and it will be able to use these patterns to classify which sources the different sounds in a soundscape comes from. The goal of this project has been to explore this technique in an attempt to see if pruning of unfit patterns can improve a learned model.

An algorithm was implemented for source separation using Python and Csound. This algorithm was then trained to learn partial models for four different instrument classes. Using the model, source separation would be done on various combinations of instruments mixed together during a verification test. The test would be repeated for several different parameters (including pruning threshold), and the performance was recorded after each test.

It was found that under certain circumstances, pruning with a low threshold on classification accuracy could result in a modest improvement. In other circumstances the pruning has little to no effect, or negative effect if the threshold is set too high.

## Significance

The idea of digital source separation may have potential impact on several fields in music technology. Isolation of sources is an important consideration in music production and audio engineering today, and it is mostly done through means like the use of static filters, physical separation during recording, directional microphones, and isolated recording sessions. A good digital source separation algorithm may either serve as accompanying these techniques, or eventually replace some of them in certain situations.

### Potential

For a recording studio, it is desirable to perform the recordings in places with as little as possible background-noise, something which may be a great challenge to achieve. Because of this, studios use a great deal of resources and time to provide sufficient noise-immunity during recording sessions.

If digital source separation could be done well enough, it would serve as potentially replace or accompanying some of the established techniques for source isolation. This way it serves the purpose of lowering the amount of noise and crosstalk in a recording. A side effect of this, is that performers may play in a more natural setting, rather than being physically isolated from the rest of the performers.

One fact that should be noted: the majority of the most promising source separation models are based on technologies using models mimicking or estimating an instrument or voice. (Cano, FitzGerald, Liutkus, Plumbley, & Stöter, 2019) This may come in the way of the aesthetic in a purely recording session. Both performers and producers may therefore potentially be reluctant in adapting the technology for recordings used in productions, in particular for more traditional genres.

These aesthetic choices might not be as relevant in other genres, like rap, popular and dance music, where the use of sampling is a common practice. Source separation could greatly improve the separation of samples from recordings not originally intended for further use. This include already produced and mastered music. Since this is done with premade recordings, physical means of source separation is not an option, and the only possible solution is to use a filter.

Other sides of music technology where source separation may be beneficial, is for generating monitor-streams during live performances. The most important aspect of monitoring is for the performers to hear themselves in relation to the rest of the orchestra. In a live setting, physical means of isolating the sound of each performer is limited, which can lead to unintentional crosstalk. With good digital source separation, a group of musicians may record a live rehearsal or performance with fewer microphones, then separate each instrument for monitoring, additional mixing, or review afterwards.

## Technology-Dependencies

Digital source separation is founded on the use of electronic recording-technology, computer technology and digital signal-processing theory. The most promising algorithms explored today also incorporate elements from the field of artificial intelligence and machine-learning.

The first device to record audio was L. S. Martinville's Phonautograph in 1857. The instrument was intended as a tool to visualize the audio waveform, and it was not before some decades later in 1877 that the Phonograph, a device capable of both recording and playback, would be demonstrated by T. Edison. These early examples of recording-technology were based on purely non-amplified mechanical techniques, significantly limiting the possibility for later processing.

L. d. Forest introduced the Audion in 1906, a device which would later show its potential as a means to amplify electrical signals. Much earlier in 1861, P. Reis is credited with building the first electrical telephone, a device including an electrical microphone and a speaker. With the conjunction of the two technologies, electrical amplified recording and playback was made possible during the 1920s. (Millard, 2006)

Early examples of digital systems include devices operating on punched cards, like several semi- or fully automatic looms from the 18th and 19th century, as well as early tabulation-machines. C. Babbage is credited with the idea of the first mechanical digital computer, the Analytical Engine, despite the construction of this device never being finished before his death in 1871. Progressively between the 1850s and the 1930s, several contributions in mathematics laid the theoretical foundations that would formalize the digital computer and establish theoretical computer science as a separate branch of mathematics. (O'Regan, 2012)

Several electrical digital computers were developed and built between 1940 and 1950. These machines were big and had a very high power-consumption, but with advancements in technology the following decades the size of computer would decrease in combination with increased performance and efficiency. A tool for computer-generated audio was made in 1957 by M. Mathews at Bell laboratories, and in 1967 NHK began experimenting with digitally recorded audio. It was first in the second half of the 1970s that digital computers had gained the performance and the accessibility necessary for them to be used as tools for professional music-production. (Boulanger, 2000)

Digital signal processing fundamentally builds on the digital regulation-technology and control systems of the 40s. C. Shannon used control systems to pioneer information technology in 1948, and his work would later be expanded upon to form the field of digital signal processing. (IEEE History Center, 1998)

The research-field of artificial Intelligence was established through a dedicated academic conference arranged by J. McCarthy in 1956. During the following decades, research was primarily focused around intelligent agents, in particular agents taking seemingly reasonable actions when presented un-predicable stimulus. In the first few decades, the algorithms explored were often tailored for particular task, limiting the possible types of stimulus and actions, and in the mid-70s, this particular line of thought was being exhausted. General-purpose artificial intelligence would never be achieved using the traditional line of thought and the field fell out of favor for some decades. However, as computers grew more and more powerful through the 80s and 90s new resource-demanding approaches based on machine-learning and general-purpose models were becoming a possibility leading to a resurge of interest into Artificial Intelligence after the turn of the millennia. (Russell & Norvig, 2010)

Based on the improvement in manufacturing-processes, G. Moore predicted in 1965 that the transistor density of integrated circuits would double every two years for a decade to come. This prediction kept going for several more decades, and as a result the performance limit of computers using integrated circuits would increase accordingly. Today, fast computers with sufficient performance unifies all the dependent technologies that makes digital source separation possible, and it enables them to work together on readily available personal computers.

In 1999, it was shown that computers could reconstruct images of faces from learned patterns using NnMF (Lee & Seung, 1999). A few years later it was shown that musical audio could be analyzed and transcribed using the technique (Smaragdis & Brown, 2003). Since then there has been some experimentation with enhancements using multi-channel and by incorporating other techniques such as neural networks (Cano, FitzGerald, Liutkus, Plumbley, & Stöter, 2019).

## Potential of unexpected results

Music technology has a major focus on developing or utilizing new technology to for making music. In turn new technology and new tools changes the norms in how music is created and consumed. This symbiosis is an important driving-force behind how music, and in particular popular music, is perceived today.

As a developer, it is important to consider the demand for new tools, devices, and solutions. Familiarity with established techniques and equipment might render new technologies that diverges too far from the norm unfavorable for the vast majority of the market, but if the benefits of the new technology is sufficient it might break through and change the established norm.

There have been several cases where moderate and safe innovations within music technology have attracted more attention when used outside their intended purposes for radical effect. Some examples include intentionally driving guitar amplifiers into distortion for a thicker sound, cutting out speaker-cones for other distortion-effects, and using autotune with instant retune-speed to create robot-like voice. These are all examples of established tools introduced for convenience in line with existing norms at the time, but when used in unintentional ways will create unexpected and radically different sounds.

Unexpected results can arise from several factors. In the case with auto-tune, the designer left in the possibility for choosing unintentional settings. Other times, in cases such as "circuit-bending", the end user deliberately goes out of their way to change the internal workings of a device to achieve interesting results. In the case of digital effects, changing the internal workings is not practical and unintentional settings or applications for the effect remains as possible ways to experiment.

It is unknown if digital source separation may produce unexpected results, but some of the implementations are tailored for very specific instruments and might act in undefined ways if used with anything else or effects that greatly affect the source audio.

# Theory

This section will describe the most significant part of the theoretical foundation behind the project.

To understand how source separation is done in nature, it is important to have an overview of fundamental psychoacoustic concepts. This will give a clear picture of why it is necessary to use the Fourier-transform to convert the input audio data into its frequency spectrum before letting an artificial intelligence agent process it further.

Since the main task of the project is to classify and separate patterns in data, it is important to have a general understanding of how machine-learning works, and how machine-learning is related to artificial intelligence. This includes the difference between supervised and unsupervised learning, and how to verify the performance of a machine-learning based intelligent agent after training is done. To understand the implementation of the algorithm, it is also necessary to have a good understanding of the particular machine-learning algorithm used: Non-negative Matrix Factorization.

## Psychoacoustics

Psychoacoustics is the science of how the body and mind converts sound as waves in air-pressure to neural signals, as well as how these are processed by the brain. Most animals have a system consisting of two ears, which each converts sound in the form of pressure-waves in air, into neural signals with frequency-information going to the brain. The brain then interprets the received stimulus and sound is perceived.

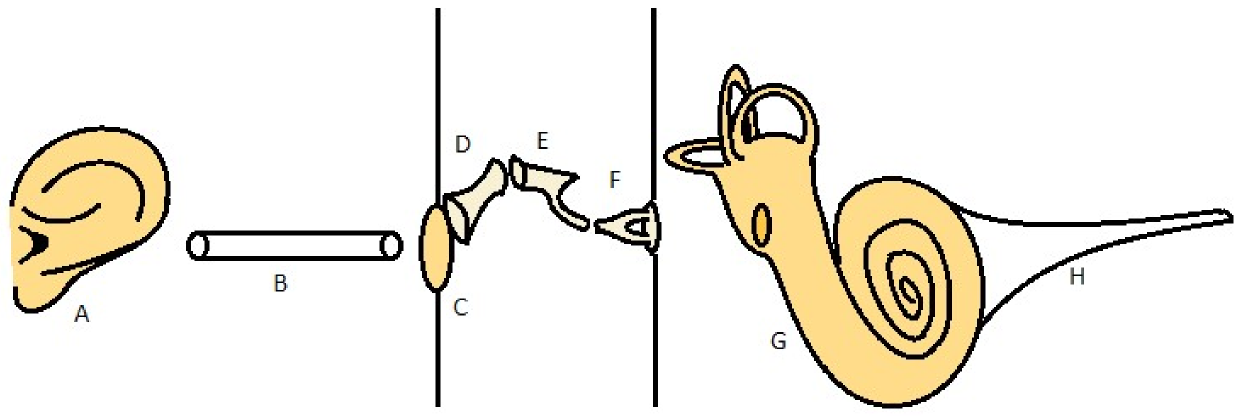

*Figure 1: Model of an ear divided into outer, middle, and inner sections. A: Pinna, B: Auditory Canal, C: Eardrum, D: Malleus bone, E: Incus bone, F: Stapes bone, G: Cochlea, H: Auditory Nerve. The outer ear is exposed to free air, while the middle ear is exposed to the oral cavity through a narrow canal.*

Humans have ears which works in much the same ways as the ears of other mammals, as seen in Fig. 1. Air-pressure waves hits the Pinna, and travels down the auditory canal. The difference in pressure between the eardrum and the middle-ear will cause the eardrum to stretch and oscillate in line with the pressure-waves, putting the malleus, incus, and stapes bone in motion. These bones serve as a means to transfer energy from the eardrum to the cochlea. In addition, the bones help transforming the low mechanical impedance of the eardrum to better match the relatively high mechanical impedance of the fluid inside the cavity of the cochlea. In the cochlea the different frequencies of the audio signal resonate at different locations down the spiral tube, activating different sensors which trigger different sections of the auditory nerve. (Bosi, 2003)

## Fourier Transform

The Fourier-transform is a mathematical transform for calculating the complex frequency-amplitude spectrum of a time-amplitude signal. It stems from the idea that every waveform can be decomposed into a series of sinusoidal components of different frequencies and phases.

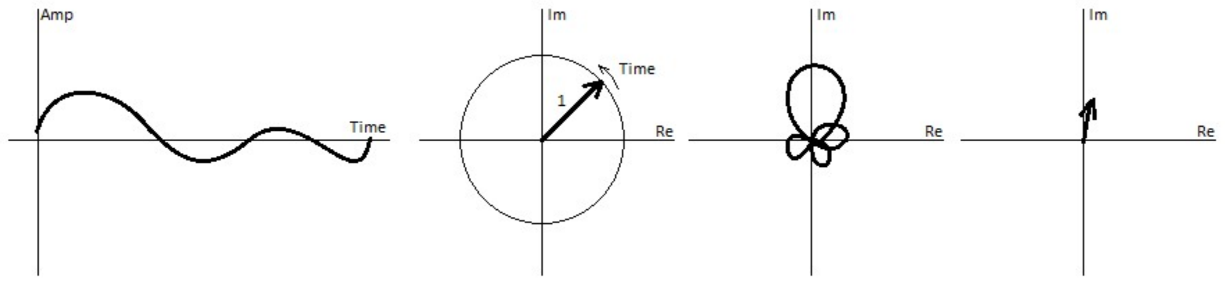

*Figure 2: Visual representation of finding the frequency component of one particular frequency in a time-amplitude signal. From left to right: Time-amplitude signal, Unit phasor rotating at the "probed" frequency, product of the phasor and the time-amplitude signal, and vector resulting from the integral of the product. Magnitude and phase of the vector represent the frequency component of the given frequency of the original signal.*

A visual way of viewing the Fourier transform can be seen in Fig. 2. It is useful to envision a unity phasor rotating around the complex plane at some constant frequency. The time-amplitude signal is then multiplied with this phasor, thus the signal is wrapped around origo at the given constant frequency. Any component of the signal that matches the frequency (the rate of phase-change) of the phasor will therefore collect on one particular side of origo, while all other components will drift around origo due to the mismatch of phase change. To find the final component of the given frequency in the input signal, the definite integral of the wrapped-up signal is taken and the resulting complex vector represent the magnitude and phase of the particular frequency component. When taking this integral, the unmatched frequency components will cancel out due to their contributions drifting symmetrically around origo, while the frequency components matching the frequency of the phasor will stay put and add up (Sanderson, 2018). Since the result of the transform is a complex function, the whole spectrum is often split into a magnitude and a phase part. Time-delays in the signal will affect the phase but not the magnitude, and the magnitude spectrogram is therefore usually used alone to represent the frequency spectrum of a signal.

$$F_{(\omega)} = \int_{-\infty}^{\infty} f_{(x)}\, e^{-ix\omega}\, dx$$

*Formula 1: The Fourier transform F(ω) of the time-amplitude function f(x). x is time in seconds, ω is angular speed in radians per second.*

For a signal with a known mathematical function, the function for the Fourier transform can be found using calculus as seen in Formula 1. This function reflects the visual representation, with the integral of a time-amplitude signal multiplied with a phasor. When dealing with digital discrete signals, which has been sampled and quantized, the signal is no longer a continuous function and the discrete Fourier transform seen in Formula 2 has to be used instead.

$$F_{[n]} = \sum_{x=0}^{N-1} f_{[x]}\, e^{-ix2\pi\frac{n}{N}}$$

*Formula 2: The Discrete Fourier Transform F[n] of a frame f[x] of a discrete time-amplitude signal. N is the total number of samples in one frame, n is frequency-bin number, and x is the sample number within the particular time-amplitude signal frame. The center-frequency of one particular bin is n/N Hertz, or 2πn/N radians per second. Note that the transform from this particular formula is the non-normalized version of the discrete Fourier transform.*

The discrete Fourier transform works in a similar way to the non-discrete version, except instead of an integral over a continuous function, a sum is used instead to account for the discrete samples. To preserve all of the information from the original signal, the frequency domain of the transform will be divided into as many equally sized frequency-bins as there are samples in the time-amplitude signal-frame being transformed. If high frequency resolution is desired, a large frame is necessary causing the time-resolution of the time-spectrum to decrease. If a high time-resolution is required for the time-spectrum, the frame needs to be shorter, which compromise frequency resolution. (Proakis & Manolakis, 2006)

$$f_{[x]} = \frac{1}{N} \sum_{n=0}^{N-1} F_{[n]} e^{ix2\pi \frac{n}{N}}$$

*Formula 3: Inverse Discrete Fourier Transform, taking a discrete Fourier transform F[n] and returns the time-amplitude signal f[x]. The source transform is a complex spectrum where every frequency-bin has both phase and frequency. N is total number of bins in the transform, n is one particular frequency-bin in the transform, x is a particular sample in the resulting time-amplitude signal. Note that this equation assumes a non-normalized discrete Fourier transform, like the transform represented by Formula 2.*

A powerful feature of the Fourier transform is the possibility of converting a signal to frequency-domain, modify it, and then use the inverse Fourier-transform in Formula 3 to changing it back to amplitude-time domain. While the magnitude spectrum alone is very useful for comparing the frequency content of different signals, the phase spectrum becomes crucial as well for recovering transients and the original waveform when doing an inverse transform.

## General about Artificial Intelligence and Machine Learning

While it can be easy to think the main motive for Artificial Intelligence (AI) would be to incorporate full general-purpose intelligence or some kind of digital self-consciousness, an AI is rather merely an automatic agent that acts upon its environment in a seemingly intelligent way as a response to arbitrary stimulus. This can be described by the setup proposed by the Chinese Room analogy: A closed box with a tray for dropping off papers with Chinese symbols on them, and a tray for retrieving papers which will contain Chinese symbols written by the box. Inside the box is an individual who does not understand Chinese, but keeps a rulebook telling what to do when the different symbols and patterns are encountered. The box as an entity can act in an intelligent manner in Chinese, but the operator inside the box does not understand Chinese and only operates based on the given rules in the rulebook (Russell & Norvig, 2010).

An intelligent agent receive stimulus from its environment through its sensors, and act upon its environment using its actors. Internally the agent may have varying degree of complexity, from simple algorithms hardcoded to a certain environment to more complex flexible algorithms based on learning. Which type of process is best to use, depends on what assumptions one can make on the environment. An environment with non-deterministic behavior and lots of unknown rules, a flexible complex process may be necessary for the agent to adapt. If the environment has more rigid and well defined rules, the same amount of flexibility is no longer needed for the intelligent agent to appear intelligent and a less complex process with hardcoded assumptions can be used.

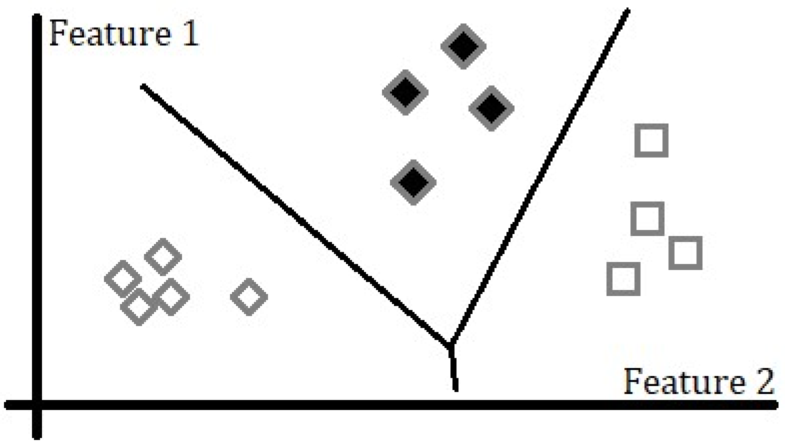

*Figure 3: Coarse example of classification based on feature-data. Two features classify the current environment into one of three classes based on observed patterns. The different shades and shapes are used to distinguish the class of the distinct observations.*

Feature extraction is one of the most important steps of an AI, be it an agent based on machine-learning or not. Extraction is done by observing sensors and using some means to quantify meaningful features. As demonstrated in Fig. 3, the current environment can then be classified, by comparing observed features with memorized or hardcoded patterns. In the case of a continuously learning model, data may either strengthen the already memorized data, challenge it, or in some cases just be evaluated for an immediate action.

In situations where the rules of the environment are unknown, the intelligent AI must rely on learning in order to work around different situations which might occur. Learning relies on memorizing features, and if consequences or hidden states have to be considered then some sort of reasoning analysis has to be done on the memorized patterns. In all cases where a learning intelligent agent is used, the learning may happen in advance, but some agents, for example neural network models, can learn continuously while being in application.

## Supervised, Unsupervised and Reinforced Learning

There are two main approaches for doing machine-learning. These are supervised, and unsupervised learning. Both approaches cover their own set of tasks and models, and which one to use depends on the application.

For supervised learning, the model is presented sensory examples along with predetermined classification. This gives a controlled learning-environment, where the agent is told what is expected when learning. Models using supervised learning are often used to predict borders in the feature-data between classes, as seen in Fig. 3, or to use regression to predict feature trends within classes. By using supervised learning, an agent can therefore be taught how to efficiently take a reasonable action dependent on previous knowledge (Russell & Norvig, 2010).

A downside with supervised learning, is that once the agent is deployed it loses the ability to keep learning. In a working environment, no correct classification data is provided, and the agent must solely rely on its learned knowledge in order to act in an intelligent manner. This leads to several shortcomings. Primarily, the agent will not be able to adapt to parts of the environment which it has not been covered by training. It may try to deal with an unknown situation based on what it knows, but it is not guaranteed to act reasonable if the situation is too far from anything in the training-set. An example would be if an AI taught to identify different species of animals all of a sudden was presented a picture of a teapot.

Some flexibility is necessary in the training set to avoid overfitting. Overfitting happens if data used for training is too similar and does not reflect the variation of the data in the environment as a whole. When overfitted, the model will be very fit to identify the dataset itself, but anything not in the dataset will stand a high chance of being miss-identified, thus the model being prone to false-negatives. It is therefore important to ensure some degree of diversity of the training data.

Underfitting is also a potential problem. If underfit, the model will not have good enough information to differentiate between classes, leading to a higher chance of false-positive classifications. Reasons for underfitting is in general too little training-data, as well as a high degree of overlapping feature-value combinations between classes in the training data.

Unsupervised learning works by having the AI continuously learn without the classification data present. Instead of employing hard classification limits, the agent needs to use the distributions of observed feature-data in order to group together the denser parts of the distributions using clustering-techniques. The agent will be able to group together similar observations and relate them to actions, but it will not have any knowledge or context for classifying the observations as anything in particular. Other challenges if the learning happens continuously, is the large amount of data that may have to be memorized. However, a benefit for unsupervised learning, is the ability to continuously adapt to unknown situations.

One popular means for a learning algorithm, is to use the reinforcement learning technique. In this technique, a measure for fitness is used by the agent to evaluate itself during learning. The fitness is a quantitative measure for how close the model is to some ideal behavior, and it is desirable for the training algorithm to optimize the model for best possible fitness.

## Matrix Decomposition and Non-negative Matrix Factorization

In linear algebra, matrices were arrays originally used to perform mathematical operations on sets of linear equations, but some of their mathematical properties make them very useful for data-representation and feature-extraction as well.

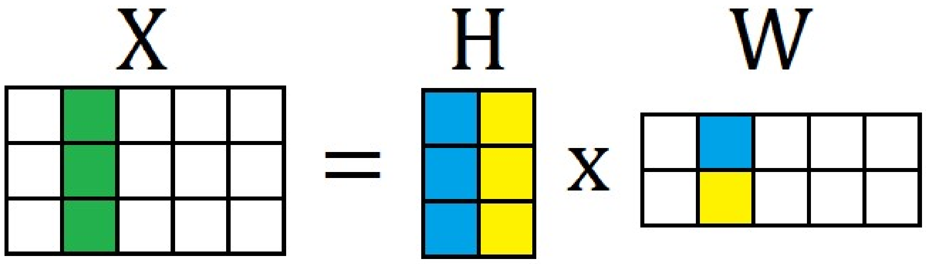

*Figure 4: Visualization of how a matrix X with data can be decomposed into a feature matrix H and a matrix W of feature weights. A column in H corresponds to a feature, while a column in W corresponds to the feature-weights which are needed to recreate the same column in X. The color of the particular cell in W represent which feature in H the cell is weighting.*

When used for feature-extraction, the input sensory data is first represented as an array of fixed points (X). Then this array is decomposed into two matrices, one for feature-data patterns (H), and one for pattern weight (W). The weights determine how much of each pattern needs to be added together to reconstruct the input-data, as demonstrated in Fig. 4. The original input X can then be reconstructed by multiplying the W and H matrices together.

Models using matrix decomposition as a means to extract feature patterns, uses machine-learning to build the feature-pattern matrix in advance. The learning-process will fit the list of patterns to the training-set, such that the training-set can be reproduced as closely as possible with the limited number of patterns. After the feature-matrix is built, it is locked and serves as a dictionary of available feature-patterns. When the model using such a dictionary is put into application, only the pattern-weight matrix is changed to fit, thus the model will be finding the combination of pre-learned patterns from the dictionary that will most closely mimic the input data. This operation is in effect a transformation which decomposes the input data into a set of weights, one weight for each pattern (Makino, 2018).

The Non-negative Matrix Factorization (NnMF) is a particular type of Matrix Decomposition. A particular benefit of NnMF, is that all arrays are restricted to only contain positive values. This way the learned patterns will not intervene with each other in ways that cancel out, and the learned patterns will to a larger extent resemble partial sections of data as it might occur in the input data matrix X.

## NnMF in Source Separation of Audio

A Non-negative Matrix Factorization model is often used to perform feature extraction of audio, in particular in conjunction to source separation. The main concept behind the appeal, is that after a transform, isolating the contribution of some single feature-patterns is as simple as nulling out the weights of all other patterns in the W matrix. By classifying learned patterns and isolating these after a transform, single instruments can potentially be separated from a complex soundscape (Cano, FitzGerald, Liutkus, Plumbley, & Stöter, 2019).

For the patterns to have any particular meaning, it is necessary to have them classified by audio source. Because of this, the NmMF feature-pattern matrix is trained using supervised learning. This way each pattern of features can be associated to its respective source, and isolation of individual sources can be done automatically using the classification data.

Each pattern consists of a fixed number of features, and it is important that the data-entries of the input data have the same number of features as the patterns in the dictionary. A consistent meaning for all the patterns is important as well. For analyzing audio using NmMF, his consistency is maintained using the discrete Fourier-transform, followed by splitting the magnitude spectrum into chunks (segments) of fixed duration. Every segment is then of equal length, and all the magnitude spectrum values in a segment can be mapped to fixed features in the input-data sample (any one column in the input-data matrix X).

In effect, the patterns learned are partial timbres and details of the magnitude spectrum. The segment length, in time, also determine the maximum size of which details can be memorized. After transforming input data, the resulting W matrix will correspond to envelope for the various feature-patterns in the dictionary.

## Verification

When a model is trained, it starts off with no prior knowledge. It is presented data from a training data-set, which it then has to fit itself to. The model them becomes a hypothesis for the training-set, and to verify the model this hypothesis has to be tested.

The purpose of a test is to test if the hypothesis is accurate in a general environment. High degree of generalization is preferred, as this is an indication that the model has the sufficient flexibility to cover a decent amount of variation.

As the training is done on a training-set, this training-set cannot be used for verification. If the verification-testing were to be done on the training-set, it would not be possible to detect issues arising due to overfitting. A separate test-set must therefore be used for verification. This test-set should present the model with the same type of data as the test-set, but due to it being separate none of the test-data will directly equal any of the training-data.

## Tools and Materials

The software, algorithm and dataset used for the project will be described in this section. The program is mainly run in Python, but it makes use of Csound and some external libraries for the more complex tasks.

### Python

Python is an interpreted high-level general purpose computer programming language. As it is interpreted, the code is parsed during runtime instead of the program being compiled into binary machine-code before being run. By being an interpreted language, code can be run immediately as soon as it is written. This, paired with flexible syntax and versatile built-in features, makes Python ideal for coding trivial utilities and prototyping algorithms.

The first available version of Python was released in 1991 by G. van Rossum, and since then it has seen active development. Versions up to and including version 2.7 are backwards-compatible, while the version 3 series is only backwards-compatible with older editions of version 3. One of the reasons for this break in backwards-compatibility was due to major structural changes in the Python syntax. These changes fixed some issues with loose-type interpretation and the possibility for code ambiguities (What's New In Python 3.0, 2020).

Another of the many appeals of Python, is the availability of solid function-libraries. By clever use of the right libraries, complex functions can be performed with short, concise code with a high degree of code readability. Back-end tasks like memory-management is also hidden from the programmer, minimizing the amount of overhead code necessary to make a program run.

A downside with Python stems from it being an interpreted language. As it is not compiled or optimized as machine-code, it tends to be slow compared to compiled application programming-languages such as C++.

### Csound

The Csound language is a specialized programming language designed for generating and processing audio on a computer. It features built-in frameworks for score-management and real-time signal-processing, as well as being deterministic on a sample-based level. Csound is a compiled language, but the compilation process is often fast and seamless. The name is based on the compiler and framework being implemented using the C programming language, compared to earlier implementations using the Fortran language.

In 1957, M Mathews were doing experiments with creating music using computers at Bell Labs, the tools for this would go under the name MUSIC. Due to the limited performance of the computers at the time, the output audio would be calculated and written to tape a few samples at a time. By 1963, it had reached version IV, a version which was later extended and ported by H. Hove and G. Williams to the system-independent general-purpose programming language Fortran (Roads, 1996). By 1985, B. Vecroe had implemented the compiler in C.

A Csound program consists of two parts, a set of instruments and a score. Each instrument is a small program of op-codes, which is run as long as an instance of the instrument is invoked. The code in an instrument defines its behavior, which may include tasks such as processing signals, controlling parameters, generating audio, and playing back audio. The score is used to invoke these instruments with appropriate parameters and timing.

Unlike general-purpose languages, the instrument-code is executed once per sample. Audio-rate variables being re-evaluated every sample, control-rate variables are re-evaluated every fixed number of samples, and init-rate variables are only evaluated once as an instance of an instrument is invoked. Several instances of the same instrument may be invoked simultaneously, and every instance act as if independent from the other instances. Other special variables like spectral-type variables exist for processing frequency-amplitude data. Variables can be local to an instrument, or global and thus accessible for all instruments.

An important part of Csound is cross-language support. This includes support for calling the Csound compiler from within Python, which enables starting and remote-controlling a Csound performance. One important feature is the ability for Python code to access global variables and tables within the Csound program during runtime.

## Sample-Library

The audio source data is stored in different sample-libraries. A sample library consists of individual wav audio-files, each file containing one audio-sample. The wav file-format consist of a stream of digital audio-data coded in Pulse-Code Modulation (PCM) format.

One particular sample-library corresponds to one particular instrument, thus the classification of the source separation is directly linked to the content of the sample libraries. In the same way, the number of sample-libraries defines the number of classes in the model. In the particular experiment, four different sample-libraries are used.

The first sample-bank is samples of a music box, specifically the Symphonion Model 106N. This sample-bank was chosen to look for effects of potential over-generalization. Symphonions were a class of disk-music boxes of varying sizes released around the start of the 20th century, the Model 106N being a larger type with a 106-teeth comb. Sound is produced by plucking the metal teeth, and the sample-bank has several samples for each tooth. A major factor in the over-generalization and overlapping with other instruments, is due to the music box having a simple timbre. The particular instrument sampled is item RMT-790A at Ringve Museum, and the instrument was sampled in an anechoic chamber at the university for a separate project.

Another included instrument class is the guitar, chosen as a tonal instrument with different timbre than the music box. For this, the IDMT-SMT-GUITAR sample-bank from Semantic Music Technologies group at Fraunhofer IDMT was used. This particular sample-bank contains several datasets aimed at different purposes, where the single-tone samples from dataset 1 were used.

For the third class, the bass-guitar is included. The intent with the bass, is to have an instrument with potential overlap with guitar, while still being somewhat distinct. The IDMT-SMT-BASS sample-bank from Fraunhofer IDMT was used as data for the bass. Sustained neutral single-tone audio samples were chosen.

The last sample-pack chosen was drums. The idea was to have an instrument with very wide variation, a main focus on vertical sounds rather than tonal sounds, thus being very distinct from the tonal instruments. The sample bank used was is the IDMT-SMT-DRUMS sample-pack, as well from Fraunhofer IDMT. The pre-mixed samples were excluded, and only the samples containing a single sound were used.

About a third of the samples in each sample-bank were separated into the test-set, while the remaining two thirds make up the training-set.

## Algorithm

There is a distinction between the program implementation and the algorithm. The algorithm describes the general process of training the model as well as testing and using it, while the implementation describes the particular means used to implement the algorithm. In the following part, the algorithm in general will be described.

The intent of the algorithm is to build on a standard setup of audio source separation using non-negative matrix factorization, then experiment with the model to see if the pattern-pruning technique leads to an improvement. Four parts makes up the algorithm, a preprocessing step, a learning step where the model is constructed, a reclassification or pruning-step, and the verification step.

### Audio preprocessing

An important aspect of source separation is to mimic and replicate parts of the human hearing system. In the preprocessing stage, the goal is to apply the same effect as the mechanisms of the ear. The ear is ultimately an organ that receives amplitude-time air-pressure signals and outputs frequency-amplitude signals to the brain over time.

Much of the work of the ear is already covered by the recording process. Sound pressure waves are picked up by a microphone, and the signals are eventually filtered or further processed before the samples are saved. There is no need to replicate the impedance matching of the middle-ear after the signal is recorded, as the acoustic mechanical portion of the process happens in the microphone itself during the recording.

What remains of the preprocessing, is the transform done by the cochlea. In the cochlea, a membrane resonates to the frequency components of the time-amplitude audio-signal, where different frequencies resonate on different locations of the membrane. Sensors on the membrane picks up the amplitude of the vibrations, and together all the sensors send a frequency-amplitude spectrum to the brain. This spectrum changes over time as the signal and resonance change.

For a recorded signal, a Discrete Fourier-transform will be used to find the complex frequency-amplitude spectrum as a function of time. From complex spectrum, the magnitude spectrum is found using the Pythagorean hypothenuse equation to find the length of the complex vectors.

This serves to a large extent the same purpose as the function of the cochlea, and in the same way results in output of amplitude in terms of time and frequency.

## Representation of Audio Data

When building the model, training-data has to be decomposed into feature-patterns learned by the non-negative matrix factorization model. This pattern-data has to be classified with respective instrument sources as well.

Since a non-negative matrix factorization model is to be used, it is important that the spectrum data from the preprocessing step is prepared to a format compatible with the NnMF. An element in the NnMF is a single array of values, with each value representing a particular feature. It is therefore important that the data presented as training-data uses this format.

The magnitude spectrum can be represented as a grayscale image, a two-dimensional function, where amplitude is represented in terms of pitch (frequency bin) and time (frame number). This cannot readily be used as training-data, as it is not a fixed-length array of features. It should however be noted that the number of frequency-bins is fixed, and this can be taken advantage of. One frame of the spectrum will thus be a fixed-length array, where each value will correspond to a particular frequency bin. This can be used as input to the NmMF, where the frequency-bins are used as features.

Having the patterns learned by the NmMF limited to one frame of the spectrum can be very limiting. This prevents the model from learning any patterns greater in length than a single frame. The “asymmetric sampling in time” hypothesis (Poeppel, 2003) suggests that the brain processes sound features of different lengths in different ways simultaneously. Shorter patterns of 20mS to 40mS, like syllables, and longer patterns of 150mS to 250mS, like vowels. The processing for shorter features is suggested to focus on identifying and separating transients, while the processing for longer patterns is suggested to focus on identifying variations in pitch. In the case of using the frequency-spectrum information for source separation, the model must be able to cover the learning and recognition of longer patterns. A way of doing this, while still complying to the format of the NmMF is to take the spectrum of a fixed number of consecutive frames, a spectrum segment, and append the spectrum of one frame after another into one array. This “repackaging” will form an array where the features from consecutive frames are arranged in one single dimension instead of two (bin number and frame number).

When appending the spectrum of consecutive frames for the data in the model, retrieving data from the model will return an array with a similar structure. This array needs to be split back into the individual consecutive frame-spectrums, and then arranged as a function of time to form the corresponding two-dimensional spectrum segment. If a signal is run through the model in segments like these, all of the output spectrum segments needs to be combined at the right points in time to form the full magnitude spectrum of the output.

## Building the Model

The main goal of the training-process, for building the dictionary, is to fit the model to the spectrum segments in the training-data. The patterns in the dictionary also needs to be labeled with appropriate classification.

One of the key features of the NnMF fitting process, is to derive at a set of patterns to best reproduce all the examples in the training-set. The number of patterns is a lot fewer than the number of example patterns, and this forces the algorithm to generalize. This process, however, is not by itself a supervised learning mechanism and uses clustering for deriving the patterns. In order to make sure the learning of the model is supervised, class-information has to be assigned to learned patterns in parallel to merely fitting the training-data to a NnMF model.

A particular way supervised learning can be achieved with NnMF, is to separately fit each instrument to a portion of the entire dictionary. By doing this, patterns can be classified with the instrument used to train the respective portion of the dictionary. A simple way of doing this, is to isolate each portion of the full dictionary as separate smaller dictionaries during the training. After all dictionary-portions have been trained and fitted to their respective instrument training-data, the separate portions can be combined into a single dictionary along with matching classification data.

Every pattern in the model is at last normalized. This will not affect the model, as the shape of a pattern is not affected by a scalar change in magnitude. The normalization is necessary for later steps, as it serves to make the dictionary independent from pattern loudness.

## Statistical Re-classification (Pruning)

After building the dictionary, the NnMF model can be used for source separation tasks. However, one weakness of using the technique for training each instrument independently, is that some patterns may overlap between several instruments.

As a means to attempt to improve the model, an additional re-classification step is performed to prune the patterns which are potentially not unique to a single instrument. In order to do the pruning, the classification probability (instrument utilization factor) for every pattern in the dictionary must be found.

For finding the classification probability for the patterns, the first step necessary is to use the NnMF model to transform the training data. Every spectrum-segment from the training-data is run through the model. The transform will return a W matrix with weights for each input pattern. The values in a column in the W matrix will give the dictionary-pattern weights needed to recreate the pattern of the corresponding spectrum-segment from the training data. Each row in the W matrix will on the other hand tell how much each input spectrum-segment utilizes a particular pattern in the dictionary. It is this row-data which is of interest when finding the ratio between which instruments a dictionary-pattern is used by.

It is to note that for one pattern in the dictionary, the ratio of utilization by all the input segments of a particular class will be collected and evaluated. Because some input spectrum segments have overall higher amplitude and more energy than other segments, it is desirable to normalize each spectrum-segment to equal loudness. This can be done directly in the W matrix by normalizing the weights in each column. Since the loudness of the patterns in the dictionary are normalized in advance, normalized weights per column in W ensures that the loudness of each segment will be similar if the W matrix is inverse-transformed.

When all the columns are normalized, the instrument utilization of a particular dictionary-pattern can be calculated by taking the average of all weights corresponding to the given instrument class and dictionary-pattern in W. This will be a selection of values from a row in W, the columns matching the segments of the instrument class and the row matching the dictionary-patten. After the average utilization of a pattern is found for all instruments, the instrument utilization ratio can be calculated.

A dictionary-pattern distinct to its class and with high classification accuracy, should have a high instrument utilization ratio for the instrument that matches the instrument class of the pattern. Patterns which are not distinct and overlaps with patterns in other instruments should on the other hand have a ratio more evenly divided between several instruments.

## Verification Test

The purpose of the verification is to get a measure of how well the model performs in a controlled application-type setting. It is important that the verification is done on the test set and not the training set. To verify the model, several tests are run. Each test probes the performance of the model with a given combination of instruments, and every test has a different combination of instruments.

One test uses a stream of mixed samples for each instrument involved. There is one stream for each instrument, as well as one stream with all the individual streams mixed. By transforming the mixed stream through the model, a matrix of weights can be obtained. The weights of each instrument can be isolated, and these can be used to make source separated streams by separately inverse transforming the isolated weights. In the same manner, weights from patterns with an instrument utilization rate below a certain threshold can be zeroed out before the source separated streams are inverse-transformed back and combined into full magnitude-spectrums.

When the magnitude-spectrum for a source separated instrument stream is ready, it is compared with the magnitude-spectrum of the original unmixed instrument stream. The ratio between the error and the total amount of energy in the spectrum determines the performance of the model. This performance is the performance in the situation of separating the given instrument from the other instruments used in the test, and more tests must be done to get a more general performance score.

During a full verification, it is reasonable to run several tests covering every combination of a given number of instruments multiple times.

## Implementation of Program

The overall implementation of the program is done in Python, with the use of various libraries, including Csound integration. A single main menu is used as the user interface, where the user can invoke actions by selecting one out of ten operations. Some operations perform parts of the algorithm, other operations are for saving or recovering model data.

It is to be noted that certain parts of the algorithm may require several operations performed in sequence. A benefit of such an approach is that it enables the user to save data at intermediate stages of the experiment, avoiding the need to repeat time-consuming preliminary operations. Internally, operations are again divided into smaller functions hidden from the user. These functions may be more general, and one function may potentially be usefull for several different operations.

Each instrument class is given equally many patterns from the overall dictionary. This number of patterns per instruments must be set correctly in the program at the start of the process. Setting this number will clear all other loaded data and reset the model.

For the particular implementation used, Audio-data is in the form of wav-files divided into folders. Training-data for learning separate instruments is stored in one folder per instrument, while the statistical re-classification test expect all the files in the training-set to be stored in the same folder. The test data for the verification also expects all the files to be stored in one single folder in a similar manner. When one folder contains files from different instruments, the first character in the filename represents the instrument class of that particular file. It should be noted that for this particular implementation, since the filenames explicitly defines what instrument-class audio-samples belong to in some of the tests, care must be taken when chosing which instrument slot is used when training for a new instrument or loading previously trained instrument data.

The Non-negative Matrix Factorization algorithm used, is from the the SciKit-Learn plugin (F. Pedregosa, 2011). Default settings were used, except for using a custom number of patterns in the model.

## Loading Audio Data

Several operations require loading audio data for the model. This is done using Csound-integration, as seen in Fig. 5. The filename is added to a Csound script, which is loaded and run. The script returns the complex spectrum for the audio-file, which is divided into a phase and magnitude spectrum in Python. The magnitude-spectrum is then divided into spectrum-segments. In the case of data from several files being loaded from the same instrument, the segments are all added to the same array.

For the sake of maximizing variation of data, adjacent segments have N-1 frames overlap, where N is the number of frames in a segment.

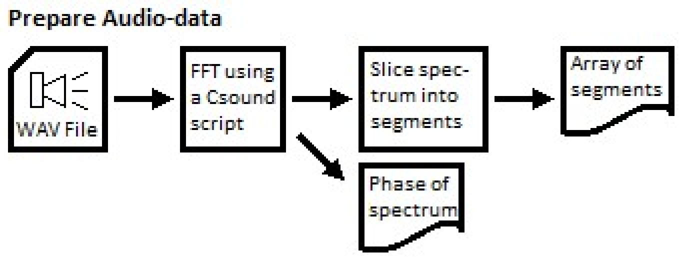

*Figure 5a: Function for preparing audio data. The preparation of audio data will result in an array of magnitude-spectrum segments. One segment consists of a set number of adjacent FFT-frames from the full magnitude spectrum.*

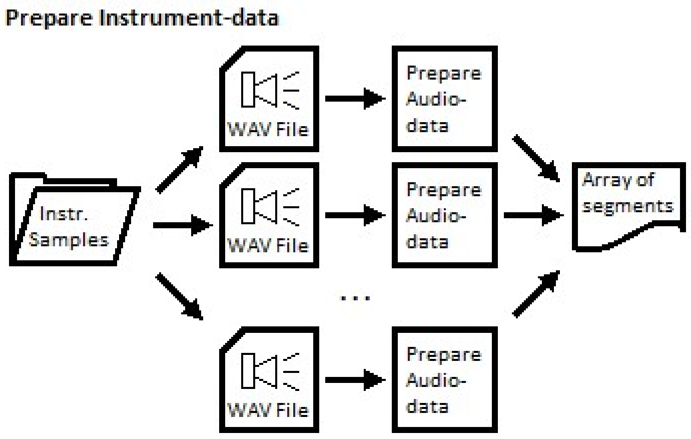

*Figure 5b: When preparing multiple files from the same instrument, in the case of training a model, all spectrum segments from all the audio files are collected in one dataset.*

## Operation 0, Set Parameters and Reset

This operation is used to set parameters without having to manually change the program. After parameters are set, the model is reset, and the state of the program is reset.

## Operation 1, 2, 4, Source Separation

It is possible to manually invoke source separation of an audio file, using the currently loaded model. This function is mainly intended for demonstrating the model in operation, as it gives the user the ability to use the model directly in a non-test setting.

As seen in Fig. 6, the source separation is a simple process consisting of loading the audio data, transforming, isolating the pattern-weights for the instrument class to be separated, inverse transform the isolated weights to get the spectrum segments of the source separated audio, combine the segments by averaging the overlap to get the full magnitude spectrum, and finally inverse Fourier-transform the spectrum to get the final source separated audio.

The original phase-spectrum is used as a best-bet solution together with the source separated magnitude spectrum in order to recreate the complex spectrum needed for the inverse Fourier-transform.

There is not much difference between the three operations presented, one operation only transforms and inverse-transforms the input without any source separation, another operation source separates an instrument based on classification but without any pruning, and the final operation source separates an instrument based on both classification and pruning.

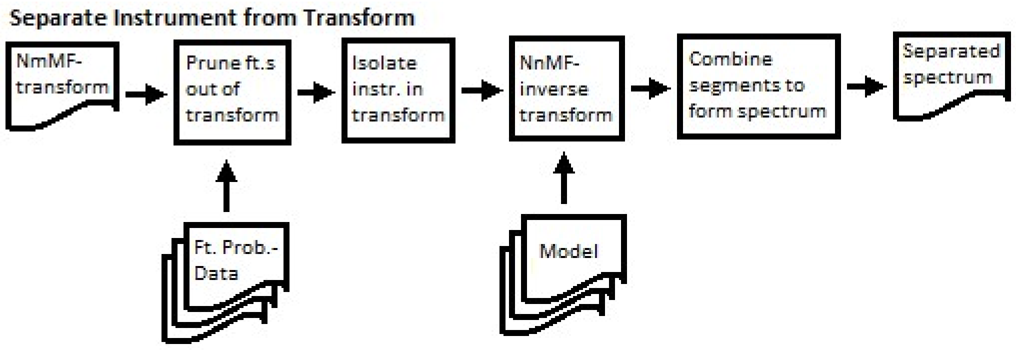
Separate Instrument from Transform
NnMF-transform
Prune ft.s out of transform
Isolate instr. in transform
NnMF-inverse transform
Combine segments to form spectrum
Separated spectrum
Ft. Prob.-Data
Model

*Figure 6a: Function for source separation of sound already NnMF-transformed. The input is the W matrix from the NnMF-transform, containing weights of all the patterns for every spectrum-segment. All weights which does not fit the classification of the instrument to be separated, are nulled out in the weighting-matrix before the NnMF inverse transform.*

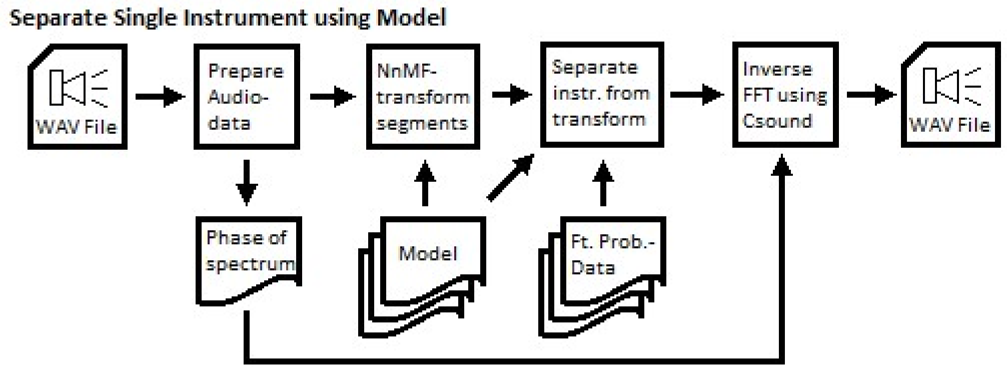

*Figure 6b: The full function for source separation of an audio-file. Audio is prepared, NnMF transformed using the model, source separated, then inverse-transformed back into audio.*

## Operation 3, Statistical Test

The reclassification part of the algorithm is performed by this operation. It requires a model to be learned or set up in advance, and it will produce the instrument-utilization ratio for all the patterns in the currently loaded model.

In order to prevent running out of memory, the transform and pattern-weights are collected from 100 source-files at a time. When data from all files in the training-set has been obtained, the ratio is calculated per the description in the algorithm. The instrument utilization ratios for each pattern is stored in a separate “Feature (pattern) probability table” matrix within the program.

Fig. 8 represent the entire learning and reclassification step, where the statistical test operation covers the three blocks in the lower right corner of the chart.

## Operation 5, Correlation Test (Verification)

One of the most important parts of the program is the verification test. This test requires both a model and matching instrument utilization statistical data to be loaded or set up in advance.

The test-setup is a two-part system. One part, as represented by Fig. 7, is one individual test, where a certain combination of instruments is tested. The other part is a process that invokes a test for each instrument-combination a given number of times, and then collects the result when a particular test is over. This part prints out the overall results after all combinations have been tested as well. For each instrument combination, the average and standard deviation for each measure in the test is printed, as well as overall average and standard deviation per instrument.

One particular test can have a combination of two or more instruments, depending on the settings. All managing of input audio is handled with a Csound patch, where a random selection of audio-samples is played for each instrument. Each instrument has its own audio-stream, but Csound also mixes the individual streams together in a separate mixed stream. Then the Csound script takes the Discrete Fourier-transform of all the streams. Each instrument stream will have the same number of audio-samples, but the particular samples played will be random within the respective instrument class. The times at which the audio-samples are played is random as well.

The mixed stream goes through a standard source separation process as represented by Fig. 6a, where some separated streams are recovered for every instrument in the mix. These streams are then compared to the unmodified stream for the respective instrument. The average of the absolute difference of each bin in every frame is found, and this gives the correlation measure used to evaluate how close the source separated stream is to the original.

For each instrument, the five streams compared to the original unmixed instrument stream are:

1. The original spectrum of the mixed un-processed stream
2. The spectrum of the mix after a NnMF transform and inverse transform
3. The spectrum of the source separated instrument using classification only
4. The spectrum of the source separated instrument using both classification and pruning
5. The spectrum of the source separated instrument using only pruning.

Comparison of measure nr. 1 and 2 is intended to show to what extent the performance is deteriorated by the NnMF transform. Comparison between nr 2 and nr 3, 4 and 5 will give information on to what degree classification and pruning improves the transformed mix.

The test setup can be seen in Fig. 7, but it should be noted that only one comparison (for measure nr. 4) is shown for clarify. The other measures will be found by similar comparisons, but with selected parts of the signal-path omitted.

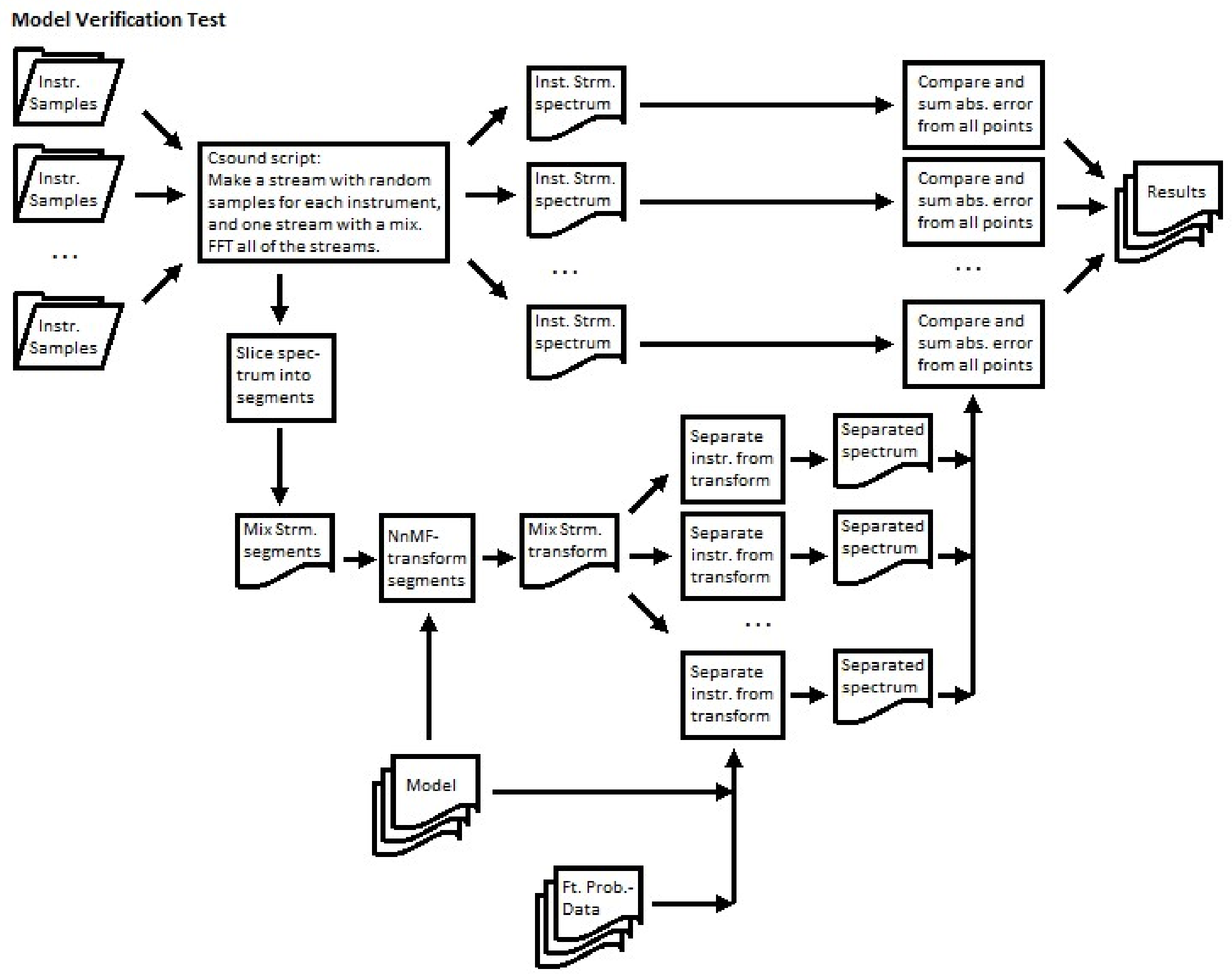
Model Verification Test
Instr. Samples
Instr. Samples
...
Instr. Samples
Csound script: Make a stream with random samples for each instrument, and one stream with a mix. FFT all of the streams.
Slice spectrum into segments
Mix Strm. segments
NnMF-transform segments
Mix Strm. transform
Inst. Strm. spectrum
Inst. Strm. spectrum
...
Inst. Strm. spectrum
Compare and sum abs. error from all points
Compare and sum abs. error from all points
...
Compare and sum abs. error from all points
Results
Separate instr. from transform
Separate instr. from transform
...
Separate instr. from transform
Separated spectrum
Separated spectrum
Separated spectrum
Model
Ft. Prob.-Data

*Figure 7: Verification test setup. A CSound scrip creates streams with random audio, one stream with audio from one given instrument. A mix of the streams is run through source separation, then the separated instrument steams are compared to the original unmixed streams.*

## Operation 6, 7, 8, Save and Restore

As learning a model and finding the statistical instrument-utilization ratios for the model is time consuming, the program has the ability to save and restore this data. This makes it possible to reuse model and statistical data across several experiments which uses the same setup.

The data is saved as text-files, each text file containing a two-dimensional matrix of floating-point values. This makes it easy to parse the data from other programs as well, for example for the purpose of visualize some of the model data.

## Operation 9, Train a Portion of the Model

Each instrument is independently fit to a smaller portion of the model. All training-data for one single instrument is loaded, prepared, and fitted to a NnMF model with the same number of patterns in its dictionary as one instrument-class portion of the full dictionary. This operation needs to be repeated for each instrument. In Fig. 8 this operation is represented by one of the several parallel paths on the left.

For the implementation, each portion of the dictionary is stored separately and combined into the full model only when it is necessary to use it. This ensures that it is simple to update or replace any portions in an easy manner between experiments.

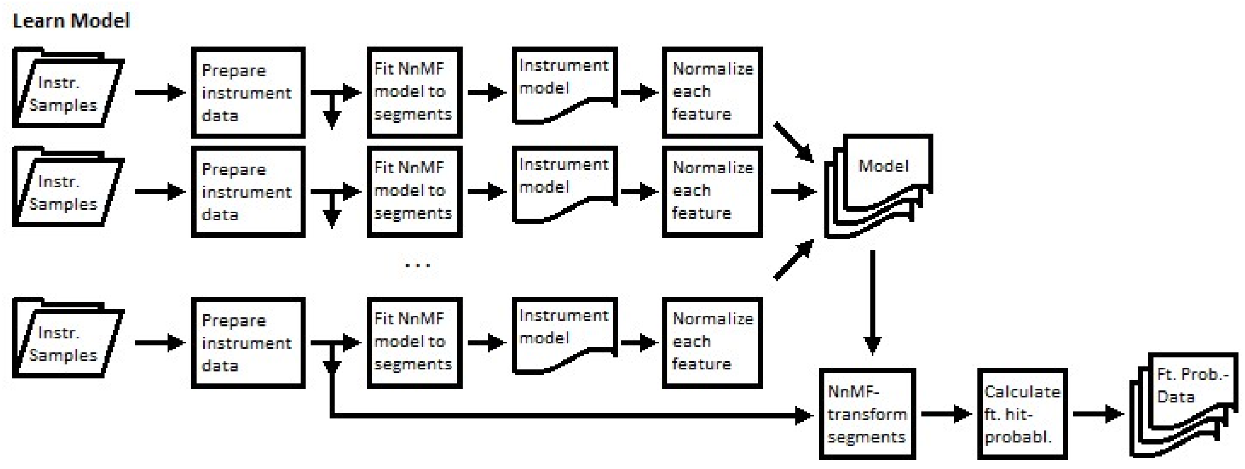

*Figure 8: Algorithm for learning the model. Individual smaller portions of the dictionary are trained for each instrument, then added together to form the overall dictionary. The model is then statistically analyzed to find the instrument utilization ration for the dictionary-patterns.*

## Method

In this section, the practical aspect of the work will be presented. This work will be in the form of an empirical experiment, based on an attempt to improve the standard NnMF source separation technique mentioned in literature. The goal is to evaluate the potential improvement in source separation by pruning uncertain patterns in the NnMF dictionary, and the experiment will test and evaluate this technique by comparing a hypothesis with some end results.

### Presentation of Experiment

When pruning the model, it is done as a means to counter inaccuracies as a result of over-generalization. A well generalized model is necessary for avoiding overfitting and the ability to cover a wide range of input, but too much generalization can lead to overlapping between classes and poor classification. Since portions of the model is trained independently, the generalization of one portion may be blind to the existence of other portions, leading to potential ambiguous patterns. Several factors can affect generalization, including number of features per pattern, and the number of patterns per instrument.

The hypothesis is that overlapping patterns and classification challenges will occur when learning the model partially, and that pruning the model will improve the performance.

In order to test the hypothesis, a model will be trained with the four instruments, 44100 samples per second, 2048-sample FFT frames, and 10 frames per pattern giving a 464 millisecond pattern length. Training, reclassification, and verification will be done for dictionary-sizes of 20, 40, 60, and 90 patterns per instrument. Verification will be done in combinations of two instruments, repeating the verification tests 10 times per combination. One verification test uses streams lasting at least 15 seconds, with on average 1.75 sound samples per second per instrument-stream. The full verification will be repeated for pruning-thresholds of 10, 20, 30, 40, 50, 60, 70, 80, and 90 percent pattern classification accuracy.

All results will be compiled as described under the section on "Operation 5". Overall results will then be compared for variations in test-parameters. Some of the notable instrument-combinations will be presented as well, before the average instrument utilization ratios are shown. The last set of data presented is a set of learned patterns from the dictionary along with corresponding classification accuracy data.

## Representation of Result Data

The experiment will produce a lot of data, including performance data on all twelve source separation scenarios for all nine pruning thresholds in combination with all the dictionary-sizes tested.

For the bulk of the data, graphs with means and standard deviation for the five measures will be presented. These graphs may be the average overall results, or one of the particular instrument combinations. The domain will be the different pruning thresholds, and different graphs may be given for the same experiment on different dictionary sizes. Every graph of means will be accompanied with the corresponding graph of standard deviation. The values represented by of these graphs will be the likeness compared to the original instrument stream. Likeness is calculated as the equation 1 – Error, and 100% is thus exactly similar to the target source stream and 0% is similar to either silence or that the matching spectrum is exactly canceled by the amount of error. Negative percentages correspond to a spectrum which has even higher error, where -100% would correspond to an error equal to two times the original magnitude of the spectrum and so on.

Instrument utilization data will be presented as average utilization of one class of dictionary-patterns, by the input audio data of various instrument classes. Data is presented in percent, where a point at 100% represent that all patterns of the chart's instrument class is used by only audio data from the one indicated instrument class. On the contrary, 0% means that the patterns of the chart's instrument class will not get used with input audio from the indicated instrument-class.

Patterns from the dictionary will be presented as monochrome images of their power-spectrum. The corresponding instrument utilization ratio will be shown as a graph of percentages. The domain of this graph will be the different dictionary-sizes.

## Results

In this section, results from the experiment will be presented. Overall Averages will present the overall results, Particular Cases will present the results from selected instrument combinations, Instrument Utilization Ratios will present the average utilization ratios for patterns from different instruments and learned patterns will present visualizations of learned patterns.

### Source Separation, Overall Averages

The following diagrams are derived from the overall average results. They are selected to highlight notable points derived from the experiment.

Results are expressed as percent likeness to the unmixed target source. The closer to 100% on the vertical axis, the better the performance score.

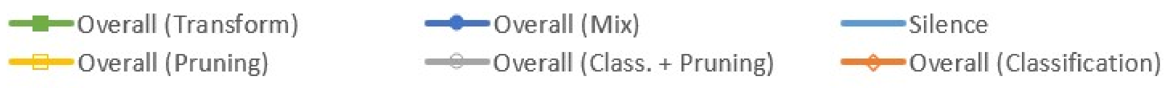

*Figure 9: Graph legend for the result mean likeness and standard deviation error graphs.*

The legend in Fig 9 will be used throughout the section, and it represents the various stages in the signal path compared to the unmixed instrument-stream magnitude spectrum.

The “mix” results will be the likeness between the initial unprocessed input mix and the unmixed target stream to be separated. “Transform” is the same as mix, but after it has been run through a NnMF transform and inverse transform. Comparing the “mix” and “transform” measures will give the effects the NnMF transform on the result, before source separation takes place.

The next three series are all for result after various different attempts at source separation. “Classification” is the results after source separation with no pruning, and it is intended as a baseline reference when evaluating results from source separation with both classification and pruning. The third series is if source separation is done using the entire dictionary, no classification, but with pruning. This last measure is included to test for how strong the pruning mechanism is alone, and if it potentially could be used on its own.

The line at 0% is the equivalent of silence, and it is provided for reference. At this line, the similarities are equally large as the dissimilarities when compared to the unmixed stream.

Each datapoint in Fig. 10 and 11 is based on 60 individual tests. Data which depends on pruning appears as a line, while data not dependent on pruning have no line. When the data is not dependent on pruning, the variation in performance score for those measures is due to the random factors of the verification-test. These series are therefore present mainly for reference, while the series dependent on pruning carry the main results.

All diagrams for average performance are immediately followed by accompanying diagrams for standard deviation. These represent the certainty of the average performance scores, and use the same legend as described.

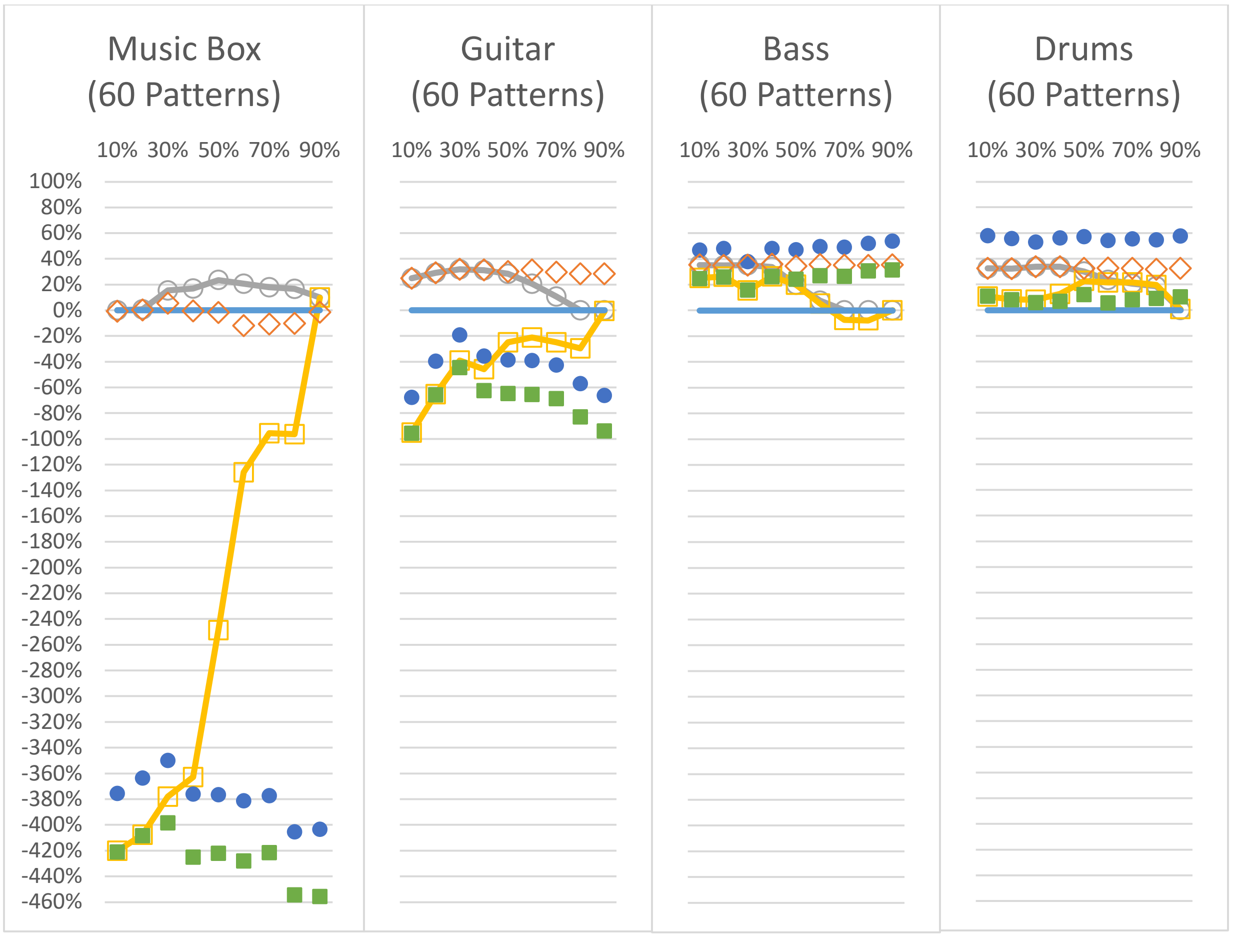

*Figure 10a: Average likeness of magnitude spectrum between signals compared to unmixed instrument streams. Horizontal axis is Pruning thresholds for instrument utilization ratio. Notice the general difference in loudness of the mixes compared to the unmixed streams.*

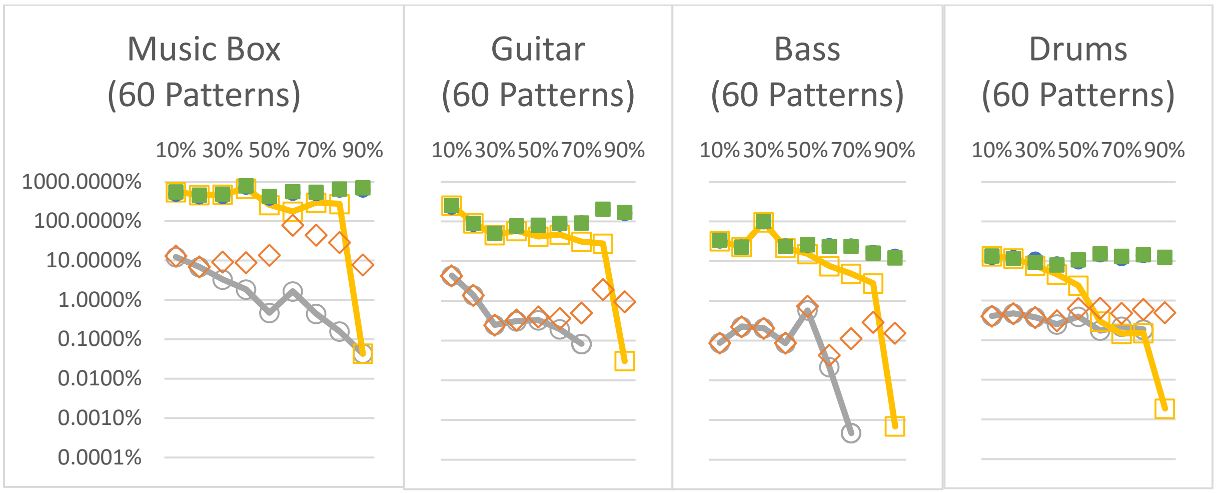

*Figure 10b: Standard Deviation for data respectively presented in Fig. 10a.*

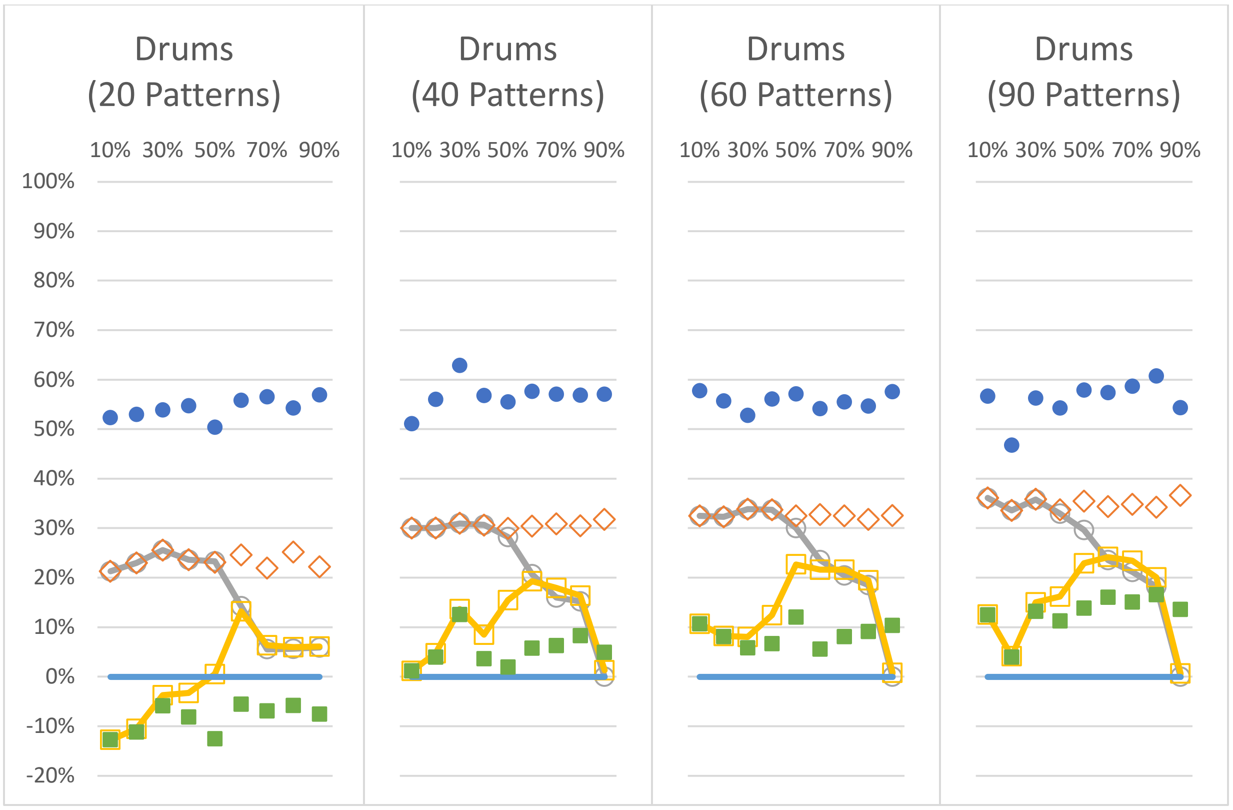

Figure 11a: *Average likeness of magnitude spectrum of results compared to the drums stream over a range of different dictionary sizes. Horizontal axis is Pruning thresholds for instrument utilization ratio.*

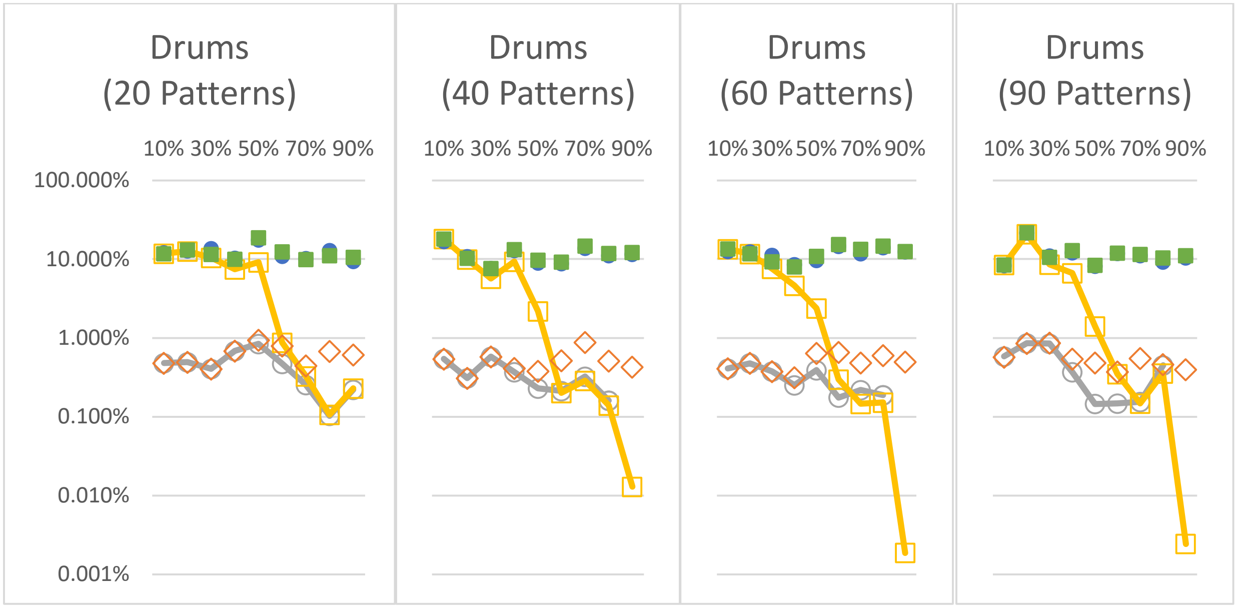

*Figure 11b: Standard Deviation for data respectively presented in Fig. 11a.*

## Source Separation, Particular Cases

The following section contains the average source-separation results for three particular instrument combinations.

Each instrument combination has a pair of diagrams, one for separating each instrument in the instrument combination. A diagram-pair is based on the average of 10 tests.

Like for the overall average results previously presented, the same legend and representation is used. This can be seen in Fig. 9. All the results in this section are using 60 patterns per instrument for the dictionary size. Also like the overall averages, accompanying plots of the standard deviation is also present for reference.

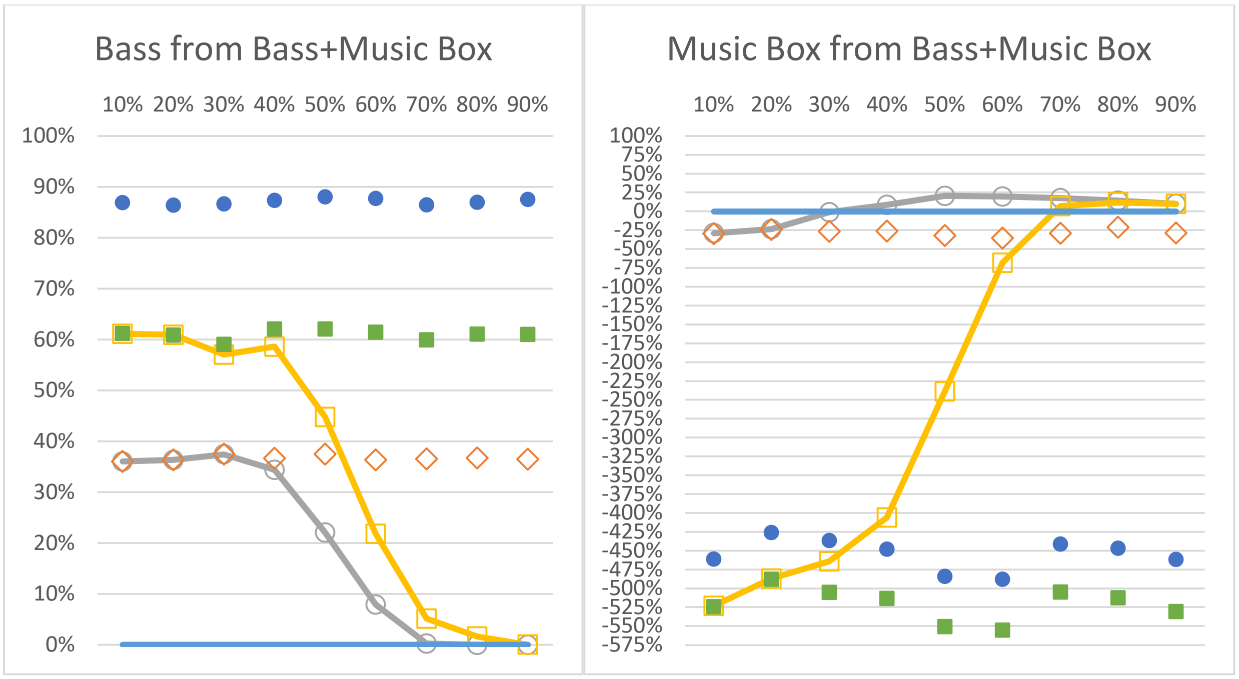

*Figure 12a: Average likeness for separation between Bass and Music Box, 60 Patterns per instrument. Horizontal axis is Instrument Utilization Factor pruning threshold.*

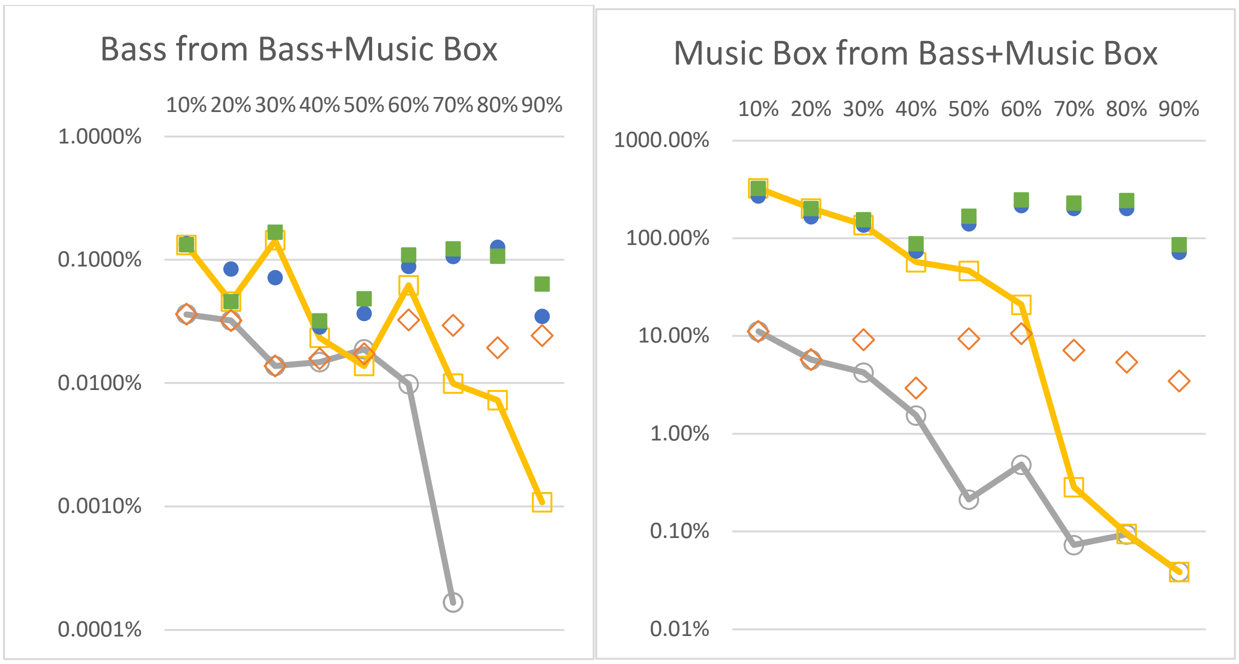

*Figure 12b: Standard deviation for separation between Bass and Music Box, 60 Patterns per instrument. Horizontal axis is Instrument Utilization Factor pruning threshold.*

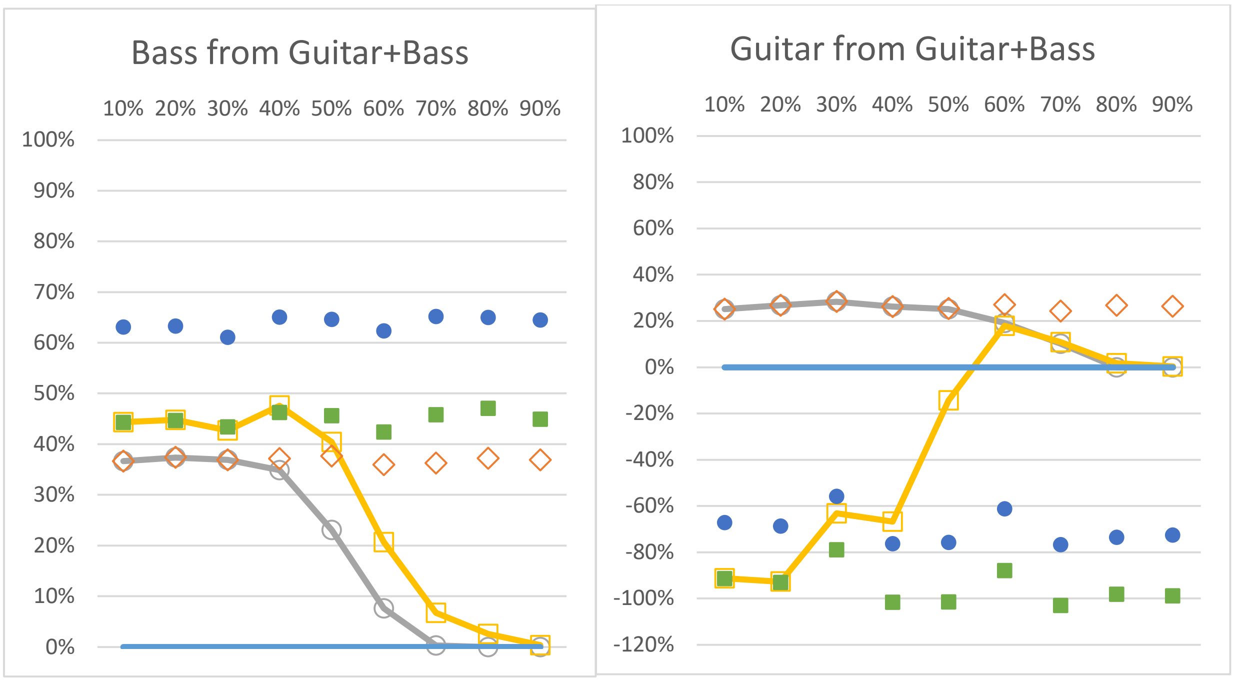

*Figure 13a: Average likeness for separation between Bass and Guitar, 60 Patterns per instrument. Horizontal axis is Instrument Utilization Factor pruning threshold.*

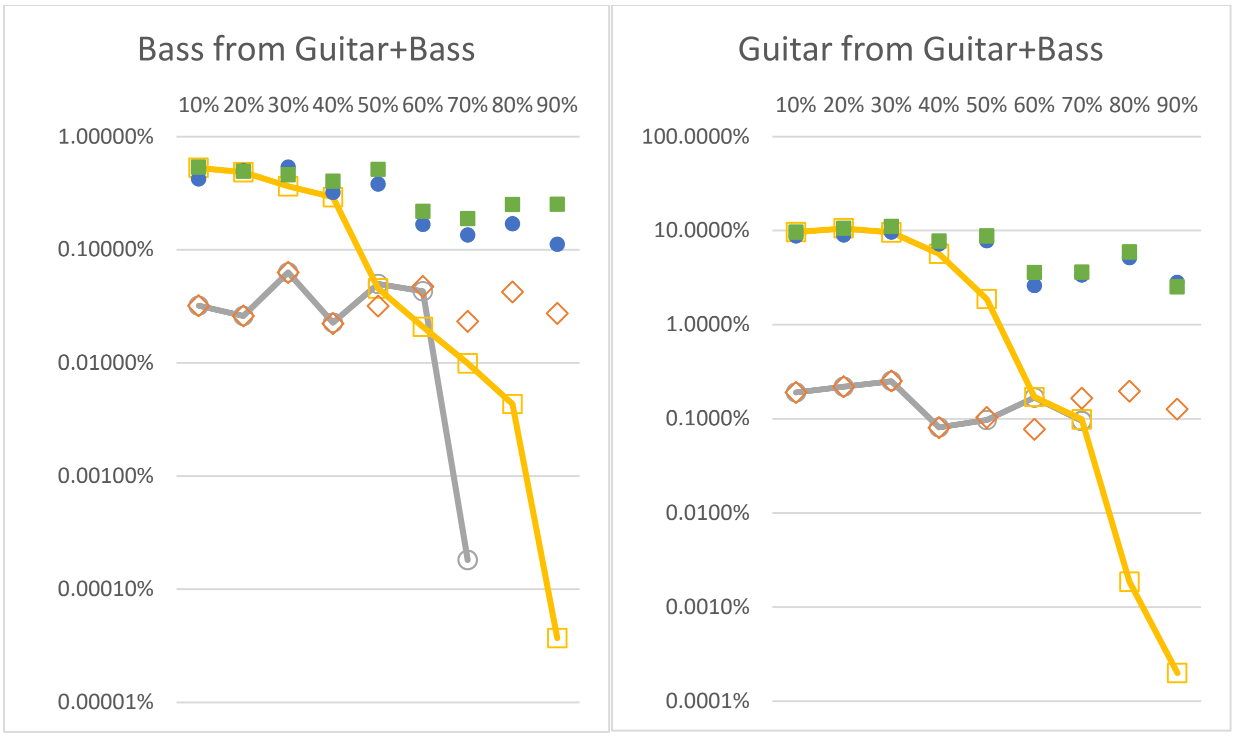

*Figure 13b: Standard deviation for separation between Bass and Guitar, 60 Patterns per instrument. Horizontal axis is Instrument Utilization Factor pruning threshold.*

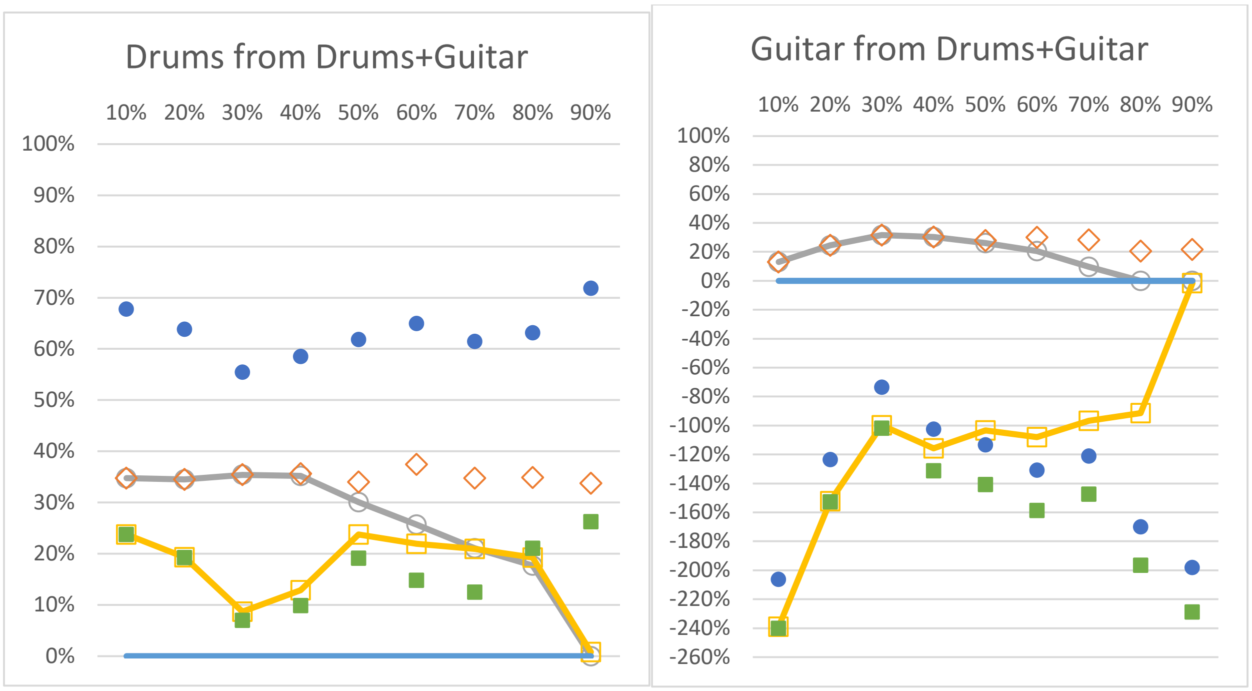

*Figure 14a: Average likeness for separation between Drums and Guitar, 60 Patterns per instrument. Horizontal axis is Instrument Utilization Factor pruning threshold.*

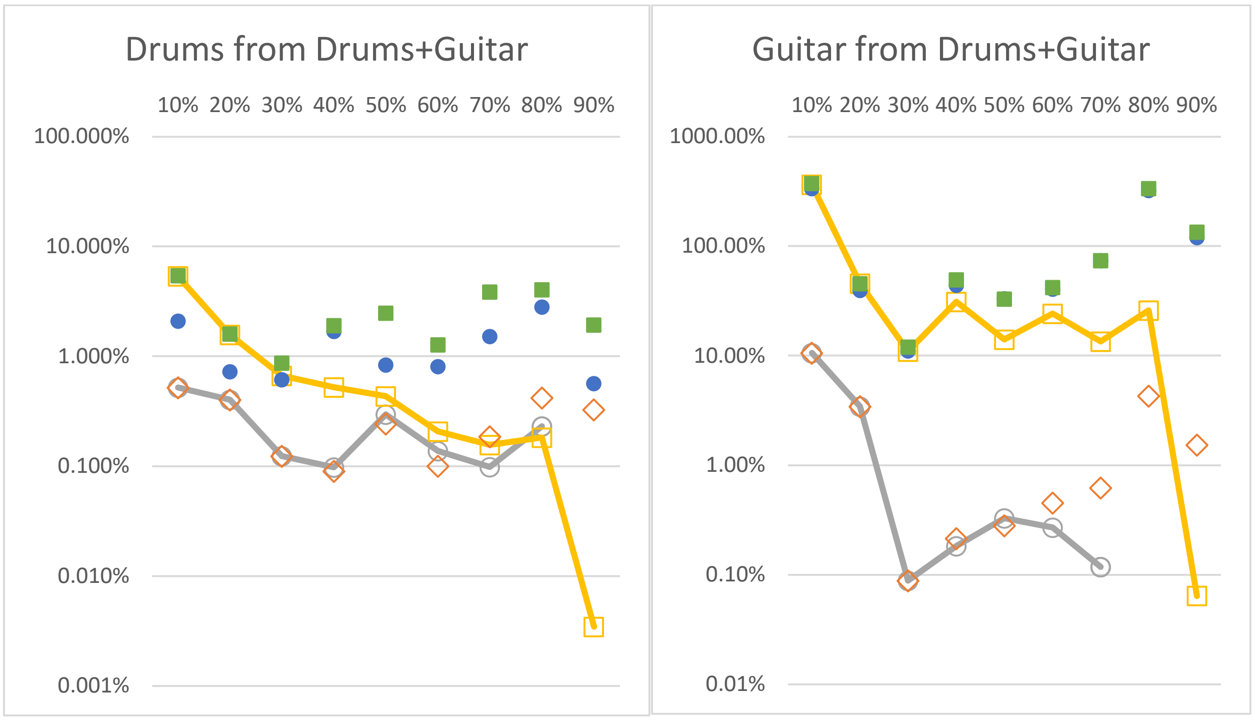

*Figure 14b: Standard deviation for separation between Drums and Guitar, 60 Patterns per instrument. Horizontal axis is Instrument Utilization Factor pruning threshold.*

## Instrument Utilization Ratios

Data in this section presents the average of instrument utilization ratios for patterns of all the instrument classes, over the different dictionary sizes.

Each diagram contains the mean instrument utilization ratios for the dictionary-patterns of one particular instrument class, represented in percent use by input audio data of all the instrument classes. The instrument utilization ratio represents the probability for a pattern in the dictionary to be used by audio from a given instrument source. The classification accuracy of a pattern is the instrument utilization ratio of the same instrument class as the pattern belongs to. A classification accuracy of 100% indicates that a pattern is only used by input data from the matching instrument.

Diagrams with standard deviations for the average instrument utilization factors is included for reference.

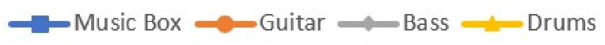

*Figure 15a: Legend for Instrument Utilization Ratio graphs. Audio training-dataset classes.*

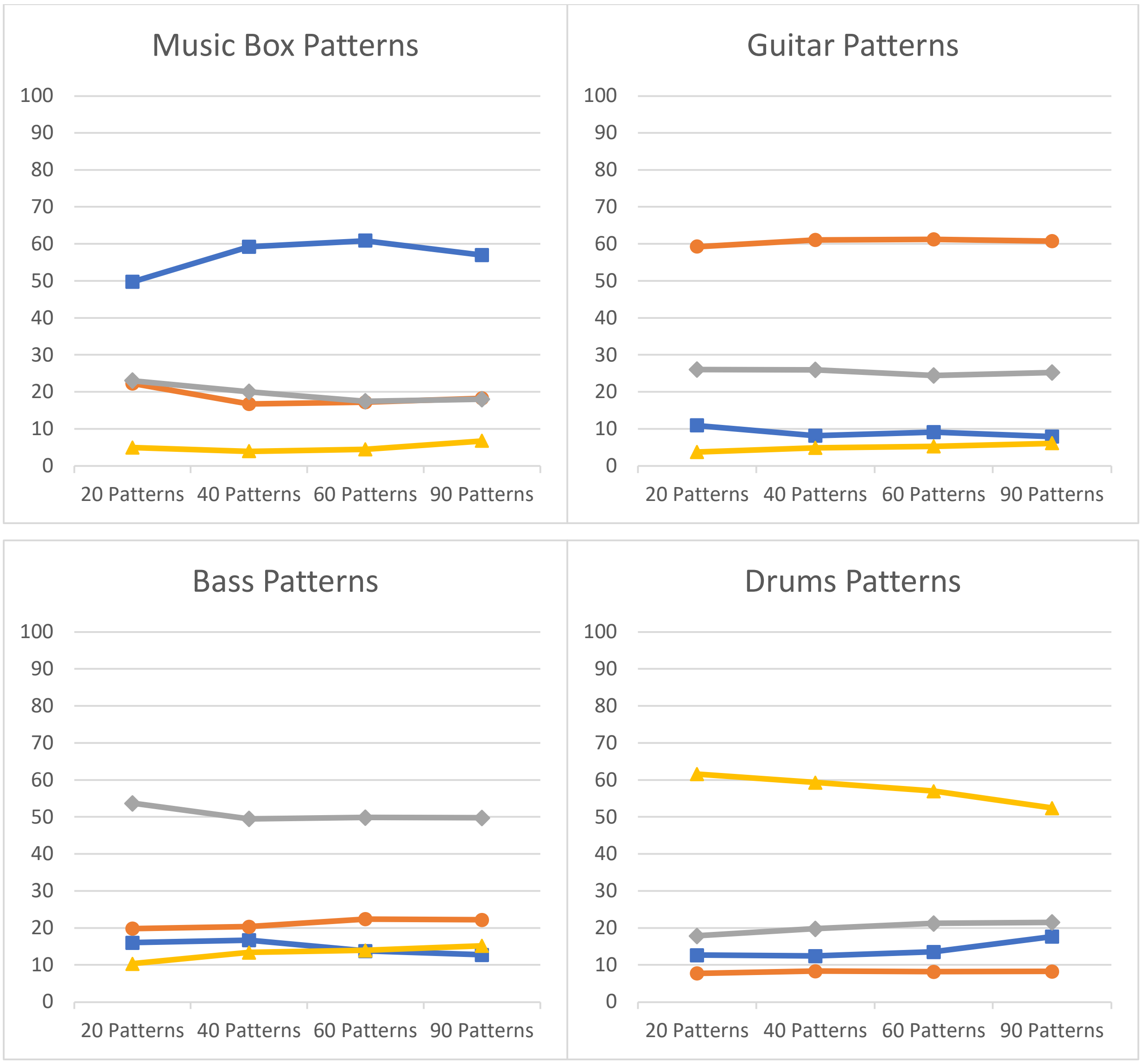

*Figure 15b: Mean instrument utilization ratio for learned patterns of the different instrument classes. Vertical axis is percentage usage. Horizontal categories give dictionary size per instrument.*

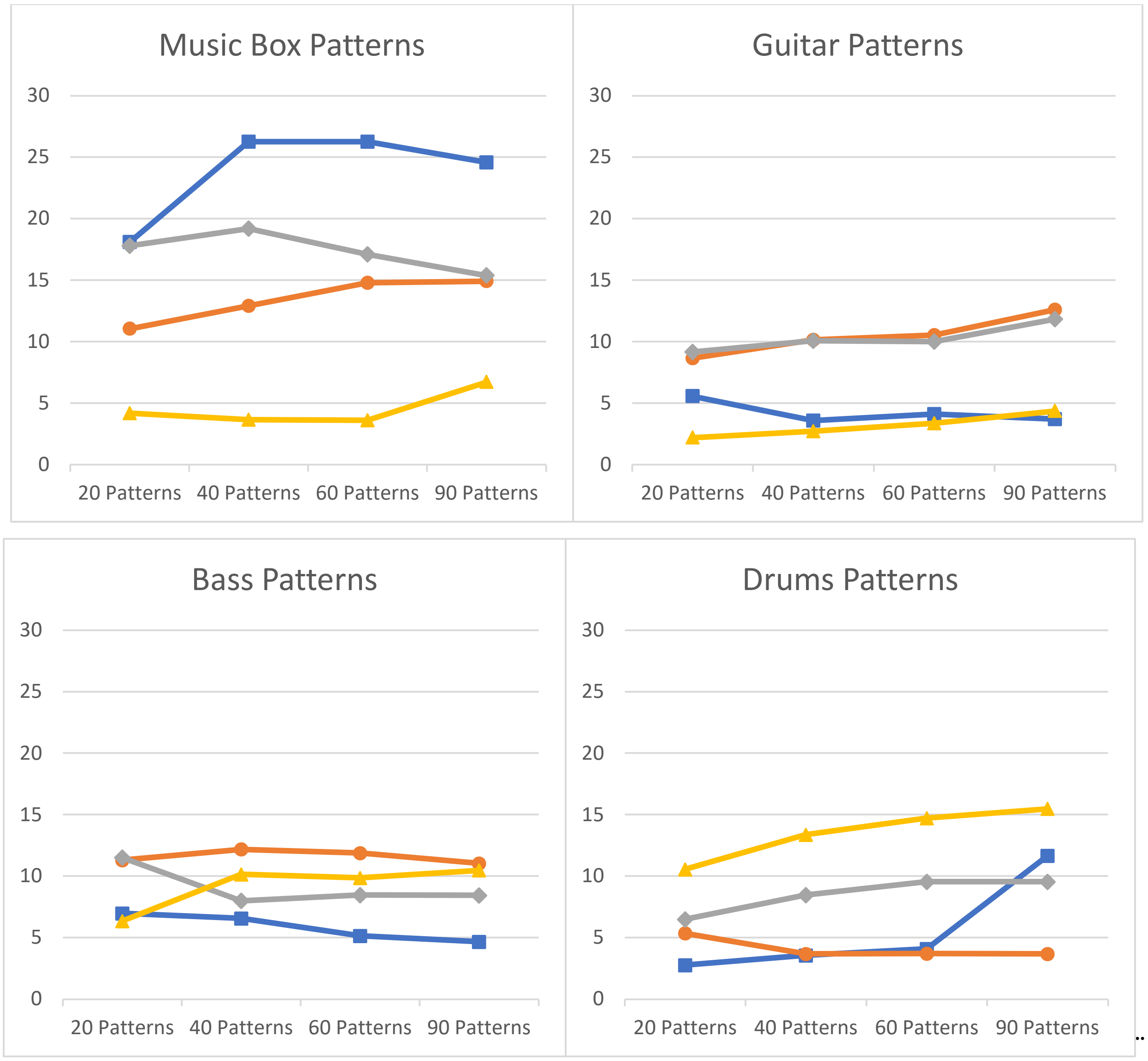

*Figure 15c: Standard Deviation of mean instrument utilization ratio for learned patterns of the different instrument classes. Vertical axis is percentage usage. Horizontal categories give dictionary size per instrument.*

From Fig. 15b, it is clear that the patterns hit the right instrument-audio on average around 60% of the time the given pattern is used. Fig. 15c also indicates that there is some variance, with most patterns hitting the right instrument on between 80% and 40% of the audio. Music box patterns have significantly more variance in average classification accuracy compared to the other instruments.

## Learned Patterns

This section contains the power-spectrum of the learned model for 60 patterns per instrument. The patterns are sorted from high classification accuracy to low, from left to right. The line in the bottom bar of each graph represents the accuracy for the corresponding pattern above. The top of the bottom bar corresponds to 100% classification accuracy, the bottom of the bar corresponds to 0% classification accuracy. The vertical axis for one particular pattern is frequency, from 0Hz on the bottom to 20KHz on the top, and the horizontal axis is time in discrete Fourier-transform frames. The brightness corresponds to the power of a given frequency at a given time, where black being either none or the lowest power of all the patterns presented and white is the highest power of all the patterns presented.

For the Music Box and Drums patterns, in Fig 16 and 19, it is worth noting that the classification accuracy divides the different patterns into several groups. Another thing worth noting is that for Guitar and Music box in Fig 17 and 16 there is a trend that classification accuracy gradually decreases along with the dominant frequency component in the pattern. The Bass, as seen in Fig. 17, has a lower overall accuracy than the other instruments, which is also reflected in Fig. 15b.

The patterns for Drums, seen in Fig 19, does the greatest job at generalizing, due to most patterns only affecting a small portion of the spectrum.

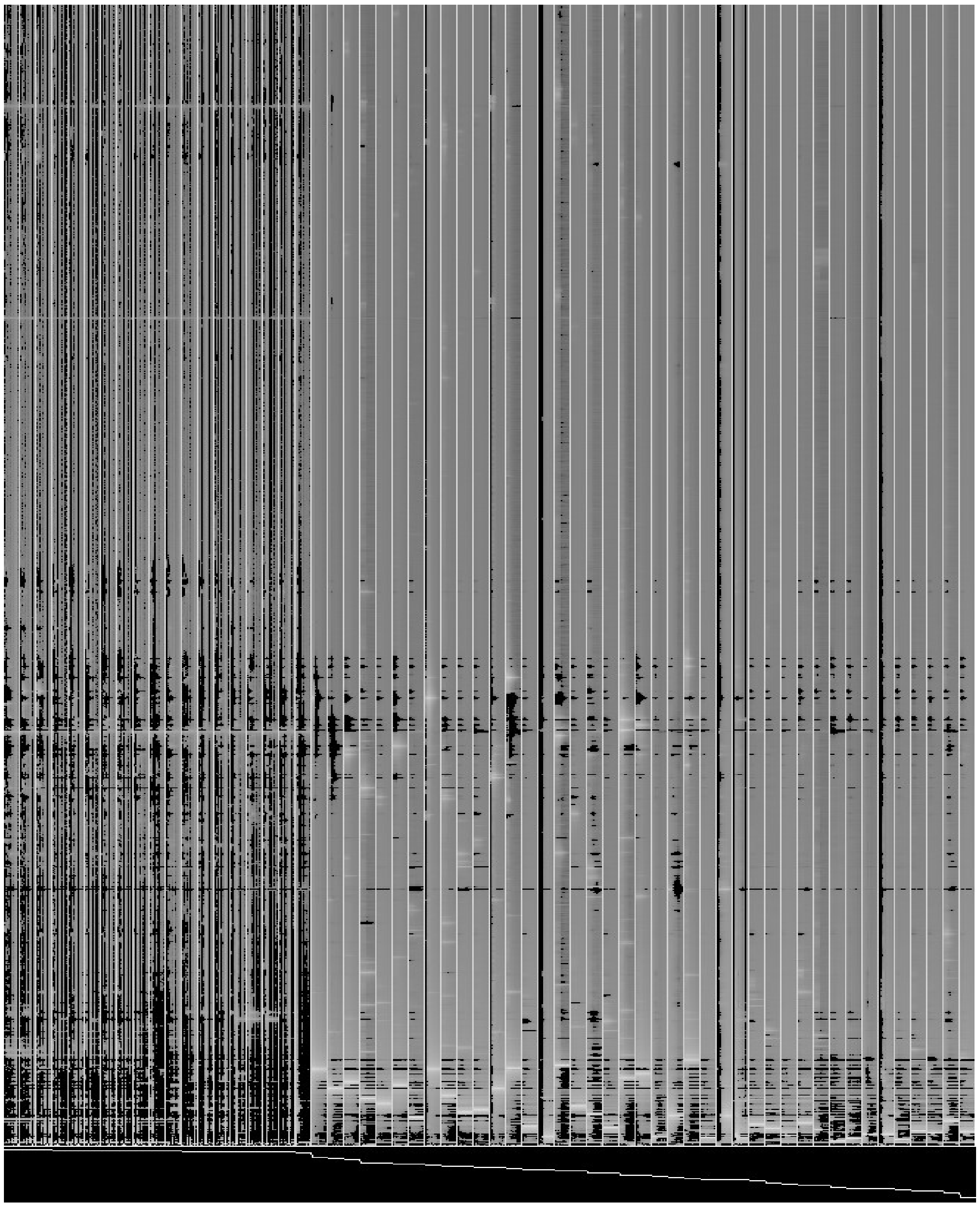

*Figure 16: Learned patterns for Music Box, 60 patterns per instrument. Decreasingly fit Instrument Utilization Factor towards the right, as represented by the horizontally plotted line.*

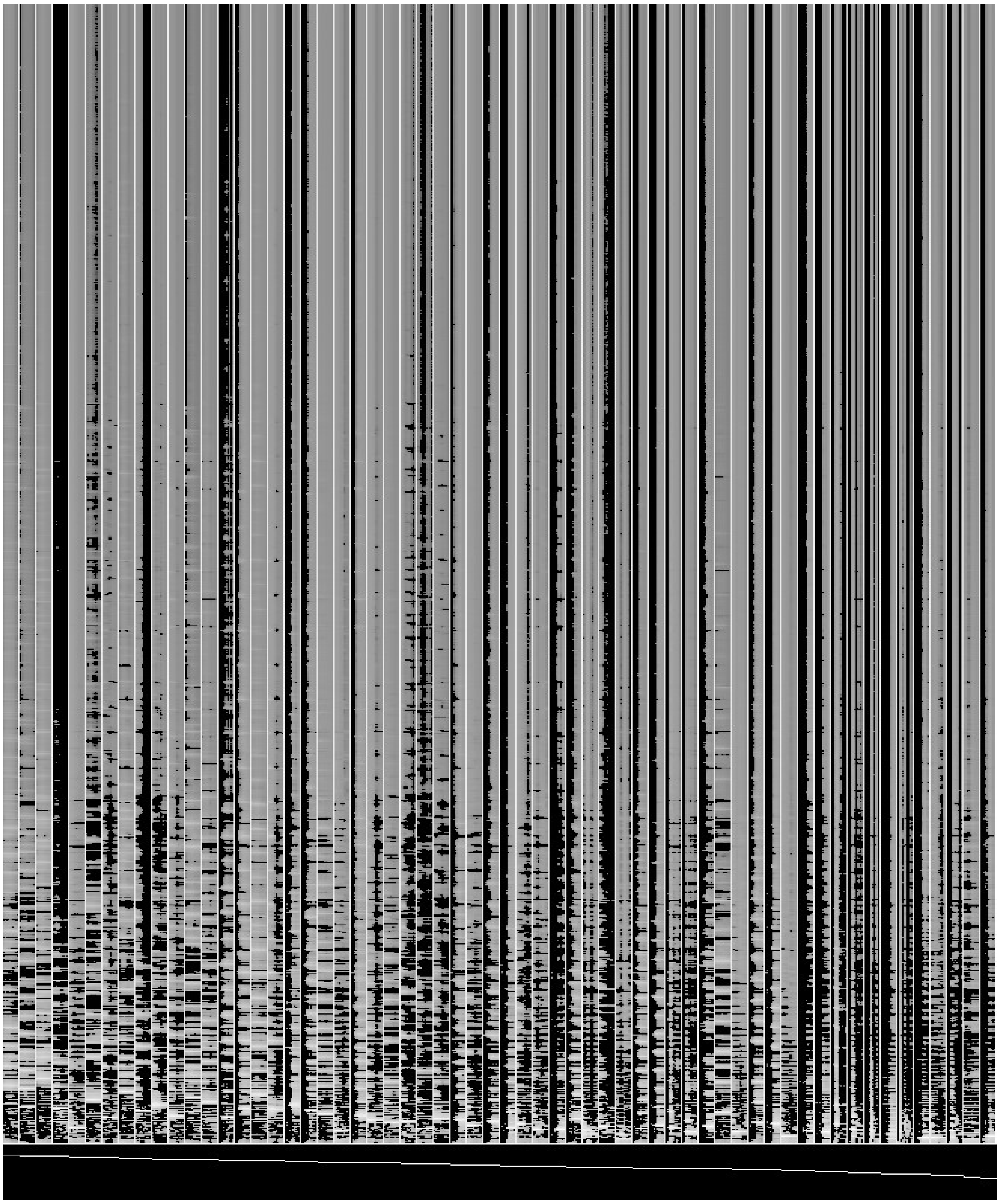

*Figure 17: Learned patterns for Guitar, 60 patterns per instrument. Decreasingly fit Instrument Utilization Factor towards the right, as represented by the horizontally plotted line.*

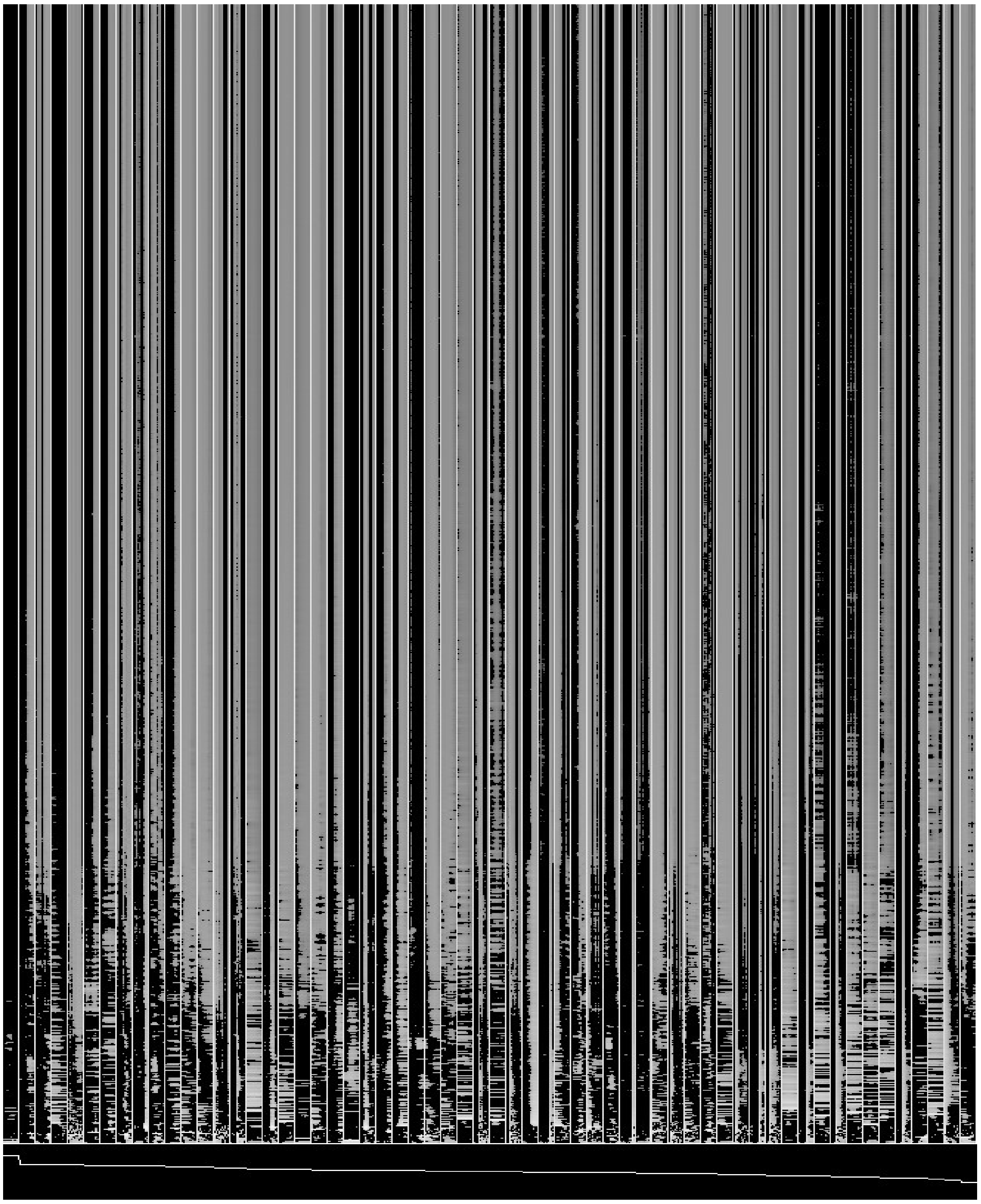

*Figure 18: Learned patterns for Bass, 60 patterns per instrument. Decreasingly fit Instrument Utilization Factor towards the right, as represented by the horizontally plotted line.*

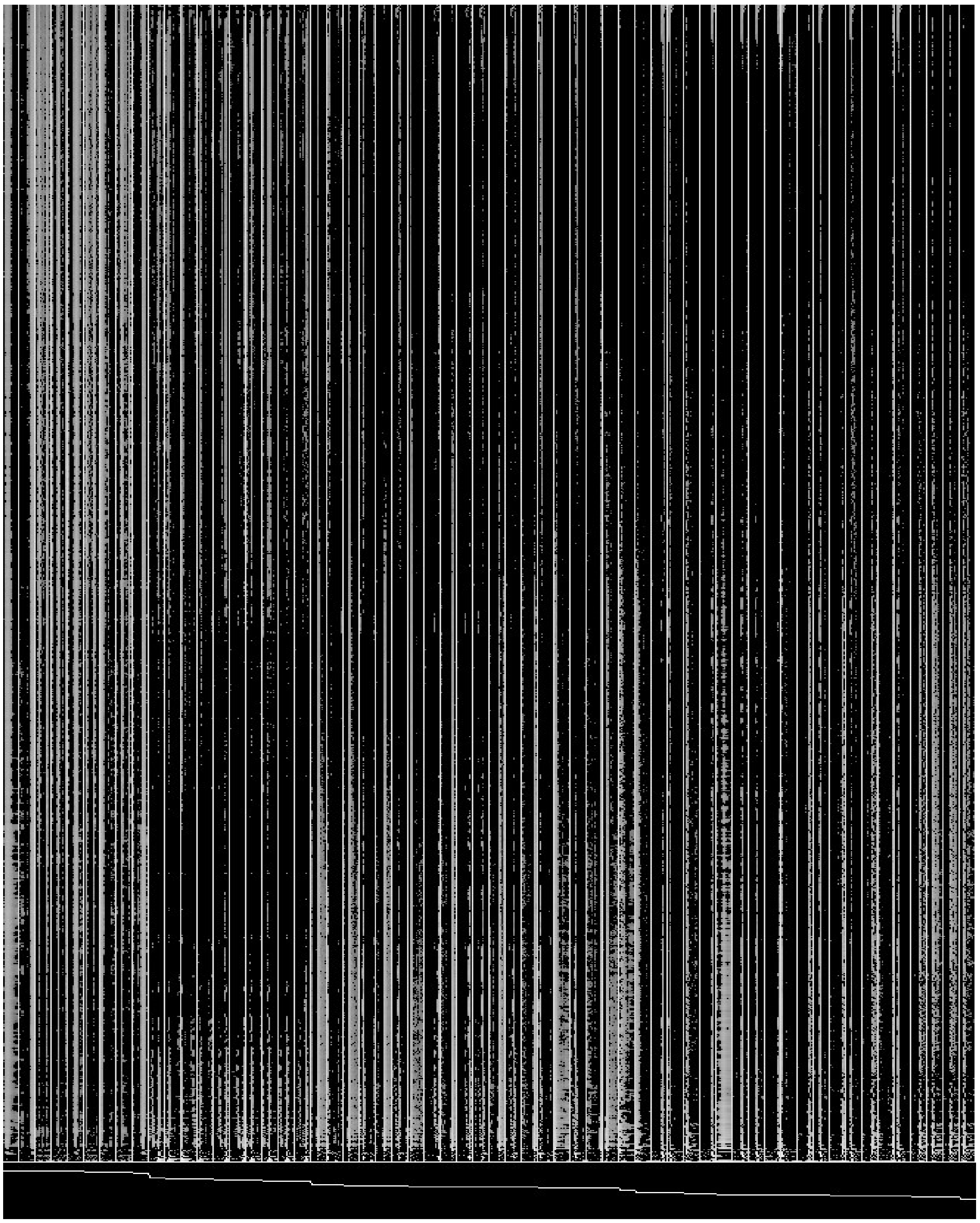

*Figure 19: Learned patterns for Drums, 60 patterns per instrument. Decreasingly fit Instrument Utilization Factor towards the right, as represented by the horizontally plotted line.*

## Discussion

From the results, some general trends can be observed. In particular the difference in loudness between different instruments plays a big role for the expected results when separating one instrument from another using the presented technique. In cases where a quiet instrument is separated from a loud instrument, the model generates results that are both more favorable than silence, and more favorable than the original transformed data. On the other hand, if the instrument separated is louder than the other instrument in the mix, the results will most often be less favorable than not doing any attempt at source separation at all.

### Effects of Non-negative Matrix Factorization

It is clear from the results in Fig.10 to Fig.14 as to why the performance in some cases can be worse than the mixed stream without any separation-attempt. The NnMF transform itself will introduce some artifacts and distortion which deteriorates the data, as seen by comparing the mix and transform lines in the figures (specified in Fig. 9). The distortion is not surprising considering how matrix factorization works. As the transform attempts to reconstruct the original spectrum using a limited amount of pre-learned generalized data, there will always be some information which is not possible to recreate from the model, as well as the model might have to accept some compromises and use patterns including incorrect artifacts.

The lossy nature of the NnMF transform can be made clear if considering the transform as a compression-algorithm. Each input pattern is transformed by the model into a column in the resulting W matrix, and if one such column contains lesser features than the number of features in the original pattern then the pattern has been compressed. The ratio of compression is dependent on the number of original features in relation to the number of features in the compressed representation. In this regard, the NnMF transform is locked to a constant and in the case of the particular experiment quite high compression-ratio. As the worst-case compression-ratio in lossless compression schemes equals no compression, the NmMF transform is forced to be lossy by its constant locked compression-ratio.

Given a large enough dictionary, where the number of patterns equal the number of features in one pattern alone, the NnMF transform could be lossless. This way each pattern-feature could have its own dedicated pattern in the dictionary. However, this would eliminate the purpose of the transform. In effect, the dictionary would become an identity matrix, and any resulting transform would be a W matrix with exact copies of the original untransformed input patterns. The purpose of the transform is to be able to classify and separate identified patterns of a given class, and this require the dictionary to be limited to only patterns distinct enough to be identified as one particular class. Restricting the amount of generalization in the dictionary is a requirement necessary for making classification possible. As such, the NnMF transform becomes a means where the data has to be deteriorated some, in order to potentially get closer to the goal of a source-separated end result later.

As seen in figure 10a, deterioration is potentially worse when the mix is much louder than the target instrument source. Should extra artifacts from the loud model be added in the transform, these added artifacts will be large compared to the quieter instrument to be separated. The ratio between the extra artifacts and the quiet source is reflected in the amount of deterioration.

Another variation is seen by comparing the deterioration in the overall average when separating drums and bass in Fig. 10a, as well as the particular case of separating drums and guitar in Fig. 14a. It is clear that the drums deteriorate much worse than the other instruments despite bass and drums being about equally loud, or the guitar being quieter than drums. From Fig. 10b and 14b, it is clear that the standard deviation of transformed and untransformed mix is almost similar, suggesting that the difference in deterioration between the instruments is systematic.

It seems like drums are prone to much worse deterioration than the other instruments, despite it being one of the loudest. An explanation for this is suggested in Fig. 11a, where the deterioration for drums significantly decreases as the dictionary size increases. This indicates that the learned model-portion for drums is a lot less generalized than the model-portions for the other instruments. Such a scenario would lead to the model having trouble recreating patterns from a drum instrument source, leasing to increased artifacts and distortion should the drums be present in the mix to be analyzed.

The NnMF transform will in all cases do what it can to recreate input patterns using its model, even if the input patterns do not resemble anything the transform is capable of recognizing. The model will handle such a scenario by using a combination of patterns of closest fit, despite the patterns being potentially very different. Such situations may lead to the input data being reconstructed with a mix of patterns from any of the instrument classes, which will lead to a high number of artifacts and reduced performance.

## Effects of Learned Patterns

To further investigate how certain instruments sources gets more distorted than other instruments after a NnMF transform has been performed, it is important to consider the model itself. If the model is not able to generalize patterns of a given instrument, it is not able to classify and separate it correctly. In Fig. 16 to 19, the entire model for 60 patterns per instruments is presented. These reflect several highlights of behavior seen in some of the tests.

Fig. 19 can be used to find the reason why the drums portion of the model has low performance, with its high deterioration of the signal. The patterns for the drums are divided into very distinct groups of patterns, where each group seems to store the same particular sound but at different time-delays. This can directly be associated with the vertical and short sounds associated with the drum-set, where one particular sound may start anywhere inside an input pattern. The benefit of storing several copies of the same sound at different timing, is being well suited for sharp and short sounds which can occur anywhere, while the disadvantage is that a lot of patterns are used for the same sound. In effect, this is limiting the size of the drums portion of the dictionary. As seen in Fig. 11a, increasing the dictionary size greatly increases the performance of the drums portion of the model.

The music box portion of the model can be seen in Fig. 16. This portion is also divided into groups of patterns with somewhat similar classification accuracy. About a third of the portion, the portion with the highest accuracy of the entire model as a whole, turns out to be patterns learned from some static background-noise in the music box audio clips. Although an unintended occurrence, it does not come as a surprise as the noise seems to be particularly distinct and consistent in the recordings of the sample-bank. The music box is also a quiet instrument, and it does nothing to dominate over the noise where it appears in the frequency-spectrum. This might explain some of the good results when separating music box from other instruments, as seen in for example Fig. 12a. On the other hand, it has to be considered if it is primarily the music box sound or the static background noise which is being separated.

Another observation from in Fig. 16, is that after the static noise patterns, the following patterns distinctively resembles timbres of the music box, with some metallic overtones. It is expected that these patterns are good classifiers since they are quite unique to the music box. The classification accuracy then decreases gradually, along with the general dominating tonal frequency in the patterns. The least accurate patterns also lack the metallic overtones, and it is therefore not surprising that these patterns may be utilized to recreate patterns from the other tonal instrument classes as seen in Fig. 15b.

Like the music box, the guitar portion seen in Fig. 17 shares several trends. In particular the patterns with highest accuracy have a distinct similarity to the tonal timbre of the instrument, and in the case of the guitar bearing strong traits of the harmonic series. The accuracy also decreases along with dominating tonal frequency. A possible explanation for this may be that some of the patterns of the guitar, as shown in Fig. 15b, to some extent matches with the sound of the bass.

From Fig. 15 it is also possible to see that the different dictionary-sizes tested did not necessarily alter the average instrument utilization ratio by a lot, if something it may have slightly increased the variation in instrument utilization ratio some for certain instrument classes. This may indicate that a bigger model does not always result in a better ratio between patterns with high and low classification accuracy. With 20 extra patterns per instrument in the dictionary, if these patterns are filled with pattens of the same average accuracy than the rest of the dictionary, no change in average accuracy would happen.

The overall variation in what types of patterns seen in the dictionary is indeed varied, and as shown directly linked to the performance of the NnMF transform itself. Since the patterns are also the blueprints used to reconstruct the input data, the patterns are directly related to the performance of the source-separation as well. In the model presented in Fig 16 to 19 there are, as discussed, some priorities that seem odd from an application standpoint, like using much of available space to store patterns of unimportant noise, but many patterns genuinely represent distinct spectrum from the respective instruments. This last majority of patterns contribute positively to the performance when using the model for source-separation work.

As the learning-process is unsupervised within one particular instrument, the model cannot know which features in the training-data is important for the listener. It will still try to learn patterns by clustering and generalization, and trends in features are prioritized based on qualities such as extent and similar occurrences in the training data. This approach focuses on performance alone, and this in turn leads to occasions where things like static noise is prioritized. If the problem is to be avoided, more varied training-data is necessary, like including recording of training-audio in varied environments. This makes the noise non-systematic, and thus less important for the learning algorithm.

One of the concepts of the proposed algorithm, was to have the model learn the portion for each instrument independent from another. A potential issue with this, is that it is impossible for the learning algorithm to take the other instruments into account when fitting the model. This has the potential for leading to similar conflicting patterns in different instrument classes. Looking at the patterns in the model, this seems to be the case for many of the patterns with poorest classification-accuracy. In particular patterns with the majority of their energy in the lower frequency range seems to do worse in this regard, something which is not surprising due to many instruments covering this low frequency-range. Higher frequency components in sound are often sparser and more distinct, and thus yield better classification when featured in patterns.

## Effects of Source Separation

From the above sections, there should be a clear understanding for why the NnMF transform introduces some artifacts deteriorating the input data, as well as how the patterns in the model affects the deterioration as well as providing the foundation for a good source-separation. If the input data after the transform and source separation is closer to the target unmixed instrument compared to after the transform alone, the source separation can be considered to have been beneficial for the end result. The minimum requirement for the source separation to be successful, is thus to at least counter the deterioration introduced by the NnMF transform. This will yield a separated audio data which is more like the target instrument than the original unprocessed mix.

As the results from the experiment is presented as likeness to the target isolated instrument, both the removal of non-similar features from other instruments and keeping similar features from the target instrument will help improve the performance score. For some instrument combinations, in particular where the other instruments are much louder than the target instrument, this may lead to a situation where the initial likeness before source separation is negative. Such scenarios can be seen in the second graph of the separation pairs of Fig. 12a to 14a. In these situations, the elimination of mismatching features will improve the likeness to the target a lot more compared to preserving the matching features, and thus silence will seem like a very favorable result despite it being contradictory to the intent of source-separation.

Positive likeness is the only requirement which will guarantee in all cases that there is at least some resemblance between the output of the separated audio data and the target instrument audio data. In this case the preserved similar features outweighs the dissimilar features. From Fig. 10, it can be seen that on average, source separation of all the instrument classes are capable of achieving this to some extent. Looking at figure 12a, it is however clear that despite a positive overall average, less favorable results may occur with certain instrument combinations.

Another observation from Fig. 10, is that the improvement by doing source separation is again directly related to the loudness of the content to be separated compared to the loudness of the mix. The benefits of the source separation greatly decreases the louder the target source is, and when the source to be separated becomes the dominates the loudness the effect of the source separation is never a good enough improvement to counter the data-degradation introduced by the NnMF transform. In certain cases, in particular when separating bass in Fig. 12 and 13, source separation even further decreases the likeness. This may be due to a potential overlap between bass and another instrument, but it is not helped by the fact that bass is the loudest instrument of the mix in both set of tests.

When separating a quiet instrument from a louder mix, it is possible to get some significant improvement, and this is where the algorithm has its best strengths. A prime example of this is when separating guitar in Fig. 13 and 14. Although the likeness is only around 20%, it is on par with the results from other instrument combinations, above silence, and greatly above the initial mix.

The loudness dependence of the performance for the source separation indicates that the algorithm under the tested conditions is a crude operation, which is better at eliminating unmatching features rather than to preserve detailed matching features.

## Evaluation of Pruning

Pruning is by nature elimination of targets. In this particular case, additional patterns which should otherwise be included during source separation are removed during the operation. The pattern classification accuracy is used together with a threshold for what patterns to prune or not.

In the case of source separation, pruning has the potential to be both a benefit and a disadvantage. On one hand it improves classification accuracy, on the other hand it removes patterns which, despite some uncertainty, may still contributes to the desired target. Pruning with a high threshold ensures that the source separated audio is very likely from the correct source, yet it may only be a small fraction of the complete target source.

The purpose of pruning is to remove patterns with low accuracy from a class. This should further increase the tendency of the model to favor elimination of unwanted features. As noted, this tendency is already very strong with the algorithm. This might be beneficial in some cases, but it has the potential to create an algorithm which is too aggressive.

From a purely technical standpoint, removal of uncertain patterns should be beneficial if two conditions are fulfilled. The first is that the pruned features are in use, the second that their inclusion degrades the mean likeness to the target instrument class.

Based on the average results section, pruning makes almost no difference together with classification. It only showed positive effect when separating the music box instrument class as seen in Fig. 12. Fig. 10 confirms that this improvement is consistent for separating the music box class from other instruments as well. Otherwise, for all other instruments it was either no significant difference or a decrease in likeness compared to pure classification.

There is a trend that the effects of pruning seem mostly independent of the other instrument in the mix, in other words it is mostly dependent on the particular instrument class to be separated. This in turn suggests a dependency on the performance gain or loss from pruning depends on the composition of the portion of the dictionary corresponding to the instrument class to be separated.

From Fig. 15c it can be seen that the variation in accuracy is relatively low in most instruments, but for music box portion of the model it is significantly higher than for the other classes. Moving on to the visual graphs in Fig. 16 to 19, it is clear that the music box class is the odd one out as it has reasonably many patterns with high classification accuracy, as well as a somewhat large amount of patterns with very low accuracy as well. With pruning it is therefore possible to keep many and a diverse variety of patterns, preventing too much of a loss in performance. In addition, the music box being the quietest instrument source also benefits more than the other instruments from a more aggressive algorithm.

The dictionary-portions for other instruments than music box have a distinct lack of patterns with very low accuracy. A low to moderate amount of pruning have little to none effect, and when pruning starts to have an effect, the disadvantages have already started to outweigh the benefits.

## Conclusion

The experiment has shown that pruning away patterns with low classification accuracy from a non-negative matrix factorization dictionary can have a modest effect on source-separation under certain conditions. For the pruning to have a significant effect, the model must on a per-class basis contain a sufficient number of patterns with both very low and very high classification accuracy.

Based on the instrument utilization ratio, there was indeed a moderate amount of overlap between several of the instruments. However, only the music-box instrument class showed any improvement at all when pruning was used. This shows that patterns with some overlap alone is not enough, and there must be patterns of very little accuracy present before a benefit from pruning can be seen.

The conditions which makes pruning beneficial is by no way guaranteed by the learning algorithm. An ideal model should not contain features with very low accuracy in the first place, so pruning with a low threshold of below 40% accuracy can be regarded more as a safety-measure to improve the performance of a poor model rather than improve a high-performing model. This limit will likely not be the same in all cases, and it will likely depend a lot on the number of distinct source classes. In a case where pruning is necessary to improve a model, there might be more potential for improvement in adjusting the learning-algorithm or training dataset to improve the model as a whole.

## Further work

This section will suggest some improvements to the techniques used in this project

### The Wavelet Transform

While the discrete Fourier-transform with fixed frame-size is an industry standard for signal-processing, it has some limitations when it comes to flexibility. As the frequency resolution directly depends on the length of a frame, the resolution in time is directly related to the desired frequency resolution. Increasing frequency resolution decreases temporal resolution and opposite, which means that the temporal resolution is compromised across the entire frequency-range if a high enough frequency-resolution is needed.

As the purpose of the Fourier-transform in the algorithm is intended to replicate the workings of the cochlea of the ear, it can be worth to compare to how the cochlea transforms sound. As the transform in the cochlea is done directly with resonating oscillations triggering independent sensors, changes in the frequency-time domain can be registered and acted upon as soon as the oscillations start at the given frequency. Because of this, the temporal resolution is dependent on each observed frequency individually rather than all observed frequencies being locked to being observed at the worst-case temporal resolution.

There is another transform that lies closer to the technique of the ear. This technique, wavelet Transform, works in much the same way as the Fourier transform, but corelates the signal to a wavelet instead of a continuous sinewave function. In rough terms, the wavelet transform operates on the principle that the frame size is adjusted in terms of the observed frequency, giving simultaneously high frequency resolution and low spatial resolution for lower frequencies, while low frequency resolution and high spatial resolution for higher frequencies. (Graps, 1995)

Using the wavelet transform in source separation would bring the properties of the input data closer to the properties of the data the brain is presented by the ear. This has the potential to better identify the same type of patterns the brain would work with, as the resolution in frequency and time would be more compatible with the human perception.

## Realtime Considerations

The first computer programs dealing with digitalized sound were not able to play the samples in real-time. Likewise, source-separation algorithms based on matrix factorization poses great challenges when challenged with real-time operation.

One of the most resource-expensive operations for the source-separation, is by far the NnMF transform. The fitting algorithm iterates through hundreds of cycles of big matrix multiplications in a process of regressive approximation for every single pattern. Unoptimized and worst-case matrix multiplications has a cubic big-O performance cost, resulting in a cubic increase in computation time as the size of the matrices increases linearly. Due to this, in combination with the large matrixes necessary for the NnMF model, it is not practical or possible to implement a good source separation in real-time using an algorithm based on NnMF. For real-time source separation, another algorithm based on faster techniques should preferably be used.

## Phase Recovery

A challenge with digital source-separation techniques is the reliance on using the magnitude spectrum only. Converting this back to audio data can be a challenge when the phase is not present.

The algorithm in the project merely reuses the phase from the original unprocessed input spectrum. This may work well if the output is reasonably loud in the mix, but in parts where another instrument is dominant before separation, the phase of the other instrument would dominate in such a case, leading to a greatly incorrect transient of the output source-separated audio. This is a well-known problem for source separation, and for using the algorithm in an application setting some of the more recent phase recovery algorithms could be used for improved results. In one particular promising example, the phase spectrum is slowly adjusted to maximize temporal continuity (Magron, Badeau, & David, 2018).

## Acknowledgements

Thanks to:

Prof. Øyvind Brandtsegg, Professor at Norges Teknisk-Naturvitenskapelige Universitet, NTNU, for being the guidance consultant for the project.

Semantic Music Technologies group at Fraunhofer IDMT for the availability of the audio sample-banks.

Magne Hagen Ring and Ola Aune for moral support.

## References

Bosi, M. G. (2003). *Introduction to Digital Audio Coding And Standards.* Springer Science+Business Media, LCC.

Boulanger, R. (Ed.). (2000). The Csound Book: Perspectives in Software Synthesis, Sound Design, Signal Processing, and Programming. Cambridge, MA: The MIT Press.

Cano, E., FitzGerald, D., Liutkus, A., Plumbley, M. D., & Stöter, F.-R. (2019). Musical Source Separation: An Introduction. *IEEE Signal Processing Magazine*, 31-40.

F. Pedregosa, G. V. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 2825-2830.

Graps, A. (1995). An Introduction to Wavelets. *IEEE Computational Science and Engineering, 2*(2).

IEEE History Center. (1998). *Fifty Years of Signal Processing: The IEEE Signal Processing Society and it's Technologies 1948-1998.* The IEEE Signal Processing Society.

Lee, D. D., & Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, 788–791.

Magron, P., Badeau, R., & David, B. (2018, June). Model-Based STFT Phase Recovery for Audio Source Separation. *IEEE/ACM Transactions on Audio, Speech and Language Processing*(6).

Makino, S. (2018). *Audio Source Separation.* Springer International Publishing AG.

Millard, A. (2006). *America on Record: A History of Recorded Sound, Second Edition.* Cambridge University Press.

O'Regan, G. (2012). *A Brief History of Computing, Second Edition.* London: Springer Science+Business Media.

Poeppel, D. (2003). The analysis of speech in different temporal integration windows: cerebral lateralization as 'asymmetric sampling in time'. *Speech Communication*, 245-255.

Proakis, J. G., & Manolakis, D. K. (2006). *Digital Signal Processing, Principles, Algorithms and Applications, 4th Edition.* Pearson.

Roads, C. (1996). *The Computer Music Tutorial.* The MIT Press.

Russell, S. J., & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach, Third Edition.* Upper Saddle River, NJ: Pearson Education, Inc.

Sanderson, G. (2018, 01 26). *3Blue1Brown*. Retrieved from https://www.3blue1brown.com/: https://www.youtube.com/watch?v=spUNpyF58BY

Smaragdis, P., & Brown, J. C. (2003, October). Non-Negative Matrix Factorization for Polyphonic Music Transcription. *IEEE Workshop on Applications of Signal Processing* *to Audio and Acoustics* (pp. 177-180). New Paltz, NY: Institute of Electrical and Electronics Engineers.

*What's New In Python 3.0*. (2020, 05 18). Retrieved from Python v3.0.1 documentation: https://docs.python.org/3.0/whatsnew/3.0.html