Friedrich-Alexander-Universität Erlangen-Nürnberg

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Master Thesis

Data-Driven Singing Activity Detection for Music Recordings

submitted by

Halil Erdogan

submitted

November 22, 2021

Supervisor / Advisor

Michael Krause
Prof. Dr. Meinard Müller

Reviewers

Prof. Dr. Meinard Müller

AUDIO LABS

International Audio Laboratories Erlangen
A joint institution of the
Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and
the Fraunhofer-Institut für Integrierte Schaltungen IIS.

Fraunhofer IIS

Erklärung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form in einem Verfahren zur Erlangung eines akademischen Grades vorgelegt.

Erlangen, November 23, 2021

Halil Erdogan

Acknowledgements

First and foremost, I would like to express my gratitude to Michael Krause and Meinard Müller. From the beginning of my study to the end of it, they supervised me and give me an opportunity to see different topics and work on whatever we would like to work on. Also thanks to their help, I never worried about trying some new things and failing. Because with each failure, we really go into details and learn from that thanks to their principles.

While my writing process was under influence of a global pandemic, my family supported me a lot. And also from my childhood to today, they always provide to opportunity to chase my dreams and become what I would like to be. So I am very grateful to them.

I also want to thank my friends who are really always with me and open to talking about everything. Especially my thank goes to Talha, Emre, Özgür, Oguzhan, Selim, Kubilay, Rubar, Brad, Chien-Hung, Alex, Kaan, Sinan, Yuşa, Canberk, Furkan and Onat.

Furthermore, I would like to thank some professors who supported and inspired me from the beginning of my bachelor as Müjdat Çetin, Umut Şimşekli and Sertan Şentürk.

Moreover, I want to thank people who really influence my mindset to be able to understand how much I can push my boundaries and keep motivated as Emil Cioran, creators of *Bu Mu Yani?* podcast and Derek Sivers.

As a final sentence, also would like to express my gratefulness to my idol who is Lewis Hamilton.

Abstract

The singing voice is one of the most salient parts of music. Thus, singing voice-related tasks are always significant for different areas in Music Information Retrieval (MIR), such as cover song identification, music recommendation, singer identification. We exclusively focus on singing voice detection (SVD) and gender classification in this thesis as they are two of the most popular tasks. On the other hand, researchers use source Separatios (SS) which is a technique to separate components of the mixture. Thanks to that, we can have cleaner vocal signals, bass signals, etc. We use SS as a pre-processing step for the singing voice-related tasks, and we are expecting to have better results for those tasks by applying SS models. To validate that idea, we utilize the most popular open-source musical source separation models as Demucs, Open-Unmix and Spleeter. Thanks to different models, we can see how the properties of models change the results of singing voice-related tasks. As a dataset, we used Richard Wagner's Ring Opera dataset and Smule Dataset. With those datasets, we can see the robustness of our methods. Our main contributions are that understanding how SS models can help to be better in singing voice related tasks, robustness of SS models with different scenarios and how different SS models perform with those tasks. Thus, we can also see which SS models are better suited to tasks. According to results of singing voice detection experiments, using SS models are mostly beneficial and Spleeter outperforms all other SS models. Also, we investigated the longest errors of classifiers to see possible reasons for them.

Contents

Chapter 1

Introduction

The singing voice is one of the most salient parts of music. Thus, singing voice-related tasks are always significant for different areas in *Music Information Retrieval* (MIR).We exclusively focused on Singing Voice Detection (SVD) and Gender Classification in this thesis, both of which play an important role as pre-processing steps for many applications, such as cover song identification, music retrieval, music summarization, and singer identification.To perform these singing voice-related tasks, we selected Random Forest Classifiers (RFC) as our classifier, because it performs well for the different applications within MIR. On the other hand, MFCC is selected as feature representation because MIR researchers show that MFCC mostly gives the best results when we compare it with alternatives.

In addition to that foundation, we also applied source separation (SS) models as a novel approach to evaluate the results with that additional pre-processing step. We decided to use Demucs, Spleetern and Open-Unmix for the SS models, as they are well-known open-source models. To evaluate how SS can be helpful in improving those singing voice-related tasks, we initially focused on a Wagner Dataset, which includes complex orchestral pieces with many singers and instruments. For SVD, we used three different methods: *SS + Thresholding*, which determines vocal or non-vocal based on energy-thresholding, *only RFC*, which uses just RFC as a classifier without any SS, and *SS + RFC*, which uses SS as a pre-processing step for RFC. Our conclusion is that *SS + Thresholding* performs well, especially with Spleeter, and adding SS into RFC is mostly beneficial. For Gender Classification, we are using *only RFC* and *SS + RFC*. What we found is that adding SS as a pre-processing step into RFC always improves the scores. Furthermore, we worked on the Smule dataset, which consists of clear vocals of non-professional singers. We did not apply SS to them, in order to evaluate whether or not we could achieve clean vocal signals with amateur recordings.

1.1 Thesis Organization

In Chapter 2, we formalize our tasks definitions. Then we give an overview of used datasets, possible feature representations, possible classifiers and evaluation measures. We selected RFC as the classifier and MFCC as the feature representation for our experiments. So we also give details about them in this chapter.

In Chapter 3, we explain the fundamentals of source separation as that is one of our core concepts in this thesis. Then, initially, we formalize source separation and what can we expect from that. In the literature, the two primary methods of source separation are spectrogram based and waveform based source separation. In Section 3.2 and Section 3.3, we are giving detailed explanation about those main methods. In Section 3.4 we are discussing open-source-implementations of those source separation models and some details about their real-world cases.

Then, after understanding the fundamentals of our tasks and source separation, we explain how we can utilize source separation models for selected singing voice-related tasks in Chapter 4. After a description of how we can combine everything and utilize them, in Chapter 5 we are giving results of all of the experiments with detailed explanation. In addition to that, in Appendix A, we can find figures for selected excerpts that are related to those experiments.

1.2 Main Contributions

Our main contributions as follows:

- We utilize source separation models to see can they be helpful for Singing Voice Detection and Gender Classification by removing accompaniment from vocal segments and removing most of the energy from the non-vocal segments. We showed that all source separation models help to have better performance with varying degree for Singing Voice Detection.
- We analyze that differentiation of source separation models in performing that vocal stem extraction and how this differentiation affects Singing Voice Detection and Gender Classification. According to experiments, Spleeter outperforms other source separation models
- We utilize source separation models to see can they be helpful to remove accompaniment from vocal segments to distinguish male and female from each other in case of Gender Classification. We showed that using source separation models as pre-processing step are beneficial for Gender Classification.
- We show that what if we have clean vocal recordings from non-professional singers and perform Gender Classification with them.

Chapter 2

Fundamentals

2.1 Task Formalization

In this thesis, we focused on the understanding effect of source separation methods for *SVD* related tasks which are *Singing Voice Detection* and *Gender Classification*. Before going into details of those aforementioned tasks we firstly would like to introduce **why singing voice is important and interesting**. In addition to that, we would like to share some thoughts about **how it can help to improve another system**.

Singing Voice Detection is important because if we can successfully detect singing voice regions in the piece, we can use those regions for music summarizing [27], music retrieval [6], transcription [47], genre classification [54], audio segmentation - indexing, language detection [24], singing melody extraction [42] and singer identification [20, 33]. On the other hand, a recent study showed that some of the most salient components of music are singers (vocals, voice) and lyrics [9]. Thus, the singing voice is the main focus of attention in musical pieces with a vocal part, most people use the singers' voice as the primary cue for identifying a song [33]. As we can see from those examples, SVD can be considered as a foundation for many advanced applications and can work as a fundamental pre-processing step. For instance, with knowing the singing region, we can improve the performance of singing voice extraction and this helps to get the cleaner vocal signal. With clean vocal signal, singer identification and lyric transcription algorithm will perform better. Also, the commercial industry can benefit from that and the evident area is the karaoke industry which has billions of dollars estimated value [19]. On the other hand, with better lyric transcription, we can find explicit content from pieces without needing manual check and with that, we can help to protect kids from those inappropriate contents [4]. Also with lyric transcription, we can detect the mood of the piece [8, 16]. Furthermore it is possible to understand cultural differences [18], cover song identification and detect language [3] with clean singing voices' features.

Gender identification is helpful for other tasks as pre-processing steps. Especially gender bias is a big problem in music recommendation systems and we can use gender classification algorithms to help this research area with providing more data without need manual labels [45]. Also with more data, we can understand how gender stereotypes evolve across time [11] and determine how some words usage changes by different genders [35].

2.1.1 Singing Voice Detection

Singing Voice Detection (SVD) is a binary classification task to detect which frames have vocals (singing voice) in the given piece. Detecting the vocal regions sounds not a complex task for humans to perform despite the complex nature of music such as dynamic of articulations, instrumentation in the background, singing in the different language, different voice characteristics of singers, extreme range of vocal tone diversity, etc. However, automatic detection of the singing voice is still an unsolved task despite recent improvements in this area [24]. There are two comprehensive literature reviews about this task which we will share.

We have two comprehensive literature-review about SVD. Lehner et al. [24] wrote at 2013 and Lee et al. [22] wrote at 2018. Now, we will go over both reviews to see state-of-art models from those papers. The reason why we choose these two reviews is that they re-implemented methods and also explain how models are working very nicely. Let's start with the first one.

Lehner et. al. [24] compared their proposed models with Vembu and Baumann [52], Mauch et al. [31] and Ramona et al. [40]. Now, we can go over the details of those models.

- ***Vembu and Baumann method*** [52] : MFCC features are extracted with the 0 triangle shaped filters and extracted 13 coefficients, without the 0th coefficient. Also 39 *Perceptual Linear Predictions* (PLPs) and 12 *Log Frequency Power Coefficients* (LFPCs) are used with the SVM classifier.
- ***Proposed Method*** [24] : This is optimized version of Vembu and Baumann method with just using optimized MFCC features (*do not use PLPs or LFPCSs as another features).* MFCC features are extracted with the 0 triangle-shaped filters and extracted 13 coefficients, without the 0th coefficient for each 200 ms frame. Random Forest is selected as a classifier. For post-processing, a simple median filter with a window length of 1.4 seconds (which corresponds to seven frames) is utilized. Then MFCC features and classifier are optimized with the following steps.
  - Number of coefficients and size of filter-bank is optimized. Based on experiments, 30 coefficients with 30 triangular-shaped filters are selected as the most optimized ones.
  - Observation window's length is optimized. 800 ms total window size around the 200 ms centre frame is the best based on results. Also, delta MFCCs will be used as well.

– Parameter of the classifier is optimized. 128 trees with 5 attributes are giving the best results. Also, the threshold is set to 55%

	MFCC	I	II	PROP	VB
acc [%]	69.14	74.51	78.74	82.36	77.16
recall	0.727	0.783	0.834	0.883	0.819
precision	0.712	0.757	0.788	0.810	0.774
f-measure	0.719	0.770	0.810	0.845	0.796

Table 2.1. Parameter optimisations' results which are compared to the VB which is baseline method. Columns of the table represent as following: MFCC: unoptimized (standard) MFCCs. I: after optimisation of number of MFCCs and filterbank as stage I. II: after optimisation of observation window and delta MFCCs as stage II. PROP: proposed method after optimisation of Random Forest and median filter as stage III.

As we can see from Table 2.1, optimized MFCC features can outperform unoptimized MFCC features and other features like PLPs and LPFCs

- ***Ramona et al. method*** [40]: In this method, we have SVM as classifier. The feature set is consists of 116 components and some of them are short-scale features and others are long-scale features. These include MFCCs, LPCs, ZCR, sharpness, spread, f0, and aperiodicity measure via monophonic YIN library. With *Inertia Ratio Maximization with Feature Space Projection* (IRMFSP), they decreased the dimension of the feature set to 40.

- ***Mauch et al. method*** [31]: In this method, SVM is a classifier and there are 4 complimentary features with MFCC features as following.

  – *Polyphonic fundamental estimator* to get pre-dominant melody.

  – *Pitch fluctuation* which is calculated as frame-wise standard deviation for intra-semitone f0 differences. According to experiments, this one is most important for their test set.

  – *MFCCs of the re-synthesised predominant voice* to learn timbre.

  – *Normalized amplitude of harmonic partials* to get more information about timbre which is not available via MFCC.

With Table 2.2, we can see that the proposed method performs not much worse than the method of Mauch et al. (accuracy scores are 85.9% vs. 87.2% respectively). And if a more complex post-processing strategy involves an HMM and the Viterbi algorithm (*PROP+*), that difference disappears.

If we would like to sum up this review, according to all experiments, we can realize that regarding the features, appropriately parametrized MFCCs along with their first derivatives are sufficient to achieve results as good as much more complex state-of-the-art systems. The proposed method

Mauch	accuracy	precision	recall	f-measure
MODE	0.654	0.654	1.000	0.791
MFCC	0.738	0.739	0.926	0.822
FMRH	0.872	0.887	0.921	0.904
Proposed	accuracy	precision	recall	f-measure
MODE	0.604	0.604	1.000	0.753
MFCC	0.718	0.764	0.771	0.767
VB	0.813	0.827	0.808	0.818
PROP	0.859	0.858	0.918	0.887
PROP+	0.868	0.879	0.906	0.892

Table 2.2. Proposed method's results compared to the methods of Vembu & Baumann and Mauch et al. We can see the distribution of classes in the respective test set is given as an addition to the methods (MODE row – vocal's overall proportion), as well as the results which we got via the standard MFCCs, and Vembu & Baumann's Method (VB row). Clearly, despite the majority of the data used is the same as used by Mauch et al., there are significant differences in terms of vocals' content, which causes unfeasible fair comparison. To obtain results of PROP+, post-processing is applied involving the Viterbi algorithm.

is simple, fast, and requires no look-ahead. Thanks to these advantages, we can use that for SVD applications if it needs real-time usage. Also if Viterbi is used as post-processing for the proposed method, results can be improved.

Lee et al. [22] also reviewed SVD methods in 2018 [1]. The authors re-implemented three well-established models to compare their performances. Now, we can see details of those models in the following part.

- ***FE-VD (Lehner et al.)*** [25] : This is feature engineering (FE) based model. They compute six different features (fluctograms, spectral flatness, vocal variance, MFCCs, delta MFCCs, and spectral contraction) for each segment. The feature set includes 116 features per segment, and each segment is 1.1 seconds. They also take account of adjacent frames' features to have more information about the context. Then RFC is used as a classifier. Median filtering is applied with an 800 ms window size as post-processing. According to their experiments, the proposed method works better for strings whereas it has a high error rate for the woodwind instruments like saxophone and pan flutes.
- ***CNN-VD (Schluter et al.)*** [43] : This model is using one of the deep learning models which is called as convolutional neural network (CNN). The proposed model is using 3-by-3 2D convolutional layers and mel-spectogram as input to use local time-frequency patterns. Also, some data augmentation techniques are used like time stretching and pitch shifting to have a more diverse dataset in the original paper but for re-implementation, the authors did not apply those augmentations. 1.6 seconds is chosen as input size.
- ***RNN-VD (Leglaive et al.)*** [23] : This model is using another deep learning model

[1] Lehner et al.'s [24] review is from 2013.

which is called as recurrent neural network (RNN). With RNN, the aim is to hold temporal information across the network and have better contextual information. For RNN architecture, bi-directional long short-term memory units (Bi-LSTMs) is selected because it performs better than vanilla RNN according to experiments. To extract features from pieces, they apply harmonic percussion source separation (HPSS). With HPSS, the audio signal is decoupled into harmonic and percussive components. Then, they are combined to have input.

For mentioned models, Jamendo is used for training, validation, and testing. Models perform quite similarly in terms of accuracy and other evaluation parameters. When they classify errors, the most appearing one is coming from pitch-fluctuating instruments and mostly false positives. Because woodwinds, strings, and brass show similar characteristics with the singing voice.

	FE-VD	CNN-VD	RNN-VD
Acc.(%)	87.9	86.8	87.5
Recall(%)	91.7	89.1	87.2
Precision(%)	83.8	83.7	86.1
F-measure(%)	87.6	86.3	86.6
FPR(%)	15.3	15.1	12.2
FNR(%)	8.3	10.9	12.8

Table 2.3. Experiment results for FE-VD, CNN-VD and RNN-VD with Jamendo Corpus. FPR and FNR refer to false positive rate and false negative rate, respectively.

As we can see from the Table 2.3 , RFC (what we used in FE-VD) can perform a similar performance as CNN (CNN-VD) or RNN (RNN-VD) with proper features for SVD.

2.1.1.1 Singing Voice Detection in Opera

In this thesis, we are specifically focusing on SVD and gender classification in the operas as we mentioned previously and we are using ideas from SVD so we can also check how SVD performs for operas. For that, we will go into the details of two papers in the section.

Towards Cross-Version Singing Voice Detection [10]: In this paper multiple versions of the 24 songs of the cycle *Winterreise* by *Franz Schubert* has been used to understand *can bootstrapping be viable technique to stabilize SVD for multiple conditions* and *with the cross-version fusion approach, can we have better results.* As a baseline model, Lehner et al. [24] is used. We already explained that in the previous section but we can mention most related parts as well. According to the authors of this paper, fluctogram is the most notable feature from Lehner et al. [24]. Basically for fluctuating signals concerning pitch, fluctogram is a good choice. And with the fluctogram, we do not need pre-dominant pitch tracking and it makes the process online. For windows, the analysis window size is 800 ms and hop-size is 200 ms which

means that there are 5 segments per second. Then RFC is utilized with 128 trees and they are trained with 146-dimensional feature sets randomly selected 5 features. As a first pre-processing, 7 frames width median filtering is applied. Also, a threshold is applied with 0.5. Thus, if the predicted value is less than 0.5, we set it into 0, otherwise, we set it into 1. We can see details of bootstrapping and cross-version fusion in the following part.

- *Bootstrapping*: The main idea of bootstrapping is creating automatically adapting classifier models based on initially trained ones. Firstly, we are training the classifier and then use the trained classifier with the test set. Then, some prediction values of the classifier will be not in ideal shape. Values of ideal shape are near the threshold of the selection and we would like to have values that are the same as our edges. Thus, we can select feature vectors' subset based on where they correspond in the decision function values' range and we will select which are close to the edge. At the final stage, those selected subsets will be used for the model.

- *Cross-version*: First of all, we will synchronize our audio segments based on what corresponds same measures/bars in the different recordings. Then we will run our classifier for all versions/recordings and we will get different prediction values for each segment of each version. Then we use their average (for the same segment from a different version) for the new prediction value.

In the paper, the authors trained the model with popular songs from RWC and Jamendo Corpus for bootstrapping. Initially trained with RWC and Jamendo datasets to select feature subsets. Then train with selected subsets with training data which consists of Schubert. With this method, the model gives better results than without applying to bootstrap. But cross-version performs worse than without using it. Also, interestingly best result comes from the combination of bootstrapping and cross-version fusion. In the paper, there is no exact reasoning for that.

Cross-Version Singing Voice Detection in Opera Recordings: Challenges for Supervised Learning [34]: In this work, authors are using a subset of the *Wagner* dataset [2] and using three different state-of-art techniques. For each model, non-overlapping 3 seconds length segments are taken to compute features and 250-band mel-spectogram are computed per segment. We can see details of each model in the following part.

- 0μ ***- CNN***: Mel-filterbank logarithm is used as feature set. Then 2D convolutions and max-pooling are applied via CNN. To achieve sound-level-invariant SVD, [44] introduced zero-mean convolutions which is an update rule that constrains the CNN kernels to have zero mean. That zero-mean update is a fundamental part of 0μ - CNN as well.

[2] "Die Walkure" of Richard Wagner's opera

- ***PCEN - RNN***: Mel-filterbank is used as feature set. Then per-channel energy normalization (PCEN) [53] is applied to have a sound-level-invariant system as suggested in the [44]. The normalization step is followed by recurrent neural networks.

- ***LP - PCEN***: Mel-filterbank is used as feature set. Then low-rank encoder with per-channel energy normalization and Bi-GRUs is applied. Bi-GRUs is a specific version of the recurrent neural networks and can perform better than other variants of RNN [46].

The reason why researchers use zero-mean or energy normalization is that they would like to have a sound-level-invariant classifier. Because the sound level can directly affect the performance of the classifier by adding bias if it is present.

Data Split	DS-1	DS-2	DS-3
Training	Bar, Hai, Kar	Bar, Hai	Bar
Validation	Kar	Kar	Hai
Test	Kar	Kar	Kar

Table 2.4. Data splits used for the experiments.

Data Split	DS-1			DS-2			DS-3		
Models	CNN	RNN	LP	CNN	RNN	LP	CNN	RNN	LP
Precision	**0.95**	0.93	0.94	**0.96**	0.91	0.92	**0.97**	0.87	0.90
Recall	0.91	**0.92**	0.90	0.81	**0.88**	**0.88**	0.69	**0.76**	0.74
F-Measure	**0.93**	**0.93**	0.92	0.88	0.89	**0.90**	0.80	0.81	**0.82**

Table 2.5. Experiment results for CNN (0μ-CNN), RNN (PCEN-RNN) and LP (LP-PCEN) with different data-splits

In addition to that, they have two great findings.

- Loudness dependencies can not be removed with zero-mean or per-channel energy normalization directly. As reported for popular music, singing parts are mostly louder than non-singing parts of the piece. Those cause errors such as false positives *(loud parts can be predicted as singing but it is not)* or false negatives *(soft parts can be predicted as non-singing but it is singing)*.

- Systems can not generalize well across cross-version because of different acoustic characters *(which microphone is used, where we put microphones, the acoustic environment of the building, etc.)* even we have the same recording in the training set. The reason for that can be related to the nature of those pieces. Wagner's operas are challenging especially because of singing styles and orchestration. It consists of highly expressive and different singing styles. On the other hand, orchestration is highly complex and large with many different instruments.

2.1.2 Gender Classification

Gender identification is another binary classification task to label gender of which singer/s is singing in the given piece for whole song. We can see how we are performing that with Figure 4.4 and Figure 4.5.

2.2 Datasets

2.2.1 General Datasets

2.2.2 Wagner Opera

In opera pieces, there are some characters and that situation is quite similar to characters in movies such as *Legolas* in the *Lord of the Rings*. With *character identification* we are trying to label each frame with character names automatically. For operas, we can have different recordings/performances from different times, locations, conductors, etc., and most importantly different singers will sing for the same character in different recordings. We can introduce definition of ***act*** and ***performance*** for better understanding. In operas, we can have different acts in the opera. In the opera's act, we can have the same and different characters. Also, the opera can be performed in different locations, at different times, by different conductors etc. In this case, we have different performances for the opera. For Wagner's Ring opera, different acts consist of the whole opera. Also, we have different performances for that opera like *Karajan, Barenboim, Haitink* which corresponds to different conductors. According to our best knowledge, there is no work about *character identification* and especially for *operas*.

With *Wagner Opera*, we can have different experimental setups. Let's briefly looks at those setups:

- ***Act Split***: Same performance with a different act (focusing on different acts of same performance so the same singer are singing for the character but lyrics will be different, also orchestra is playing different notes etc.)
- ***Performance Split***: Different performances with the same act (different singers are singing for the same character but what lyrics are the same)
- ***Neither Split***: Different performances with a different context

In experiments, we used **Wagner Dataset**. In Wagner Dataset, there are 3 different *performances* for each of 11 *acts* of *Der Ring des Nibelungen* [3]. Those performances are conducted by Barenboim

[3] We will refer as The Ring in following parts.

in 1992 (*Bar*), Haitink in 1988 (*Hai*) and Karajan in 1966 (*Kar*).

The Ring is based on *Norse sagas* which is historic stories from Iceland and also *Nibelungenlied* which is an anonymous epic poem written around 1200 in German and consists of oral stories from individuals and historic events. *Nibelung* in the title of opera corresponds to *Alberich* which is the dwarf and heroic legend of German Mythology. In the *Nibelungenlied*, he is the guardian of the treasure of Nibelung and has the strength of twelve men. In *The Ring*, Alberich is head of the one race of dwarves which is called as *Nibelungen* [2].

There are 4 operas in *The Ring* cycle as *Das Rheingold, Die Walküre, Siegfried, Götterdämmerung* and performance of *The Ring* takes 4 days. We will represent those operas with single letter for sake of simplification as following *Das Rheingold* as "A", *Die Walküre* as "B", *Siegfried* as "C" and *Götterdämmerung* as "D". Also, there are different acts in each opera. For each act of the opera, we will use numbers. For instance, we will represent first act of *Die Walküre* opera as "B1".

Before talking about experiment setup, we can focus on instruments and characters of *The Ring*. There are woodwinds, strings, and brass instruments as an orchestra in *The Ring*. On the other hand, there are many characters and we would like to provide information about the most important characters by mentioning their role in the cycle and vocal range in the Table 2.6 [2].

Character	Role in the Story	Vocal Range
Wotan (Der Wanderer)	King of the Gods	Bass-Baritone
Fricka	Wotan's wife	Mezzo-Soprano
Siegmund	Son of Wotan	Tenor
Sieglinde	Twin sister of Siegmund	Soprano
Siegfried	Son of Siegmund and Sieglinde	Tenor
Hunding	Husband of Sieglinde	Bass
Brünnhilde	One of Valkyries	Soprano
Alberich	Head of Nibelungs	Bass-Baritone
Mime	Brother of Alberich	Tenor

Table 2.6. Overview of Wagner Characters

2.2.3 Smule

Especially for the gender identification task, a newly published dataset extending the DAMP dataset named DAMP-balanced (DAMP-VPM) [49] is used. There are different DAMP datasets based on different criteria. In this thesis, we used the DAMP-balanced dataset which has 24874 *unaccompanied* solo singing recordings from 5429 singers singing a collection of 14 songs [17]. According to our knowledge, this is the largest available dataset of its kind. That dataset originating from the social music-making company, *Smule*. Users of Smule created this dataset with vocal-only recordings from mobile phones. So singers are not professional. On the other

hand, for some pieces, we can have some background noise and possibly some glitches from the accompaniment. We will explain how we are coping with them, especially background noise.

The list of songs in the Smule and how many recordings we have for that song is as follows. They are some popular *pop* songs.

Artist	Song	Count in the Dataset
Charlie Puth	One Call Away	3912
James Arthur	Say You Won't Let Go	3225
Justin Bieber	Love Yourself	3019
Lukas Graham	Seven Years	2942
The Chainsmokers	Closer	2873
John Legend	All of Me	2856
Clean Bandit	Rockabye	2737
Luis Fonsi	Despacito	1287
Ruth B.	Lost Boy	1183
Extreme	More than Words	586
Jodi Benson	Part of Your World	56
Bruno Mars	When I Was Your Man	56
Sia	Chandelier	56
Anna Kendrick	Cups	56

Table 2.7. Overview of Smule Dataset

2.3 MFCC

For our experiments, we have audio files from different datasets as we introduced. To perform singing voice-related tasks we need to convert them into some input representations and this is a classification task so we need a classifier to classify those input representations into labels that represent which character is singing or no character is singing in that segment. In addition to them, we need to have evaluation measures to understand how our model is performing. To sum them up, we need extracted features as input, model as the classifier and evaluation measures to understand how good it is.

We will extract features from audio signals and give them to the classifier as input and we can have different features. MFCC is one of the most used features in singing voice related tasks. In our experiments, we are also using MFCC. On the other hand, other well-known features are the following and we share a very brief summary about them.

- **Fluctogram**: It is using sub-semitone fluctuation as a discriminative feature.
- **Vocal Variance (VV)**: Measure of variation of the selected subset of MFCC coefficients.

- **Spectral Correlation (SC)**: Measure of reliability of the information which is in the selected frequency band.

- **f0**: $f0$ represent the fundamental frequency of audio signals and most researchers use it for pitch estimation and tracking.

Mel Frequency Cepstral Coefficients (MFCC) is a representation for the power-spectrum of a sound and is based on discrete cosine transform which is happening on a non-linear model-scale. MFCC represent *envelope of the power spectrum* of audio an it captures *perceptually important information* in other words timbral aspects. While originally developed to decouple vocal excitation from vocal tract shape for automatic speech recognition [36], they have found applications in other auditory domains including music retrieval [13, 26].

The concept and calculation of MFCC was explained by [12, 29] as follows.

- Frame the signal into short frames. In literature, mostly 20 ms is used as frame length and the reason is we can assume that signal is stationary in that range.

- Calculate periodogram estimate of the power spectrum for each frame. For that, we are taking Discrete Fourier Transform (DFT) of the frame. $s(n)$ is our time-domain signal, $h(n)$ represent the window which has N sample and we are using Hamming window for application, K is the length of the DFT, i denotes frame number. In our experiments, we set N as *2048* and K as *2048* as well.

$$S_i(k) = \sum_{n=1}^{N} s_i(n)h(n)e^{-j2\pi kn/N} \quad 1 \leq k \leq K \tag{2.1}$$

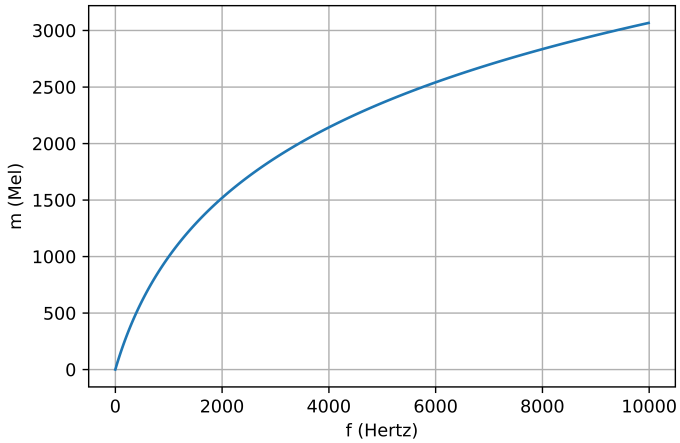
  The previous step also corresponds to *Short-Time Fourier-Transform (STFT)*. Then periodogram is calculated with the following formula:

$$P_i(k) = \frac{1}{N} |S_i(k)|^2 \tag{2.2}$$

- We are applying Mel-filterbank into *periodogram* and then getting the sum of each filter's energy. The reason why we used Mel-scale is that is related to perceived frequency (pitch). Human anatomy is specialized to understand smaller differences in lower frequencies than higher frequencies. So we are not using *Hertz-scale* and using *mel-scale*. In Figure 2.1, we can see that how Mel-scale and Hertz-scale are related to each other. The relation between *mel-scale* and *hertz-scale* looks logarithmic and is similar to how human anatomy is working with frequencies.

  For mel scale, we are using following formula (f is frequency in hertz and m is frequency

Figure 2.1. Mel scale versus Hertz scale



in mel) :

$$m = 1125 \cdot ln(1 + \frac{f}{700}) \tag{2.3}$$

$$f = 700 \cdot (e^{(m/1125)} - 1) \tag{2.4}$$

Let's see how we can calculate mel-filterbank,

- We are converting the lowest and upper frequency of our range of interests (will be 0 to 11.025kHz for our experiments) into mel-scale with the Equation 2.3. For instance, if we would like to have mel-value for 11.025kHz, we can calculate as follows

  $$1125 \cdot ln(1 + \frac{11025Hz}{700}) = 3170.69Mel$$

  For 0 Hz, we will have 0 Mel from the conversion.

- Then based on the count of filterbank (in the literature it is 26 most of the time and we will also use 26 filterbanks in our experiments) we select additional points as spaced linearly between those limits in the mel scale. For *26* filter-banks, we select 28 points between our range of interest which we decided in the previous step and put those pieces linearly in mel-scale.

- Then we convert those 28 points which corresponds to mel-values into hertz again with Equation 2.4. With this step, we have new variable which is $h(f)$.

- We can not use those values directly because we do not have infinite frequency resolution. So, we will map them into nearest FFT bin. We are using following

formula ($nfft$ represent number of FFT bins)

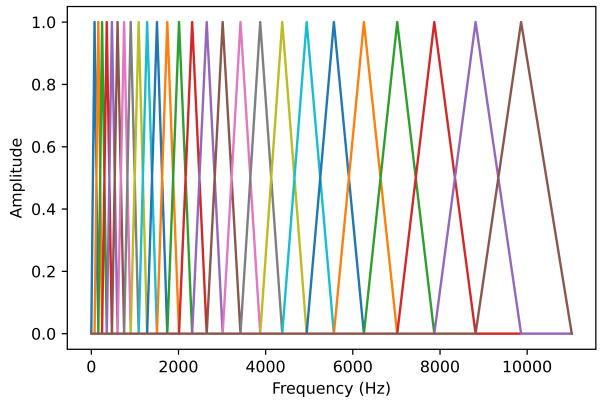
$$f(i) = floor(\frac{(nfft+1) \cdot h(f)}{samplerate}) \tag{2.5}$$

– Finally we can calculate each filter bank with the following formula. (m represent number of filter) With this calculation, we will have triangular filters.

$$H_m(k) = \begin{cases} 0 & k < f(m-1) \\ \frac{k-f(m-1)}{f(m)-f(m-1)} & f(m-1) \leq k \leq f(m) \\ \frac{f(m+1)-k}{f(m+1)-f(m)} & f(m) \leq k \leq f(m+1) \\ 0 & k > f(m+1) \end{cases} \tag{2.6}$$

If we use 26-filter in the filterbank, we will get Figure 2.2. As we can see from that, for lower frequencies we have thinner triangles. If we check higher frequencies, we have thicker triangles. This is related with how mel-scale and hertz-scale is related as we described previously and in Figure 2.1

Figure 2.2. Plot of Filterbank which has 26 Filters



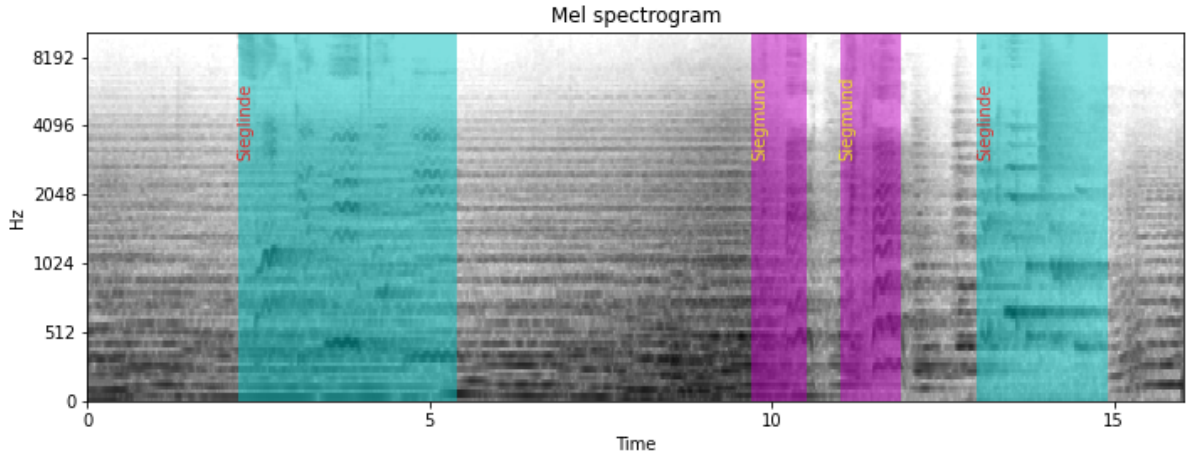
To get filterbank energies, we are multiplying the periodogram with each filter in the filterbank.

- After having a mel-filterbank result of the audio signal, we are taking the log of each filter (energies). With this step, we got *mel-spectogram* of our signal.
- Finally, we are applying Discrete Cosine Transform (DCT) of our filterbanks to get cepstral coefficients as the number of filter-banks. (For 26 filterbanks, we will have 26 cepstral

coefficients per frame.) Then we can select which coefficients will be used in the application. For instance, we use 26 filters to get Mel-filterbanks and we have 26 coefficients after DCT. But just a lower 20 of them are used for applications and also for our experiments in this report. Because the lower coefficients of the DCT are used to represent a rough shape of the spectrum and the zeroth coefficient represent voiced frames detection [4] [33].

To see how mel-spectogram and fluctogram look like we will introduce their figures for the excerpt from *Karajan* performance of act *B1*. We selected 297. seconds to 313. seconds of that recording. Because there are two characters in this excerpt as *Sieglinde* and *Siegmund* [5]. Also, the orchestra is not dominant so the effect of the singing voice can be easily seen. This *excerpt* become our *running example* for this chapter.

Figure 2.3. Plot of Mel Spectogram for the Running Example



2.4 Random Forest Classifiers

As we explained in the MFCC section, we need to have a classifier to classify our inputs (a.k.a features) into outputs (a.k.a labels). Those classifiers are one of the core parts of our model for singing voice related tasks. In this section, firstly we will briefly discuss supervised learning algorithms, then which algorithms are used in the singing voice-related tasks and will give greater details of RFC as that is our classifier across all experiments of this thesis.

[4] The very first coefficient in the MFCC (*0th coefficient*), does not hold any information which is related to the spectrum's overall shape. That *0th coefficient* just has information related to constant offset which is the value that is adding into the entire spectrum. So, many researchers discard this coefficient from MFCC values. *0th coefficient* convey information about signal loudness's overall measure.

[5] For more information about definition of performance, act and characters, please check Section 2.2

In machine learning algorithms, there are four different types of learning as supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning. In our experiments, we are going to focus on supervised learning as we have labelled data. Supervised learning, each input x_i is associated with the label or the target y_i in training stage $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ For example, we know which segment has singing voice or which vocal segment is coming from the female vocal. So, those labels are like the instructor and can supervise our classifier to what should it do. Thanks to those labels, it tries to find a function that can map inputs into outputs via patterns. Then when we give unseen data to that classifier, it should use that function to predict what is the label of the given input. So actually it is trying to generalize what it learned into unseen examples. In the literature, supervised learning is mostly used and that is also true for singing voice related tasks. There are some of them as follows.

- Support Vector Machines (SVM): Support vector machines are trying to create decision boundaries with hyper-planes to separate classes. While we are training it, it tries to minimize loss. It can have different kernels which represent a type of decision boundary [48]. We can see its application for SVD in [30, 40, 41, 52].
- Recurrent Neural Network (RNN): Traditional neural networks can only process current information and can not remember past information. Recurrent Neural Networks (RNN) solve this problem thanks to the recurrent connection via loops at their nodes. There are different versions of RNN like Long Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) [46]. RNNs are used for many different applications and we can see their application for SVD in [23, 34].
- Convolutional Neural Network (CNN): Convolutional neural networks consist of different layers. The most important one is convolutional layers and it applies convolution into our inputs with its weights and biases. Then we can concatenate more convolutional layers to have a complex representation. We apply the activation function to introduce non-linearity, we apply to pool to compress and aggregate pieces of information and the last layer is fully connected to have class-based predictions. While we are training system model updates its weights and biases to minimize loss. It learns better representation for input features [5]. CNN is also used for many different applications and we can see its application for singing voice-related tasks in [21, 34, 43].

For our experiments, we decided to use Random Forest Classifier (RFC) as that works very well for singing voice-related tasks like SVD [10, 24, 25]. After the explanation of other most used algorithms and mentioning that we are going to use RFC, we can focus on details of RFC. In layman terms, RFC is a kind of meta-estimator that used several different classifiers which are decision trees. Training of RFC is done via using different sub-samples of the selected dataset for different trees. In the prediction stage, it averages predictions of those several classifiers and

makes a prediction. Thanks to this averaging we have better accuracy. Also with using several decision trees and different subsets of the dataset, it can cope with over-fitting [1]. We can also see that how we are using RFC with the Figure 2.4.

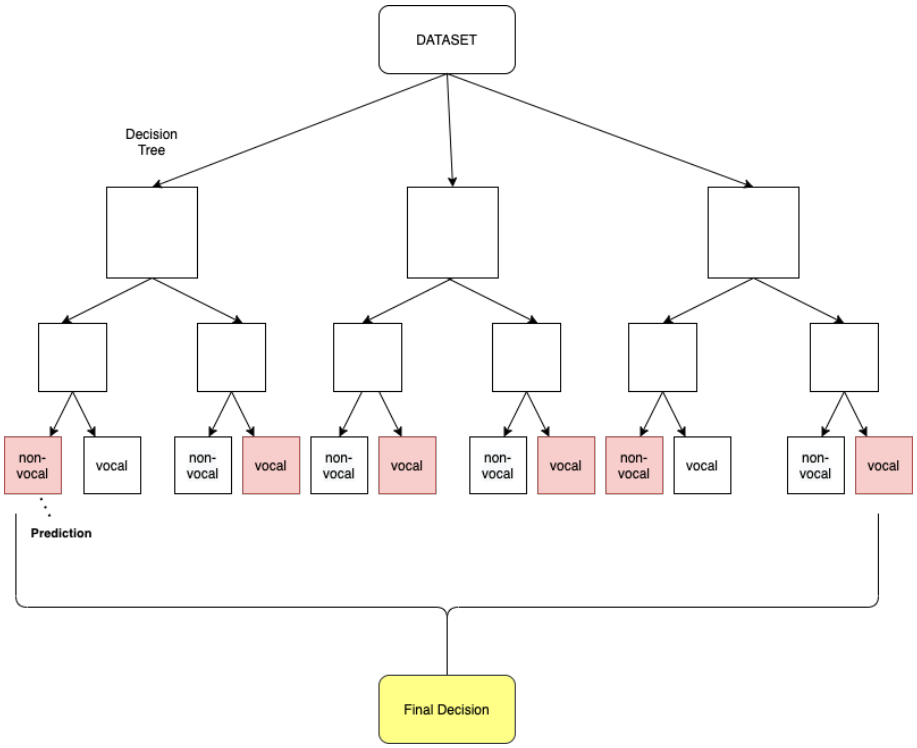


Figure 2.4. Random Forest Classifier Example for Singing Voice Detection Task

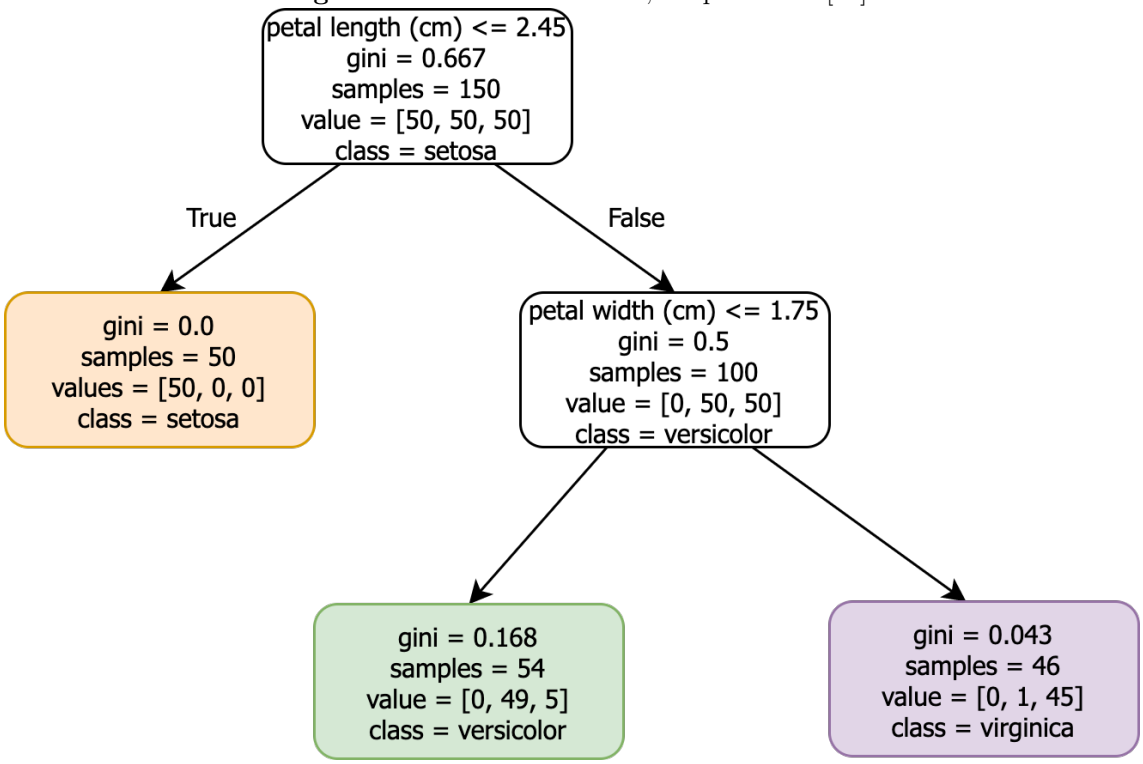
After presenting layman definition and figure of how it works basically, we are going to express more details of RFC. The concept of decision trees and Random Forest Classifiers was explained by *Aurelien Geron* in *Hands-On Machine Learning with Scikit-Learn and TensorFlow* [14] as follows.

2.4.1 Decision Trees

Decision Tree has a flowchart-like structure. Each node represents a *test* which is asking a question to decide for an attribute. For instance in the Figure 2.5 [6] we are asking a question for *is petal length of flower less than 2.45 cm root node* (which is at the top and *depth* = 0). If the answer to the question is yes, we move to the left child node of root (depth 1, left). In this situation, we reached the leaf node as there is no children node and we can not move anywhere. So we stop to ask questions and we just look predicted class of that node. That is Iris-Setosa for

[6]This example is related with Iris Dataset which contains per 50 instances for each 3 classes. Each class is associated with the type of iris plant which can be Iris Setosa, Iris Versicolour or Iris Virginica. There are 5 attributes as sepal length, sepal width, petal length, petal width and class. So basically with the classifier, we are trying to predict the class of iris plant with those attributes.

Figure 2.5. Iris Decision Tree, adopted from [14]



that case. As we can see from this example, we are testing *attributes* and with each branch, we are getting the outcome of the *test*. We continue to ask a question until reach *leaf node* and class labels are represented by those leaf nodes. So from root to leaf, we have a tree and it decides to the class label of our input with some questions to test attribute and each question is part of our tree. Also, it creates decision boundaries as we can see from Figure 2.6.

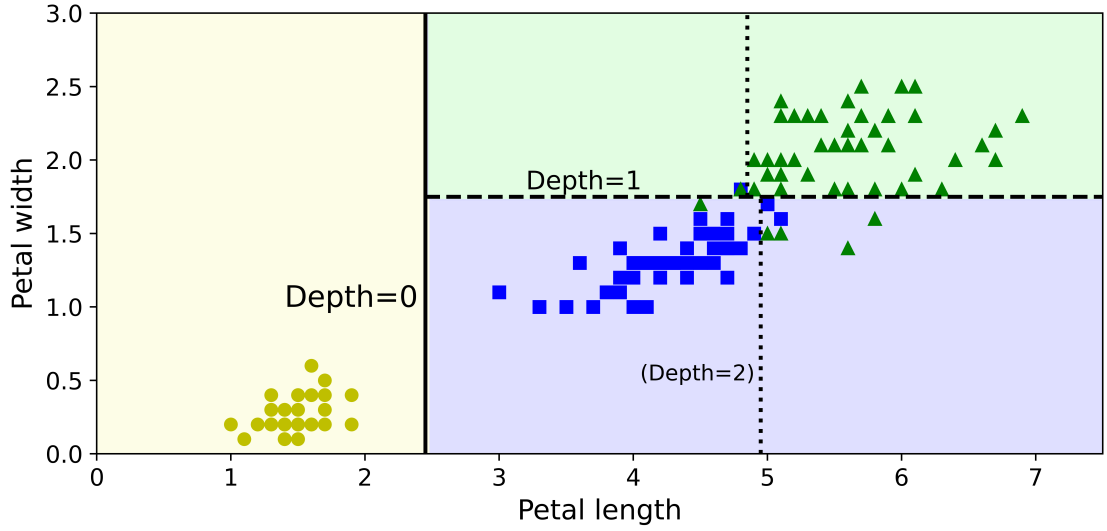
In our experiments, we are using scikit-learn implementation and it is using *CART (Classification And Regression Tree) algorithm* to train Decision Trees. The idea of *CART* is rather simple and it is trying to split our training set into two subsets (we can call that as binary split) using a single attribute k and threshold t_k Now, the question will be how it can choose k and t_k. To find them, algorithm searching for the k and t_k pair which give purest (mean lower score of G_i) subset which is weighted by size as we can see from the following equations.

$$G_i = 1 - \sum_{k=1}^{n} p_{i,k}^2 \tag{2.7}$$

where $p_{i,k}$ is the ratio of class k instances among the training instances in the i^{th} node.

$$J\left(k, t_k\right) = \frac{m_{\text{left}}}{m} G_{\text{left}} + \frac{m_{\text{right}}}{m} G_{\text{right}} \tag{2.8}$$

Figure 2.6. Decision Tree decision boundaries, taken from [14]



where $G_{\text{left/right}}$ measures the impurity of the left/right subset, and $m_{\text{left/right}}$ is the number of instances in the left/right subset.

With those equations, the CART algorithm splits the training dataset into two subsets recursively and it stops when reaches maximum depth which is defined by *max_depth* hyper-parameter, or there is no split which can reduce impurity [7].

In the Figure 2.5, We can see gini attribute G_i measures in nodes and which actually measures *impurity*. *gini* can be from 0 which means all instances of training set belongs to the same class like *depth-1 left node* as just have *setosa* to 1 which is complete impurity as all of them. In the *depth-2 left node*, we saw 0.168 gini score and we are calculating that like $1 - (0/54)^2 - (49/54)^2 - (5/54)^2$. We can also have *entropy* as a measure of impurity, however, we are not using it in our experiments. So we are skipping it.

As advantages of Decision Trees, they are powerful, not completely black box so we can interpret with it and versatile. However, it comes with a few limitations. First, it is doing binary splitting so it has orthogonal decision boundaries and which cause to it become sensitive to rotation in the training set. If we look at the bigger picture, the Decision Tree's main issue is they are very sensitive to small variations which are in the training data. To cope with that instability, Random Forest is introducing getting average of many Decision Trees.

[7] There are also other hyper-parameters which can control stopping conditions but that is out of this thesis scope.

2.4.2 Random Forests

As we can see, decision trees are powerful but it is suffering from instability. Thus, we are using a combination of those trees. Assume that, we are asking a complex question to hundreds of random people and then averaging answers of them. In most cases, that aggregated answer outperforms the expert's answer. In literature, this phenomenon is called as *wisdow of crowd*. In our case, similarly, we can aggregate predictions from many predictors and outperforms the best individual predictor. In this case, we are calling a group of predictors an ensemble and this technique as ensemble learning. Random Forest is also an ensemble method and we are a training group of decision tree classifiers as predictors via a different random subset of training data. Then, when making predictions, use the majority vote of those classifiers. Specifically, ensemble methods' core principle is based on randomization to introduce random perturbations into training procedure to create many different models from a single training set L and combine that prediction to build prediction from that ensemble [28].

To have a diverse set of classifiers with the same training algorithm as decision trees, we can use the training set's different random subsets. If we apply sampling with replacement, that is called *bagging*. After training of all predictors, ensemble (a.k.a bagging of decision trees) of them can predict for a new instance just by aggregating predictions of all predictions. Most of the time, that aggregating function is statistical mode [8]. When we use the aggregation technique, which can help to reduce both bias and variance of single predictor based classification. Generally speaking, the ensemble has lower variance but similar bias than single predictor when both trained on original training data. This means that ensemble and single decision trees are making the same number of errors in the training set but ensemble-based one has less irregular decision boundaries.

Random Forest is mostly based on *bagging* as well. However, the Random Forest algorithm introduces extra randomness than bagging decision trees when growing trees. As we introduced before, decision trees are looking for the very best feature to decide when splitting a node, however, Random Forest is searching for a random subset of features' best feature. Thanks to that, it has greater tree diversity and again we are getting lower variance despite having a higher variance. This is the result of the bias-variance trade-off. With lower variance, random forests produce a better model overall.

For the implementation of RFC, we are using the scikit-learn library [37]. In that one, there are several parameters of RFC and we will focus on parameters that are in the following list.

- $n\_estimators(est)$: How many trees in the forest.

[8] Statistical mode is similar with a majority vote. Mode of the sequence is the most frequent item in that sequence.

- *max_depth*(*depth*): The maximum depth of the tree. If we do not set any value for this parameter, the tree will be expanded until all leaves are pure or contain less than a minimum number of samples. (which is 2 as default)

- *class_weight*(*weight*): Any weights associated with classes to tackle class imbalances. As a default, it is None. Which means every label has the same weight. We can also have *balanced* as *weight* parameter. With the *balanced*, weights are inversely proportional to class frequencies.

2.5 Silence Supression with RMS

For some experiments, we would like to understand how we can label one segment as silence or non-silence. For that, we need some metrics to calculate the energy of that segment and threshold to decide what is that silence or non-silence boundary. To calculate the energy of the segment, there are different ways but we are going to use RMS (root-mean-square) calculation. For that calculation, we are using Librosa [32] implementation. Librosa can calculate RMS via audio samples or spectogram values. We are going to use audio samples as it gives faster calculation because we do not need to STFT for that. For, RMS calculation via audio signals, basically we are getting the mean of the square of the absolute value of the segment's amplitude values. Then we are getting the square root of that value. On the other hand, *hop-length* and *frame-length* are other parameters for calculation. We are using 512 samples as *hop-length* which means 23 milliseconds and 2048 as *frame-length* which means 92 milliseconds. As a threshold, we are using 0.001 and we decided that by listening experiments. For instance, we were wearing headphones and set the volume highest possible. Then if the energy of the segment is less than 0.001, it is inaudible.

2.6 Evaluation Measures

We would like to explain the meaning of metrics and how we can compute them. In the following, TP means the number of true positives, TN means the number of true negatives, FP means the number of false positives, and FN means the number of false negatives. We can explain what they mean in SVD as an example. True positive means that prediction is *singing voice* and it is correct, false positive means that when the prediction is *singing voice* and it is wrong. On the other hand, false negative means that prediction is *non-singing voice* but the correct label is *singing voice*.

Accuracy: Percentage of predicted labels of samples are matching with the ground truth labels.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision: Number of true positives divided by the summation of true positives and false positives.

$$Precision = \frac{TP}{TP + FP}$$

Recall: Number of true positives divided by the summation of true positives and false negatives.

$$Recall = \frac{TP}{TP + FN}$$

F1-score (F1): Weighted average of precision and recall

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

Also, we will provide micro-F1-score to show overall performance. Micro-F1-score is the same with *accuracy*. To calculate accuracy, we divide the sum of true positives by the total count of frames.

Chapter 3

Fundamentals of Source Separation

3.1 Formalization of Source Separation

Source separation is the way to get each component of a mixture of multiple sources and that is similar to the cocktail party[1] problem. For the sound application, we try to get individual sounds from a mixture of multiple audio sources. As an example, we can extract singing voices from the song to use it for different applications to have clean vocal recordings or we can just exclude singing voice and just get all others. Then, we can use this background music in different applications like the karaoke application. As another example, we can just get percussion/drum parts from the recording and which can be helpful for drummers to just focus on that and practice over it. Following is out of our thesis scope but source separation can be used in Speech Enhancement as it can exclude background noise and other non-wanted speeches. As we can see from those examples, we can understand that source separation is like reverse engineering. Now, we will see formal definition in the following paragraphs.

Let $x \in \mathbb{R}^{2\times T}$, with $T \in \mathbb{N}^*$ the number of samples, be a stereo signal composed of the source signals $s_i \in \mathbb{R}^{2\times T}, i \in \{1, 2, \ldots, K\}$, with $K \in \mathbb{N}^*$ the number of sources:

$$x = \sum_{i=1}^{K} s_i \tag{3.1}$$

The goal of audio source separation is to determine the source signals s_i given only x.

That source separation problem is hard because of different factors. First of all, we have *underdetermined* problem. So we have less mixture than our source count in the mixture. For

[1] How the human brain can separate surrounding sounds into its sources and can understand what people are talking about.

instance, we just have a stereo recording (2-channels) but we can have 4 sources [2] in it as vocal, piano, bass and drums. That example can be populated, however, another problem for that task is mixing is the highly complex and non-linear algorithmic procedure. So we can not extract those sources with some kind of basic assumptions [3]. With those signal processing method based approaches, we have model-based approaches. In the past, researchers mostly focused on separation singing from mixture with vocal-accompaniment separation. So methods that perform well for speech enhancement has been used as we are trying to separate speech from noise in that case. Thus, non-negative matrix factorization and Bayesian models were used [39]. But if we check their experimental results with some evaluation metrics and if we listen to those outputs, they are not performing well. On the other hand, sources of musical recordings are highly correlated. This is not the case for speech-related tasks but the music recording, sources of the mixture are changing at the same time for most of the time. For instance, if the sounds of the piano are changing at that segment of the piece, other sources like vocals, bass etc. will change as well. So those sources are correlated with each other. On the other hand, those model-based approaches can work well with linear generated mixtures but in the real-world recording, we have non-linear distortions and dynamic range compression. Also, those models are just focusing on vocal-accompaniment separation and not trying to extract bass, percussion etc. stems. On the other hand, music source separation is a very hot topic for the MIR community and also it is widely used in industry. There are some challenges about that and the latest one from AIcrowd [4] is held by Sony which is an industry leading company.

As we explain in the previous paragraphs, it is hard to cope with musical source separation just by modelling those sources with some assumptions. On the other hand, in recent years, there are significantly more data is accumulated and we have greater compute power thanks to GPUs. So researchers are trying to have better methods with data-driven approaches which are using deep-learning methods. Thanks to that, we can achieve way better performances in music recordings because it can cope especially with non-linearity and can learn how separation should be performed from data. So it does not base on assumptions of model-based approaches. In the upcoming sections, we are going to explain them based on the input type of those methods as waveform and spectogram based. For those deep-learning-based approaches, we have encoder-decoder type architecture. This architecture consists of 3 different modules and we will see their details in the following.

- **Encoder**: It takes mixture signal as input and converts it into different feature representations which will be used in the separation module.
- **Separator**: This is called a masking module as well because with this module we are

[2] We can call those sources as a stem in the musical-source-separation task.
[3] Or we can call it with prior knowledge with signal processing methods
[4] https://www.aicrowd.com/challenges/music-demixing-challenge-ismir-2021

calculating mask which will be applied into the encoder's outputs to extract the wanted source from the mixture. A mask is a matrix that has the same shape with input of the encoder's shape. Then, values of the mask of that source mean the proportion of how much energy of the original mixture will contribute to that source. As an example, if the value in that TF bin of the mask is 1.0, everything in that TF bin contributes to that source but if that is 0.0, there will be nothing taken from that bin. There are two different types of masks. One of them is binary masks and it can just have values as 0 or 1. This assumes one TF bin can just include one source which is called as *W-disjoint-orthogonality*. This is not used that much recently and we have also soft masks. It can take any value from $[0.0, 1.0]$.

- **Decoder**: It reconstructs to exact signal from output of separator. We are applying that mask with element-wise-multiplication in decoder stage. Basically we are multiplying mask with the given input of Encoder. So if a mask, $\hat{M}_i \in [0.0, 1.0]^{T \times F}$, represent ith source of, S_i , in the mixture represented by magnitude spectogram, $|Y| \in \mathbb{R}^{T \times F}$, we can make an estimate with

$$S_i = \hat{M}_i \odot |Y| \tag{3.2}$$

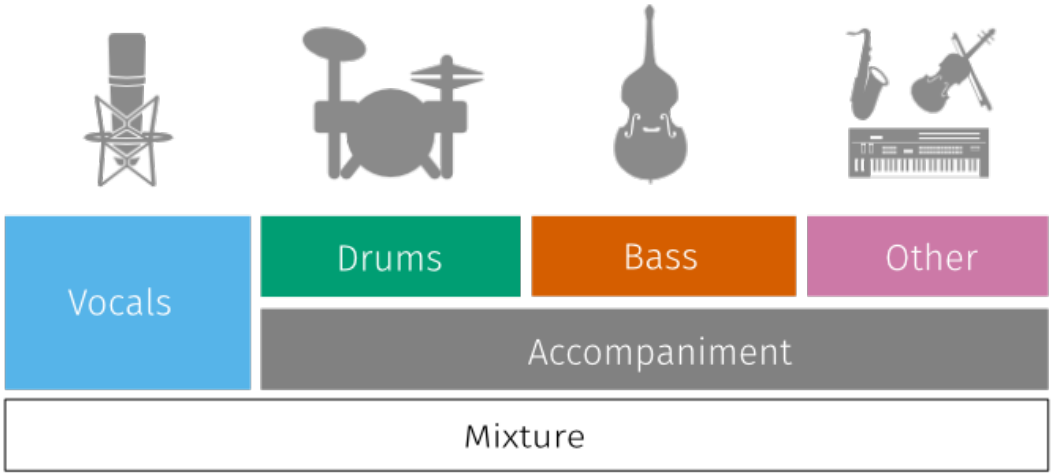
  In addition to them, summation of those mask should equal to 1 for each bin and we can represent it as following for the case which we have N sources

$$J = \sum_{i=1}^{N} \hat{M}_i \tag{3.3}$$

Spectogram and waveform based models have differences in terms of features in encoder and decoder. So basically it is related to how we represent audio in the musical-source-separation model. For spectogram based methods, we are using time-frequency (TF) representation like STFT, Mel-Spectogram etc. which are reconstructable without any loss. For waveform based methods, we are using waveform as input of the encoder and all modules learn how that should be represented jointly. So basically, with spectogram based models we are doing some pre-processing for those mixtures but waveform based models use raw inputs which is the waveform.

Before going into details, just would like to mention that MUSDB18 [38] is mostly used for those data-driven algorithms. MUSDB18 is a dataset of 150 full-length music tracks. Which include 4-stems as Vocal, Drum, Bass and Others. It has different genres in it but mostly there are Pop/Rock and Heavy Metal pieces. On the other hand, some of the implementations are trained via internal data of those companies/schools so we can not reach them publicly.

Figure 3.1. Overview of Stems of MUSDB18, taken from [38]



3.2 Spectogram Based Methods

As we explained in the previous section, we are going to use TF representations as input of spectogram based methods. One of the first papers that try to separate singing voice from the mix is Jannson's U-Net based architecture [19]. It is using U-Net architecture -*encoder & decoder network which uses CNN and skip connections, then estimating masks for given sources*- which is well-known for semantic segmentation and especially used for image segmentation. It is getting the magnitude of the spectogram of the mix and trying to estimate the mask for singing voice's spectogram. Then we can reconstruct the waveform for that singing voice via that mask. After that paper, researchers also tried to implement models which can give different stems like bass, percussion etc. instead of just vocal and accompaniments. According to evaluation measures around those models, one of the best models is MMDenseLSTM [51] which combines DenseNet (which is CNN architecture and perform significantly well for image recognition tasks.) and LSTM. MMDenseLSTM can outperform the Ideal Binary Mask oracle model, however, that does not have open-source implementations. The most well-known open-source implementations which use spectogram based inputs are in the Open-Unmix [50]. Which is designed as a baseline for especially musical source separation and other researchers can easily extend/change some parts of it. On the other hand, it also has a Speech Enhancement model and tries to isolate speech signals from noise etc.

Another very well known open-source implementation of the spectogram based model is Deezer's Spleeter [15]. We will explain the implementation details of it in the upcoming section. But as an overview, it is similar to Jansson's U-Net but this time it is trying to estimate masks for

different stems as well.

3.3 Waveform Based Methods

As we saw in the previous section, we can have TF representation as to the input type of the model. The most-used TF representation is the magnitude spectrum. Another well-used method type is Waveform based for musical source separation. As names imply, we are using waveform as input of the encoder and directly with the raw form of that in the model. So we do not need to apply some kind of pre-processing to convert that into some TF representation etc. Thus, we do not lose phase information. Because when we have a spectogram, we both have magnitude and phase information but when we use a magnitude spectogram of it, we are losing that information. Due to this loss, waveform model-based is used and it is promising to keep phase information.

In the recent research, we can see that waveform based models can outperform spectogram based models for different tasks. As an example, with ConvTasNet, researchers can beat the Ideal Ratio Mask Oracle model for speech separation. Later on, the researcher tried to use ConvTasNet for musical source separation and it gives better results than MMDenseNet based on experimentations [7].

After that, Demucs is just released and which gives the best performance in the MUSDB18 test set. We will give a detailed explanation in one of the upcoming sections. But before that, would like to give a brief overview of the modified version of ConvTasNet as this is the first waveform based method that gives promising results for musical-source-separation. The encoder is just doing one convolution and the decoder is just applying transposed convolution as its symmetric encoder. On the other hand, the separation module is consists of the temporal convolutional network which has stacks with convolutional blocks and dilation factors of them are increasing. In that stacked layers, also skip connections and residual connections are used.

Basically, we are trying to minimize our loss function. As the overall framework, each source s is waveform and represented by $x_s \in \mathbb{R}^{C,T}$ where C represent number of channels (for mono which is 1, for stereo which is 2) and T is sample count of the waveform. In the mixture, we have summation of all sources as $x := \sum_{s=1}^{S} x_s$. So with waveform based methods, we are trying to train a model g which is parameterized by θ as $g(x) = (g_s(x;\theta))_{s=1}^{S}$, where $g_s(x;\theta)$ is the predicted waveform for source s given x, that minimizes

$$\min_{\theta} \sum_{x \in \mathcal{D}} \sum_{s=1}^{S} L\left(g_s(x;\theta), x_s\right) \tag{3.4}$$

3.4 Open Source Implementations

There are different implementations of the aforementioned methods but according to different experiments Open-Unmix, Spleeter and Demucs are very well known and ready-to-use. So, we will explain their details related to details of those models, required memory usage for some examples, how many seconds/minutes do we need to get outputs from them etc.

Model	Domain	Extra-Data
Open-Unmix	Spectogram	No[5]
Spleeter	Spectogram	Yes
Demucs	Waveform	Yes

Table 3.1. Open Source Models and Their Domains

3.4.1 Open-Unmix

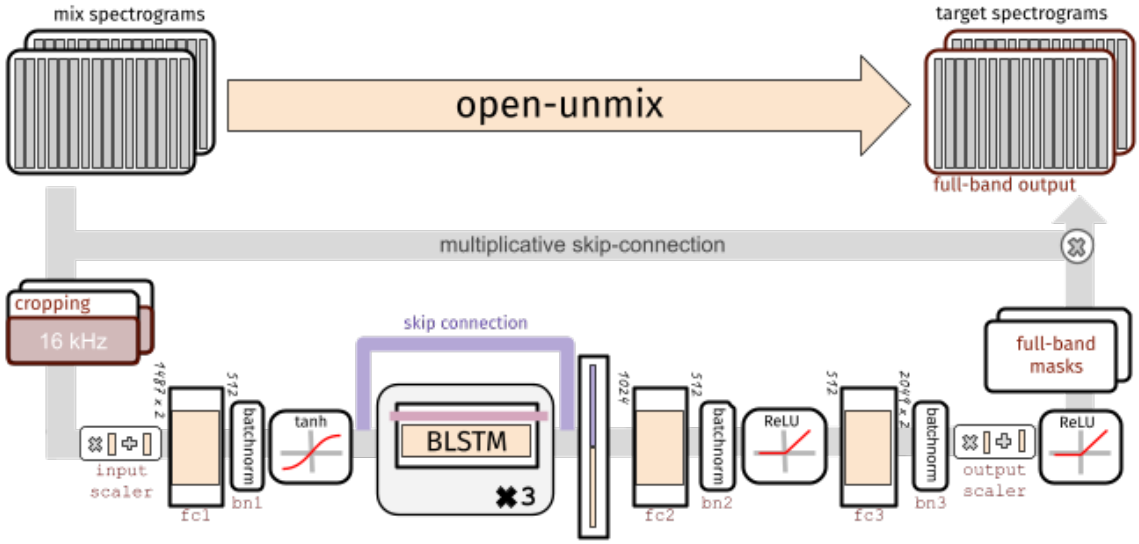
Open-Unmix is the implementation from *SigSep* and according to the definition of it, it has the best usage for pop music. This model can produce vocals, drums, bass and other stems from the given mixture. It does not use any extra data than the MUSDB18 dataset. Also, it is mainly designed for baseline. Thus, the researcher focused on simplicity rather than performance.

Now, we can also focus on details of architecture. The core of the Open-Unmix model is LSTM layers which are kind of Recurrent Neural Networks (RNN). Traditional neural networks can only process current information and can not remember past information and RNN solves this problem thanks to the recurrent connection via loops at their nodes. So, it is used in this implementation. Then, we have fully connected (FC) layers which are responsible for controlling of parameters of the model by doing dimensionality reduction/augmentation in the encoder/decoder. They are using skip connections for two different things. Firstly augmenting outputs of recurrent layers with their input and helps to better performance. Secondly, it is used when we are calculating output spectogram via element-wise multiplication of calculated mask and input spectogram. Thanks to that, the network is pushed to learn how much each TF bin should contribute to the output of that target source. To have non-linearities, we have *tanh*, $ReLU$ and *sigmoid* activation functions in the network. In addition to them, we have batch normalization and it helps to make each batch have a similar distribution. This is highly important if you have so dynamic data like musical signals. To train that model, L2 loss between actual and estimated magnitude spectrogram of each stem is used. We can see details of implementation in the Figure 3.2.

To use that model out-of-the-box, we just need to install *openunmix* python package or docker image and then it can take *.wav* input directly.

[5] At 03/07/2021, they released also model which use Extra Data. We will use latest one.

Figure 3.2. The Architecture of Open-Unmix Model, taken from [50]



3.4.2 Spleeter

Spleeter is a library for waveform based music-source-separation and which is developed by Deezer. In their out-of-box implementation, there are 2-stems (vocal-accompaniment separation), 4 stems (vocals-drum-bass-other) and 5 stems (vocals-drums-bass-piano-other) models as available. Also, it is easy to train new models or fine-tune pre-trained models with more data that can be specific for some genres etc.

In their paper, there are not so many details of implementation but we have an overview about them and now we will share it.

Those mentioned pre-trained models are based on U-nets and have similar architectures with In Spleeter architecture, there are 12-layer U-nets (6 layers for the encoder and 6 for the decoder. This U-net is estimating soft masks for each target source and to train it L1-norm loss is used between target spectograms and what we got as output for that source. As we mentioned in Table 3.1, there is extra data and it is coming from Deezer's internal datasets. As an optimizer, *ADAM* is used as well.

To use Spleeter directly we can install their python package (which can be used from command-line or python-interpreter) and also they provide the docker image. According to their claims, Spleeter can run 100x faster than real-time when we utilize it via GPU for 4-stems. We will share details about how much time we need when we use via CPU in the upcoming section as well.

3.4.3 Demucs

Demucs is the waveform model for the source-separation algorithm and uses U-net architecture with the bidirectional LSTM. Actually, in their implementation, they also open-sourced ConvTasNet implementation but we will focus on Demucs in this section as it outperforms ConvTasNet.

As a brief overview, Demucs is using the stereo mixture as input and gives outputs for each target's stereo estimate. (C = 2) It is using encoder/decoder architecture which is stacks of the convolutional encoder, bidirectional LSTM and convolutional decoder. Also, we have skip connections between encoder and decoder which is the same as what we have in Spleeter and Open-Unmix. One of the interesting findings that realize is batch normalization can cause significant degradation in the quality of the output. As we saw in the Spleeter, we also have 6 stacked layers in the encoder which are convolutional blocks. On the other hand, the decoder is kind of the inverse of the encoder and has 6 convolutional layers which are doing transposed convolution. One of the important parts of the decoder is the final layer is a linear layer with have *S (number of sources) * C (number of output channels)* channels without any activation function. So for that 4-stems use-case, we have 4 stereo channels as output and each of them corresponds to one of the 4-stems waveforms. Also, thanks to skip connections that connect raw waveform which is the input of encoder and output of the decoder are we can directly transfer phase information. We can see details of implementation in the Figure 3.3. As a loss function, we have somewhat similar approaches with spectogram based models as well. We are using one of the L1 (average mean square error) or L2 (absolute error) loss to represent reconstruction loss of Equation 3.4 which is $L\left(g_s(x;\theta), x_s\right)$ and calculated between predicted $\hat{x}_s$ and target x_s waveform of sources which has T samples and coming from source s. According to experiments, L1 and L2 perform similarly.

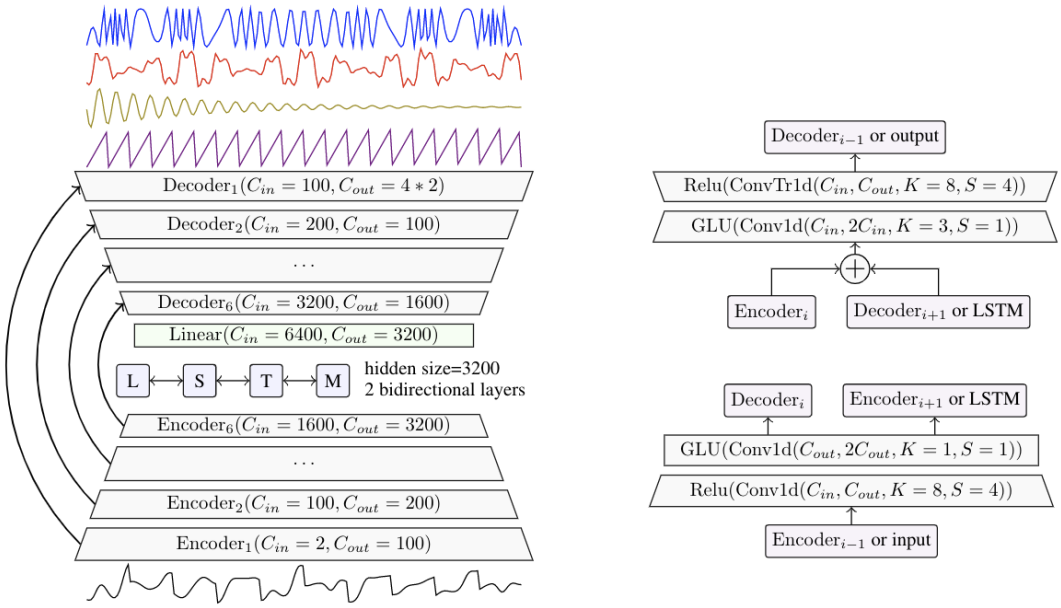
$$L_1\left(\hat{x}_s, x_s\right) = \frac{1}{T}\sum_{t=1}^{T}\left|\hat{x}_{s,t} - x_{s,t}\right| \quad L_2\left(\hat{x}_s, x_s\right) = \frac{1}{T}\sum_{t=1}^{T}\left(\hat{x}_{s,t} - x_{s,t}\right)^2 \tag{3.5}$$

For the training of Demucs, also there are some data augmentation is done. And according to the results, it significantly increased performance. According to human evaluations of results, Demucs outputs sounds natural but especially vocal stem can suffer from bleeding.

We can use the Demucs library directly by installing the python package or it has a docker version. In addition to them, if someone would like to play with it without installing, there is *Google Colab* version of it. In the Demucs module, we have different pre-trained models like Demucs trained on just MUSDB18, Demucs trained with extra data[6], ConvTasNet trained on

[6]According to Demucs' Github page, there are 150 songs as extra data for Demucs and ConvTasNet. On the other hand, Spleeter has 25000 songs.

Figure 3.3. The Architecture of Demucs Model, taken from [7]



just MUSDB18 and ConvTasNet trained with extra data. One of the biggest advantages of the Demucs is that has significant model quantization which can reduce the model size from 1GB to 150MB. So we have a quantized version of it in the Demucs module as well.

3.4.4 Comparison of Those Models in Terms of Memory and Speed

	Runtime (s)	**RAM (GB)**
10 Minute Excerpt from Wagner		
Open-Unmix	17.6	2.64
Spleeter	15.8	3.4
Demucs	25	3.4
Popular Pop Song (158 Seconds)		
Open-Unmix	15	3
Spleeter	11.8	4.54
Demucs	9.18	3.6

Table 3.2. Approximate resource consumption while separating 10 minutes of audio (Karajan B1) and one popular pop song using different pre-trained source separation methods.

As an example, we selected two different pieces. One of them is a 10-minute excerpt from Kar's B1. And second is example is *Stay from Kid Laroi ft Justin Bieber* which is a quite popular pop piece in recent days and the duration of that is 158 seconds. In this section, we will focus on maximum memory usage and spending time for those pieces with all mentioned open-source implementation. We also performed that experiment using single core or available cores which is 72 in our case. We did not see a significant difference in runtime and memory usage is the same.

So we are going to report just single-core performances. As we can see from Table 3.2, Spleeter is fastest when we have longer piece; however, for shorter one, Demucs is fastest. On the other hand, Open-Unmix use least memory for each case.

3.4.5 Comparision of Those Models Qualitative Results

We used the same excerpt as we used in Chapter 2 as *running example* and we can see how spectogram of vocal and accompaniment stems look like after applying each of SS models from Figure 3.4

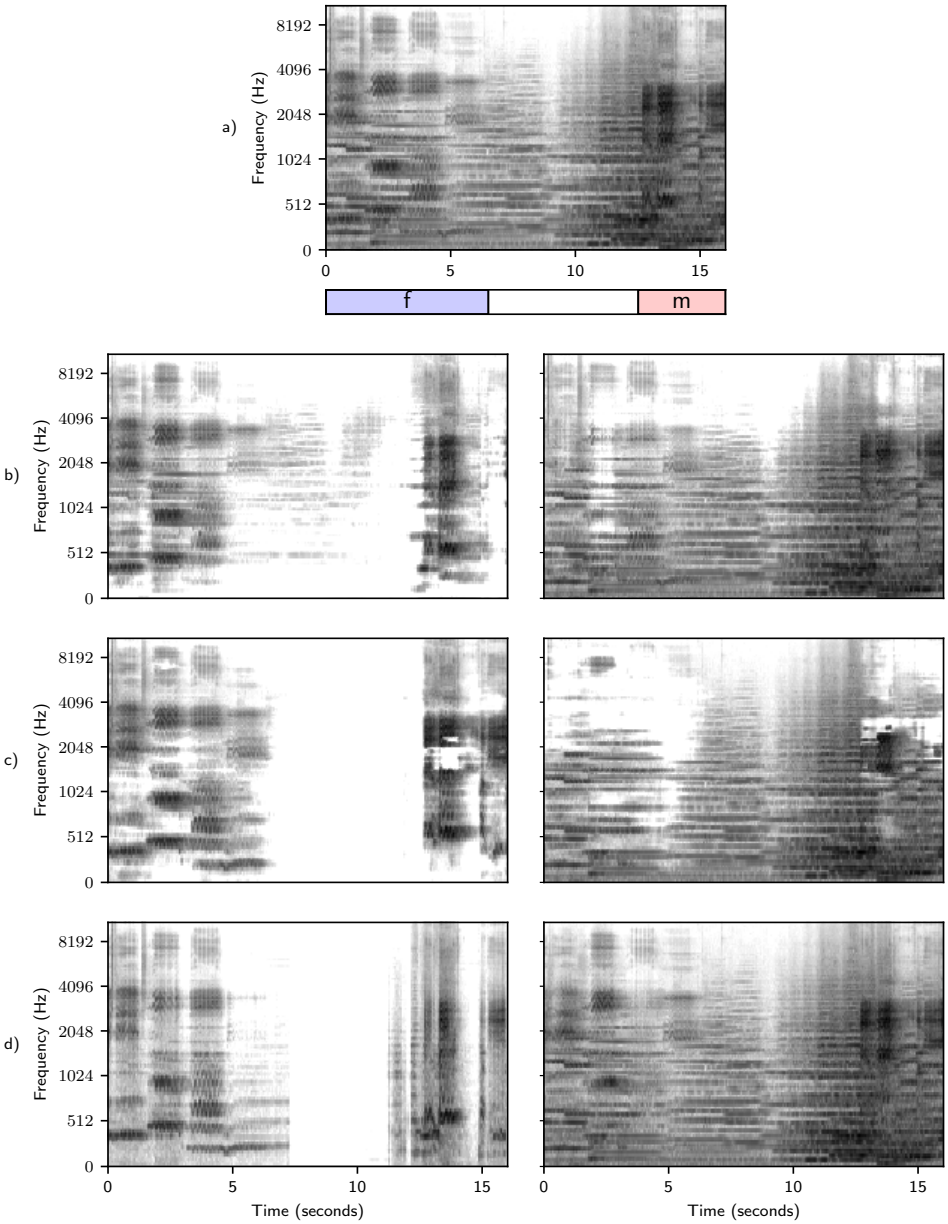


Figure 3.4. Qualitative results of running example, separated into vocal (left) and accompaniment (right) tracks using Open-Unmix (b), Spleeter (c) and Demucs (d), respectively.

Chapter 4

Source Separation for Singing Voice Detection Related Tasks

In this chapter, we will explain how we utilize source separation methods for singing voice-related tasks as Singing Voice Detection (SVD) and gender identification. For all of those tasks, we are using Open-Unmix, Spleeter and Demucs as source-separation-modules. All of them has different pre-trained models in them. We are using the following from them.

Model	Pre-Trained Models
Open-Unmix	Umxl
Spleeter	2-Stems
Demucs	Demucs Quantized

Table 4.1. Selected Pretrained Models

For every experiment, we are focusing on vocal stems and accompaniment stem. For Spleeter's 2-stems, we directly get them but both Open-Unmix and Demucs give more stems but explicitly vocal as well. So we are combining all other stems into one stem and using that as accompaniment stem.

First of all, we will start with Singing Voice Detection. There are 3 different methods and now we will explain their details. On the other hand, we can establish 3 different methods for gender classification.

4.1 Singing Voice Detection

4.1.1 Source Separation's Vocal Stem + RMS + Thresholding *(SS + Thresholding)*

For this setup, we are running our source separation module over mixture and get only vocal stem as output. Then after that, we are running our RMS thresholding to understand silence and non-silence segments. After that, we can label those non-silence segments as vocal and silence segments as non-vocal. Thus, we will have outputs as what one SVD system should have. This approach assumes that with the source separations vocal-stem we should not have any energy in non-vocal stems and we should have some energy in the vocal stems. This would be expected if we have an oracle source separation model as when it isolates vocal stem from the mixture, it should not have anything in the non-vocal segments. So, with this experiment, we can understand that how our source separation methods can be able to keep content/energy for vocal segments and more importantly can it exclude non-vocals (accompaniment in this case) completely which cause no energy from non-vocal segments. We will call this method as *SS + Thresholding* from now on. We can see an overview of that method in the Figure 4.1.

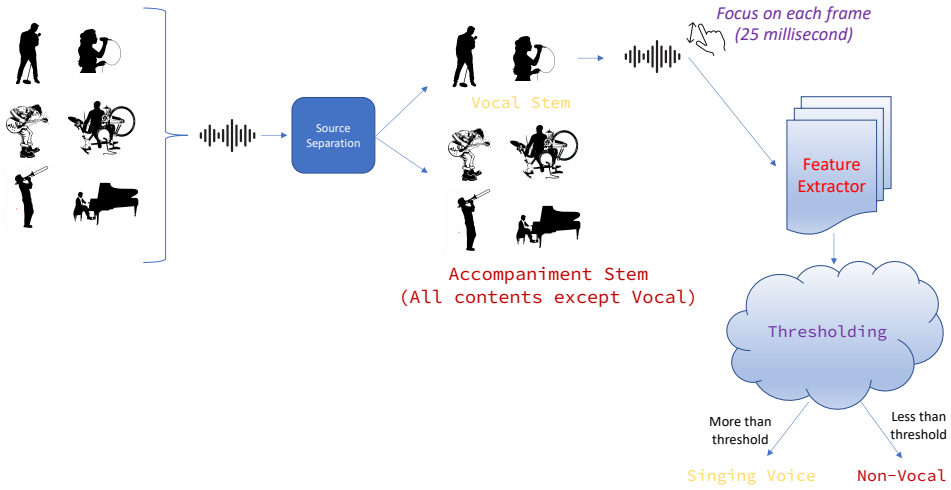


Figure 4.1. Overview of Singing Voice Detection with Thresholding via Source Separation's Output *(SS + Thresholding)*

4.1.2 RFC without Source Separation *(Only RFC)*

We can also use a machine-learning-based classifier like Random Forest Classifier to have a singing voice detection system and we can give original mixture or extracted stems (vocal in our case) via source separation as input of that classifier with some feature representation. For our experiments, we will use MFCC as a feature. In this section, we will explain that how we can

utilize RFC with original mixtures for SVD. As we discussed in our datasets, we can have vocal and non-vocal labels for segments of those pieces. And we will use those labels in the training and then we would expect a classifier will learn to how it can differentiate vocal and non-vocal segments. We will call that method as *Only RFC* from now on. We can see an overview of that method in the Figure 4.2.

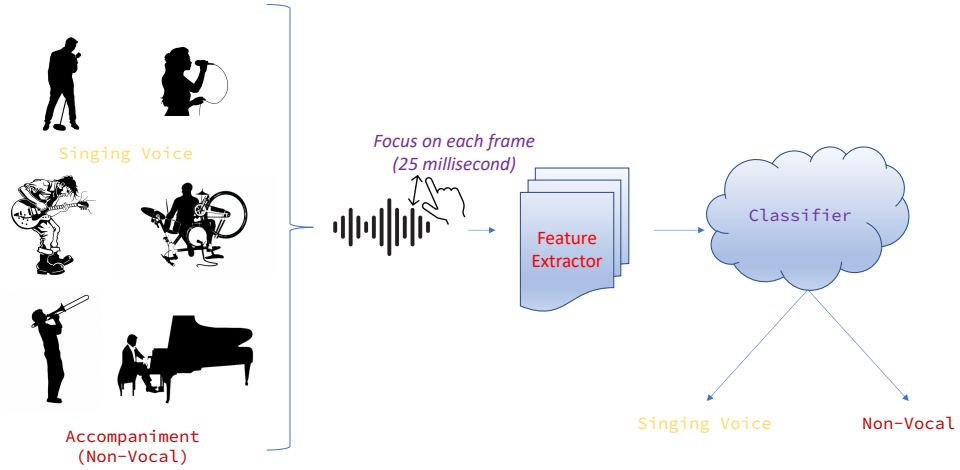


Figure 4.2. Overview of Singing Voice Detection with Random Forest Classification *(Only RFC)*

4.1.3 RFC with Source Separation *(SS + RFC)*

We are not limited to using the original mixture as input of the classifier. So, we can also use MFCC of the vocal stem of the original mixture in the training and testing stage. So basically we will apply source separation models and get vocal stem from that model. Then will get the MFCC of that vocal stem and use that as input of RFC. We will call this method as *SS + RFC* from now on. We can see overview of that method in the Figure 4.3.

4.2 Gender Classification

In gender classification, we can have Female, Male and Non-Vocal labels for our experiments. So, if we have non-vocal segments, we will keep them as non-vocals and also share results related to them. Because it can help to understand how gender classification and singing voice detection is related. Also will be helpful to see the effect of source separation methods' effect over those non-vocals labels.

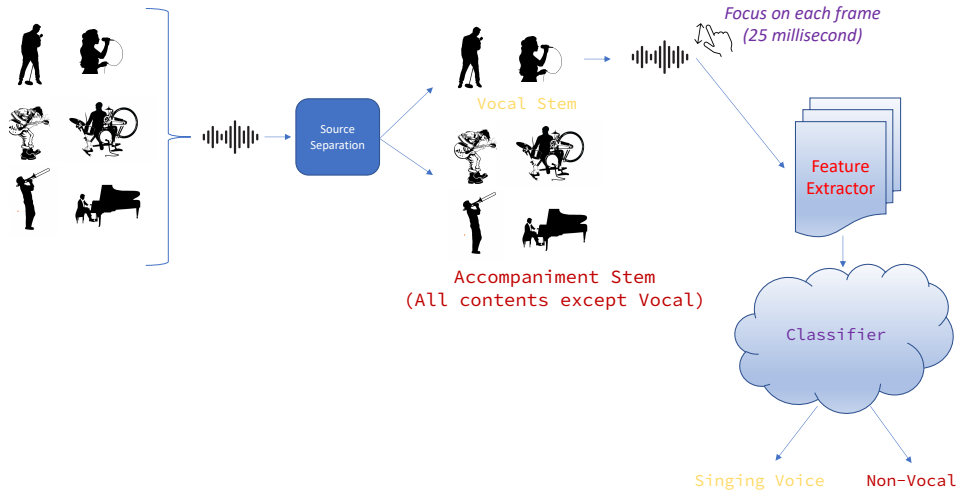


Figure 4.3. Overview of Singing Voice Detection with Random Forest Classification and Source Separation *(SS + RFC)*

4.2.1 RFC without Source Separation *(Only RFC)*

We will start do not applying any source separation and run Random Forest Classifier with male, female and non-vocal labels. We can see overview of that method in the Figure 4.4.

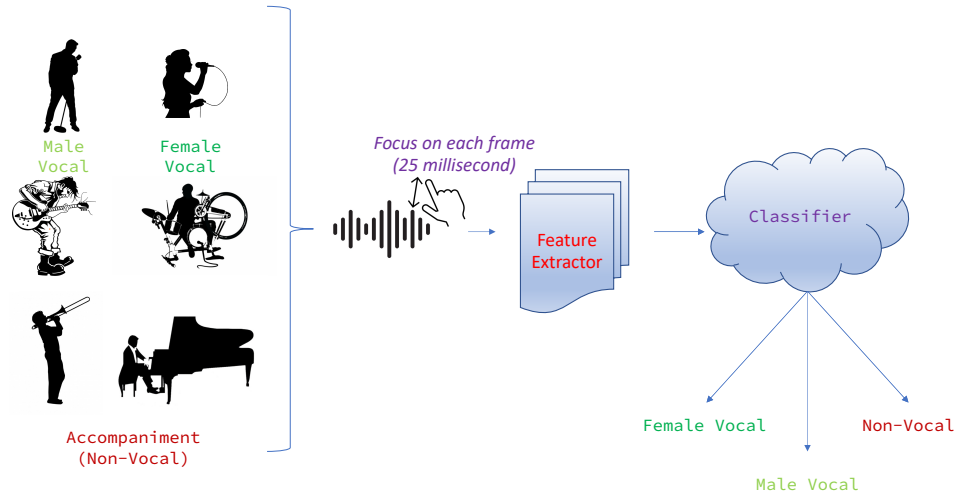


Figure 4.4. Overview of Gender Classification with Random Forest Classification *(Only RFC)*

4.2.2 RFC with Source Separation *(SS + RFC)*

After testing without any source separation, we will apply source separation into our mixture and get vocal stem as output. Then will apply Random Forest Classifier into that stem with male, female and non-vocal labels. If source separation methods are good enough, basically non-vocals

segment becomes silent in that vocal-stem and RFC can easily differentiate it from male and female labels. On the other hand thanks to applying source separation into vocal segments, we can remove accompaniments from those segments and it can help to distinguish male and female vocals from each other as an accompaniment of vocal segments does not help to that differentiation and even worse, it can confuse as well. We can see overview of that method in the Figure 4.5.

4.2.3 RFC with Removing Non-Vocals with and without Source Separation *(SS + RFC)*

As we are testing mainly gender classification, we can try to remove non-vocal labels as we already know ground-truth labels for our training and testing. Then we can just focus on male and female classification. For that, we will have two different experiment sets. First of all, we will not apply any source separation. Then for the second experiment, we will apply source separation and will get vocal-stem of it. Then, will do experiments over it using RFC as our classifier. Actually, in real-life applications, we would not know those labels but we can use the source-separation module's vocal stems and it can help to remove those non-vocals in the ideal case. We can see overview of that method in the Figure 4.5.

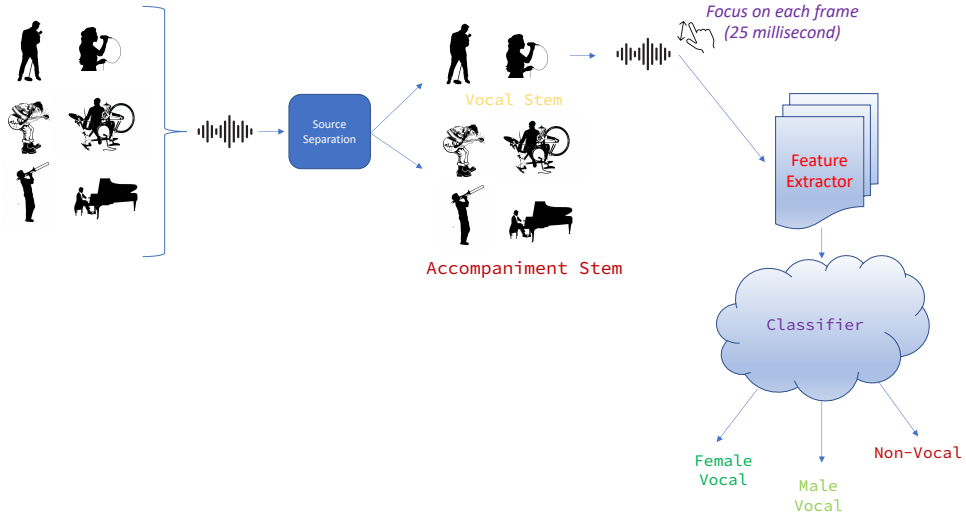


Figure 4.5. Overview of Gender Classification with Random Forest Classification and Source Separation *(SS + RFC)*

Chapter 5

Experiments

5.1 Experiments Using Wagner Opera

For *Wagner Opera* experiments [1], we are using following setup.

- ***Performance Split***: We are using Bar's B1 as the training set and Kar's B1 as a test set.
- ***Act Split***: We are using Kar's B2 as the training set and Kar's B1 as a test set.
- ***Neither Split***: We are using Bar's C3 as the training set and Kar's B1 as a test set.

5.1.1 Singing Voice Detection

For an explanation of how we built all experimental scenarios, we can refer to Chapter 4. In this chapter, we are explicitly going to discuss the outcome of those experiments.

	P	**R**	**F**
SS + Thresholding			
Open-Unmix	0.81	0.88	0.84
Spleeter	0.89	0.97	0.93
Demucs	0.66	0.98	0.79
Only RFC	0.89	0.87	0.88
SS + RFC			
Open-Unmix	0.85	0.90	0.87
Spleeter	0.90	0.94	0.92
Demucs	0.82	0.92	0.87

Table 5.1. Performance Split - SVD results

[1] We explained details about *Wagner Opera* in Section 2.2.

From Table 5.1, we can see that with *Only RFC* our F1-score is 0.88. On the other hand, when we use *SS + Thresholding*, we have F1-scores less or more than that 0.88, and it depends on which SS models we are using. Spleeter outperforms other SS models within *SS + Thresholding*, and also it outperforms using *Only RFC*. So we can say that when we compare *SS + Thresholding* and *Only RFC*, the result depends on the used SS model. In addition to that comparison, to see how those SS models perform when we apply them as a pre-processing step to RFC, we can compare *SS + Thresholding* with *SS + RFC*. With that comparison, we can see that Open-Unmix and Demucs have better scores with RFC, however, we have the same score for Spleeter with both scenarios. On the other hand, if we compare results of *SS + RFC* with *Only RFC*, there are very slight drops in the F1-scores of Open-Unmix and Demucs; however, when we use Spleeter with RFC we can outperform *Only RFC*. The overall picture of Table 5.1 shows that doing thresholding with Spleeter outperforms other SS models in case of *SS + Thresholding* and it also outperforms *Only RFC* and *SS + RFC*. In addition to that, using RFC with Spleeter instead of applying thresholding with Spleeter does not bring an advantage.

As a side note of all of those comparisons, we also explicitly focused on scores from *SS + Thresholding*. With those scores, we can see that the precision of Demucs (0.66) is low, however, recall (0.97) of that is nearly perfect. Thus, we can understand that Demucs can not get rid remove most energies of non-vocal segments. Also, Demucs does not remove the content of vocal segments as well when it is creating vocal stem. Thus, we can conclude that Demucs preserves more energy in the vocal stem than other SS models.

	P	**R**	**F**
SS + Thresholding			
Open-Unmix	0.81	0.88	0.84
Spleeter	0.89	0.97	0.93
Demucs	0.66	0.98	0.79
Only RFC	0.80	0.93	0.86
SS + RFC			
Open-Unmix	0.86	0.91	0.88
Spleeter	0.90	0.95	0.92
Demucs	0.86	0.91	0.88

Table 5.2. Act Split - SVD results

In act split case, we can see similar results with performance split via Table 5.2. The main difference is that when we combine RFC with any of the SS models, it gives a better score than *Only RFC*. That is not the case for performance split. On the other hand, act split is a harder task than performance split according to relation with training and test data because orchestral content is more different in the act split and that makes relation of training and test data more different than performance split case even same singers are singing in the act split. Because the act split scenario is slightly harder, the F1-score for *Only RFC* is now 0.86 compared to 0.88 for the performance split. From *SS + RFC* scores with Table 5.1 and Table 5.2 we can see that we

have 0.92 as F1-score via Spleeter for both act split and performance split. So we can say that SS mitigate that score drop which come from we have harder case and can help to act split to have similar scores with performance split.

	P	R	F
SS + Thresholding			
Open-Unmix	0.81	0.88	0.84
Spleeter	0.89	0.97	0.93
Demucs	0.66	0.98	0.79
Only RFC	0.83	0.88	0.85
SS + RFC			
Open-Unmix	0.87	0.88	0.88
Spleeter	0.89	0.97	0.93
Demucs	0.77	0.94	0.85

Table 5.3. Neither Split - SVD results

In neither split case we can see similar results with act split as we can see via Table 5.3 as *SS + Thresholding* outperforms *Only RFC* with just Spleeter and *SS + RFC* outperforms *SS + Thresholding* when we compare same SS models except Spleeter. The main difference is, when we compare *Only RFC* with *SS + RFC*, Demucs is the only SS model which does not bring any advantage in terms of F1-score; however, every SS model was advantageous in the case of the act split.

When we focus on all of those splits all together with looking at all tables, *SS + Thresholding* with Spleeter always performs better than other scenarios. On the other hand, we do not see significant difference when we check F1-score with different splits when we use *SS + RFC*. For instance, F1-scores of *SS + RFC* with Open-Unmix are 0.87, 0.88 and 0.88 for act split, performance split and neither split respectively. Those scores are 0.92, 0.92 and 0.93 for Spleeter and 0.87, 0.88 and 0.85 for Demucs. That shows that even we have difficult splits, we can still keep good performance thanks to the usage of SS as pre-processing step. As opposed to that, with more difficult splits, we can see a slight drop for *Only RFC* in terms of F1-score. But as we mentioned when we use *SS + RFC*, SS helps mitigate confusion points for our classifier.

We also did some error investigations and will show some examples of common errors in the following. As a side note, for those errors, we are using Spleeter exclusively for SS involved methods and if we do not exclusively say the name of split for RFC involved methods, we can assume that the performance split was used.

- With *SS +Thresholding*, biggest issue is as following
  - *Heroic trumpet and very expressive violin sound* is identified as vocal by *SS + Thresholding*. We can see how SS models behave that segment from the Figure A.1. When we check the result of *Only RFC* with that excerpt, all of segments are classified as

non-vocal. On the other hand with *SS + RFC*, 60% of segments classified as non-vocal. So this means that, with *SS + RFC* or *Only RFC*, we can have better scores than *SS + Thresholding*. On the other hand, *Only RFC* performs better than *SS + RFC*. So this example shows we can have worse result when we apply SS into RFC and *SS + Thresholding* can not be sufficient to separate non-vocal segments from vocal ones.

- With *Only RFC*, biggest issues are as followings
  - *Very expressive and care-free violin sound* is identified as a vocal by *Only RFC* with neither split. We can see how SS models behave that segment from the Figure A.2. When we check experimentally we can see that *SS + Thresholding* is working pretty well and classify 80% of that excerpt's segment as non-vocal. And when we check results with *SS + RFC* with neither split, 67% of segments are classified as non-vocal. So we can see the benefit of SS models directly with that type of sound with both SS involved methods.
  - *Heroic sound with ascending trumpet* is identified as a vocal by *Only RFC* with act split. We can see how SS models behave that segment from the Figure A.3. When we check results via experiments we can see that *SS + Thresholding* works pretty well and 75% of segments of that excerpt are classified as non-vocal. On the other hand, when we check results from *SS + RFC* with act split, 41% of segments are classified as non-vocal. So *SS + Thresholding* can help to mitigate that problem but *SS + RFC* does not work so well. But still, perform better than using *Only RFC* for the aforementioned excerpt.
  - *Quiet and eerie male vocal sound* is identified as non-vocal but ground-truth is vocal. We can see how SS models behave that segment from the Figure A.4. Also when we check the results of *SS + Thresholding* with that excerpt, we can see that 89% of segments are classified as vocal. On the other hand for *SS + RFC*, 83% of segments are classified as vocal. So we can see that with both scenarios SS mitigates that problem.
  - *Darker female sound* is identified as non-vocal but ground-truth is vocal. We can see how SS models behave that segment from the Figure A.5. When we check the result of that excerpt with *SS + Thresholding*, we can see that 32% of segments are classified as vocal. On the other hand, *SS + RFC* classified 86% of segments as vocal. So we can see the significant difference between *SS + Thresholding* and *SS + RFC*. And we can see that *SS + RFC* really worked well for that excerpt but other scenarios can not work that well.
- With *SS + RFC*, biggest issues are as followings
  - *Loud orchestral excerpt with trumpet and violin sound* is non-vocal as ground-truth

but *SS + RFC* classified that as vocal. We can see that SS models fails to remove contents from vocal segments as we can see from the Figure A.6. When we see results with *SS + Thresholding*, we can see that all of the segments are classified as vocal because still we have significant content in the vocal stem as we mentioned. So that is fully failing. On the other hand with just using *Only RFC*, we can see that all of the segments are classified as non-vocal. So if we do not use SS, it would mitigate that problem completely.

– *Female vibrato* is vocal as ground-truth but *SS + RFC* classified that as non-vocal. We can see how SS models behave that segment from the Figure A.7. When we check results with experiments, all segments are classified as non-vocal with *SS + Thresholding*. So it is always failing. On the other hand, with using *Only RFC*, 98% of segments are classified as vocal. That does mitigate the problem fully and not using SS is beneficial for this excerpt.

5.1.2 Gender Classification

In this section, we are focusing on Gender Classification experimentations with the Wagner Dataset and we will two different scenarios as we explained in Chapter 4.

5.1.2.1 Keeping Non-Vocals

With keeping non-vocals, we are not changing anything in the data. We have two different methods for gender classification as using *Only RFC* or using RFC with source separation (*SS + RFC*). We perform this experiment for three different splits as we performed for SVD. In the following tables, we are presenting evaluation measures for vocal, female, and male labels. Our classification labels are non-vocal, female, and male but we combined female and male's scores to get scores for vocal. So basically that is the weighted combination of scores for female and male labels.

	Vocal			**Female**			**Male**		
	P	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**
Only RFC	0.9	0.80	0.84	0.78	0.77	0.77	0.95	0.82	0.88
SS + RFC									
Open-Unmix	0.85	0.83	0.84	0.75	0.85	0.80	0.90	0.82	0.86
Spleeter	0.89	0.89	0.89	0.81	0.88	0.84	0.92	0.90	0.91
Demucs	0.82	0.86	0.84	0.76	0.83	0.79	0.85	0.87	0.86

Table 5.4. Performance Split - Gender recognition results on opera dataset.

When we perform performance split and check results from Table 5.4, we can realize that when we use *SS + RFC*, we have significant improvement with Spleeter but Open-Unmix and Demucs

perform similarly with *Only RFC* scenario. On the other hand, when we check how results are changing by gender, when we use Demucs or Open-Unmix, we have better scores for female labels but the score for male labels is lower. The possible reason for that, when we do not use SS, we have confusion between non-vocals and female labels especially. But, when we apply SS, this discrimination becomes easier. On the other hand, those SS models remove some content that is important for male and female discrimination.

	Vocal			**Female**			**Male**		
	P	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**
Only RFC	0.83	0.86	0.84	0.69	0.83	0.75	0.89	0.88	0.89
SS + RFC									
Open-Unmix	0.85	0.84	0.84	0.81	0.83	0.82	0.87	0.84	0.86
Spleeter	0.88	0.90	0.89	0.83	0.86	0.84	0.91	0.93	0.92
Demucs	0.85	0.84	0.84	0.89	0.84	0.86	0.89	0.84	0.86

Table 5.5. Act Split - Gender recognition results on opera dataset.

When we perform act split and check results from Table 5.5, we can realize that when we apply SS into RFC (*SS + RFC*), we have significant improvement with every SS model but Spleeter again outperforms all others. We can see that we have some performance drop for male discrimination with Open-Unmix and Demucs. A possible reason for this, is that with those SS models some contents which are important to discriminate male than female or non-vocal are also removed.

	Vocal			**Female**			**Male**		
	P	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**
Only RFC	0.80	0.76	0.77	0.56	0.78	0.65	0.92	0.75	0.83
SS + RFC									
Open-Unmix	0.85	0.8	0.82	0.70	0.85	0.77	0.92	0.76	0.83
Spleeter	0.86	0.9	0.88	0.74	0.90	0.82	0.91	0.90	0.91
Demucs	0.77	0.85	0.81	0.69	0.82	0.75	0.81	0.87	0.84

Table 5.6. Neither Split - Gender recognition results on opera dataset.

When we perform neither split and check results from Table 5.6, we realize that when we apply SS into RFC(*SS + RFC*), there is significant improvement with all of the SS models. On the other hand, Spleeter outperformed other SS models. When we compare all of those splits with *Only RFC*, neither split is performed way worse than others. But when we apply SS models, we can get similar results with all of the splits. So this means that by using SS models, we can get high scores even with not similar training data. So this shows that using SS for gender classification is helpful to get better results.

5.1.2.2 Removing Non-Vocals

With removing non-vocals we are changing data and details can be found in Chapter 4.

	Female			Male		
	P	**R**	**F**	**P**	**R**	**F**
Only RFC	0.83	0.97	0.89	0.98	0.91	0.94
SS + RFC						
Open-Unmix	0.86	0.92	0.89	0.96	0.93	0.95
Spleeter	0.90	0.95	0.93	0.98	0.95	0.96
Demucs	0.89	0.91	0.90	0.96	0.95	0.95

Table 5.7. Performance Split - Gender recognition results on opera dataset when ignoring all non-vocal frames during training and testing.

We can see the results of the performance split from Table 5.7 and we can realize that when we use any of SS model with *SS + RFC* we can get a better score than *Only RFC*. A possible reason for that improvement can come from the fact that we are removing accompaniments from vocal segments and this helps to get rid of some contents which can confuse classifiers. Also, when we closely look at each of the SS models, it is obvious to see that Spleeter outperforms everything as well.

	Female			Male		
	P	**R**	**F**	**P**	**R**	**F**
Only RFC	0.88	0.94	0.91	0.97	0.94	0.96
SS + RFC						
Open-Unmix	0.89	0.90	0.90	0.95	0.95	0.95
Spleeter	0.92	0.93	0.92	0.97	0.97	0.96
Demucs	0.87	0.94	0.90	0.97	0.93	0.95

Table 5.8. Act Split - Gender recognition results on opera dataset when ignoring all non-vocal frames during training and testing.

We can see results of performance split from Table 5.8 and we can realize that all of the SS models do not outperform *Only RFC* scenario. A possible reason for them can be removing accompaniment from vocal segments can not help to discriminate between male and female. This means that RFC can learn to discriminate between males and females without removing accompaniment in vocal segments via SS models. Also in some cases, it can confuse more and cause slightly lower scores. Generally, as we have had high scores with using *Only RFC*, it is hard to improve results more with SS models.

We can see results of neither split from Table 5.9 and we can realize that all of the SS models outperform *Only RFC* scenario. If we compare with other splits, we can see that the F1-score of the female is increased much with Spleeter. A possible reason for that, with neither split, is that it is harder to learn discriminative features than other splits but when we remove accompaniments from vocal segments, it becomes easier to learn those discriminative features.

	Female			Male		
	P	**R**	**F**	**P**	**R**	**F**
Only RFC	0.71	0.96	0.82	0.98	0.82	0.89
SS + RFC						
Open-Unmix	0.77	0.95	0.85	0.97	0.87	0.92
Spleeter	0.98	0.91	0.95	0.84	0.97	0.90
Demucs	0.88	0.89	0.88	0.95	0.94	0.94

Table 5.9. Neither Split - Gender recognition results on opera dataset when ignoring all non-vocal frames during training and testing.

5.2 Experiments Using Smule

We are not applying any SS model for Smule experiments. The main reason for that, we have those data from a karaoke application and most of the data have just clean vocal sound. So that dataset can be considered as the scenario of *what happens if we have a perfect SS model to get clean-undisturbed vocal stem* and in the future works, some can mash-up those vocals with accompaniments and try to utilize SS models to achieve similar results with our experiments and which can somewhat show SS models give clean vocal records.

5.2.1 Selected Subset

For the Smule dataset, as we mentioned in Section 2.2, there are 24874 *unaccompanied* solo singing recordings from 5429 singers singing a collection of 14 songs. For our experiment setup, that is so excessive. Thus, we decided to select a subset to focus on. Also, we are going to perform experiments over *singer split* which makes more sense if we compare with *song split*. Because when we have singer split, we would not have same singers in training and test set. So, the classifier can not cheat with that information and it makes the task more challenging and realistic. For the singer split setup, we decided to use the same song for both the training and test set. Based on the counts of each song in the dataset, we decided to select "All of Me" by John Legend as song example. One can select different pieces but we decided to use that as we need to do it. Also dataset has *country* labels in it. To get rid of the effect of *accent*, we just limited our selected subset with country labels as *GB (Great Britain)* and *US (United States)* which are countries with native English speakers. As we mentioned before, there are 2856 recordings for "All of Me" by John Legend and 840 of them are coming with US or GB labels.

For that selected song, there are 840 total recordings in the dataset with 484 of them labelled as female and 356 of them labelled as male. That is still more than what we need as we would like somewhat have a similar total duration with Wagner Experiments. So we decided to have 5% of each gender for the train set and the other 5% for the test set. So before dataset cleaning, we have 24 female and 17 male recordings in the train set, on the other hand, 24 female and 18 male

recordings in the test set. But we are going to clean that dataset in the following section further.

5.2.2 Dataset Cleaning

As those pieces are coming from the karaoke application, we have silences between singing segments. So we would like to remove those silences to make our classifier as robust as this can just focus on singing segments in both the training and testing stage. On the other hand, we realized that there is some ground-truth label error in the dataset. We also removed that to get rid of incorrect learning in training and not-correct measurement in the evaluation stage. In addition to that, we also removed some low-quality recordings which have no clear vocal sound and most of those pieces have accompaniment in recording after RMSE suppression as well. In the following sections, we can find detailed information about how we performed those tasks.

5.2.2.1 RMSE Thresholding for Silence Removal

For the calculation of Root-Mean-Square (RMS), we are using Librosa. Librosa can calculate RMS via audio samples or spectogram values. We are going to use audio samples as it gives faster calculation because we do not need to STFT for that. For RMS calculation via audio signals, basically we are getting the mean of the square of the absolute value of the segment's amplitude values. Then we are getting the square root of that value. On the other hand, *hop-length* and *frame-length* are other parameters for calculation. We are using 512 samples (23 milliseconds) as *hop-length* and 2048 frames (92 milliseconds) as *frame-length*. As a threshold, we are using 0.001 and we decided that by listening experiments. For instance, we were wearing headphones and set the volume highest possible. Then if the energy of the segment is less than 0.001, it is inaudible.

5.2.2.2 Training and Removing Wrong Ground-Truth Labels and Low-Quality Recording for Subset

As we selected a subset of the Smule dataset, we performed manual investigation over those pieces

- To change their ground-truth label if that is not correct.
- To remove piece if it has two singers which are female and male in it.
- To remove if the piece has low-quality vocal mostly because of quality of the microphone and also has some accompaniment even after RMSE thresholding.

With that cleaning, we removed 3 pieces from a training set and 4 pieces from the test set. For one piece in the test set, we changed the ground-truth label. So after cleaning, we have 23 female

and 15 male recordings in the train set, on the other hand, 22 female and 16 male recordings in the test set.

5.2.3 Gender Classification

	Female			Male		
	P	**R**	**F**	**P**	**R**	**F**
Frame-Level	0.75	0.82	0.78	0.72	0.63	0.67
song-level						
w/out CLF	0.80	0.89	0.85	0.83	0.71	0.77
with CLF	0.91	0.99	0.95	0.92	0.61	0.73

Table 5.10. Gender Recognition Results over Selected Smule Subset with frame-level, song-level with and without CLF (Confidence-Level-Filtering)

5.2.3.1 Frame Level

For frame-level experiments, we can see results from Table 5.10. As we can see from the table, recall for males is pretty lower than other results. So this means that some male label segments are predicted as female. One of the possible reasons can be classifier tends to predict the high-pitch part of songs as female. So when a male signer sings those parts, those can be predicted as female.

5.2.3.2 Song Level

As hypothetically each piece is sung by just one singer, we can have song-level labels. For song-level experiments, the classifier still predicts frame-wise but after getting all of those predictions for a piece, we are applying the majority-vote principle and decide predicted gender for the song. We can see results in Table 5.10 So apparently, we have better results than frame-level. We also have the results with CLF (Confidence-Level-Filtering) which is 90% in our experiment. To apply CLF, we need to know the confidence level of each prediction of our classifier. With scikit-learn, we can have confidence information for each prediction. With that value, we can understand how much the classifier can discriminate one label from another. So, if it is 100%, it means that the classifier has full confidence for that prediction. If that value is around 50% which means that classifier has some trouble with discriminating labels from each other. Thus, as we are applying the majority-vote principle, we can just use selected predictions which has more than 90% confidence level. This means that, if the confidence level of the prediction of that frame is more than 0.9% we are using that frame in majority-vote, otherwise we are discarding that frame from song-level prediction. We can see results in Table 5.10 So apparently, we have

better results with CLF if we compare with not applying that. So it shows that, if we just focus frames which classifier can discriminate significantly one label than another one, we can have better results as we are removing frames that confuse our classifier. Also, we can say that there is still room to make our classifier better by training more data which can help to discriminate better. On the other hand, we realize that the recall value for the male class become lower when we apply that filtering. Thus, we acknowledge that classifier's female predictions are mostly more exact than male ones.

Chapter 6

Conclusions

In this thesis, we introduced how we can utilize source separation (SS) to perform better with singing voice related tasks such as Singer Voice Detection (SVD) and Gender Classification. As SS models, we decided to use well-known open-source models which are Demucs, Spleeter and Open-Unmix.

First of all, we introduced our methods to perform SVD. We decided to use *SS + Thresholding*, which is applying RMSE thresholding into the vocal stems of SS models, *only RFC*, which use RFC as classifier, and *SS + RFC*, which has SS as a pre-processing step before that RFC. We can see those methods from Figure 4.1, Figure 4.2 and Figure 4.3 respectively. For SVD, we exclusively focus on Wagner's Ring Opera which is a complex opera piece with many singers and many instruments. With that complex piece, we can test how those models perform with different cases. Also, we performed our experiments with performance-split, act-split and neither-split. For every case, we realized that when we apply *SS + Thresholding* with Spleeter, we can outperform every other method. A possible reason for that might be that Spleeter works pretty well to remove all content from non-vocal segments when compared with other SS models, causing those segments to become completely silent. So with *SS + Thresholding*, we would have a better score; however, also recall score is better than precision, which means that our failures have mostly come from non-vocal segments that have more energy than the threshold. Still, Spleeter's precision is better than all other SS models. On the other hand, when we check performance split, *SS + RFC* performs better than *only RFC* just with Spleeter. This can also show that Spleeter is better than others in terms of pre-processing steps to get a better vocal stem. On the other hand, for act-split and neither-split, all of SS models with *SS + RFC* gives a better score than only RFC. This means that, when we have a harder case, SS can help to mitigate some confusion from our classifier and while still performing well.

Also, we did some error investigation and realized that SS can be really helpful when we have *very expressive and carefree violin sound* and SS can remove most of their energy. On the other

hand, also SS is helpful for *quiet and eerie male vocal* and *darker female*. So for most of the time using SS as a pre-processing step for RFC is beneficial. Also, without setting any RFC and using *SS + Thresholding* will give higher scores as well. On the other hand, sometimes applying SS can cause a lower score than using *only RFC* and examples for that can be *heroic trumpet and very expressive violin sound* and *female vibrato*. As a side note, when we compare Open-Unmix and Demucs, we can see that Demucs never performed better than Open-Unmix. There is no clear answer for why this is the case, so it remains a further research question.

As a next step, we focused on Gender Classification with Wagner's Ring Opera and the Smule dataset, which is a karaoke application with recordings from non-professional singers. For Gender Classification, we are using *only RFC* and *SS + RFC* and we can see those methods from Figure 4.4 and Figure 4.5 respectively. Let's focus on Wagner's Ring first. We performed Gender Classification with two different cases. In the first case, we did not touch dataset and experiments with performance-split, act-split and neither-split. According to results, *SS + RFC* performs better than *only RFC*. Especially for neither-split, which is the hardest split, we can see the biggest difference between *SS + RFC* and *only RFC*. If we compare SS models, Spleeter outperforms Demucs and Open-Unmix. In the second case, we remove non-vocal segments by looking at ground-truth data. So the male and female comparisons become easier as we removed non-vocal labels completely. For every splits, *SS + RFC* performs better than *only RFC*. And again for the hardest split, we can see the biggest difference and the best SS model is Spleeter according to evaluation measures.

Then we focused on the Smule dataset for Gender Classification. First of all, we selected some pieces based on some criteria, as that is a huge dataset, and we did some cleaning to remove silences and wrong ground-truth pieces. We performed frame-level and song-level classification.For song-level classification, we applied Confidence Level Filtering (CLF). We saw that song-level performed better than frame-level. Within frame-level scores, we can benefit from CLF. This means that, when we remove segments that confuse our classifier and then get a majority vote of frame-level predictions, we can have a higher score than other methods.

If someone would like to continue with what we presented, several things can be performed. For instance, with Wagner's Ring Opera, it is possible to use coarse-annotated data as an additional training set, and evaluate the benefits. Also, re-training the open-source SS models, which are trained with mostly pop songs, with our dataset of different opera pieces, and evaluating the results of that. On the other hand, one can work on Smule Dataset to make a mashup by adding some accompaniments and work on them with SS models to see which one can perform better for those non-professional vocal recordings. Comparing that with the results of Wagner's Ring Opera can explain which SS models are more appropriate for different cases. Then one can also perform cross-dataset experiments to see how much we can learn from Wagner's Ring Opera to perform SVD and Gender Classification over Smule-mashup and vice-versa. Also, it is possible to

apply some monophonic pitch tracking algorithms to see how they perform especially for Gender Classification with and without SS. Because we think that when we apply the SS model and get vocal stems as we did for our methods, one can apply monophonic-pitch tracking to that stem and give that feature to a classifier like RFC to see if we have better scores for Gender Classification since pitches for male and female are mostly different from each other. Also as a general point, some other deep-learning-based classifiers can be tested to compare with RFC.

Appendix A

More about Longest Errors

On the following pages, we will show some additional examples of errors that are coming from SVD experimentation with Wagner Opera. For detailed explanation about them, we can refer Section 5.1. Also, we are providing corresponding *audio* files if you send an e-mail to us.

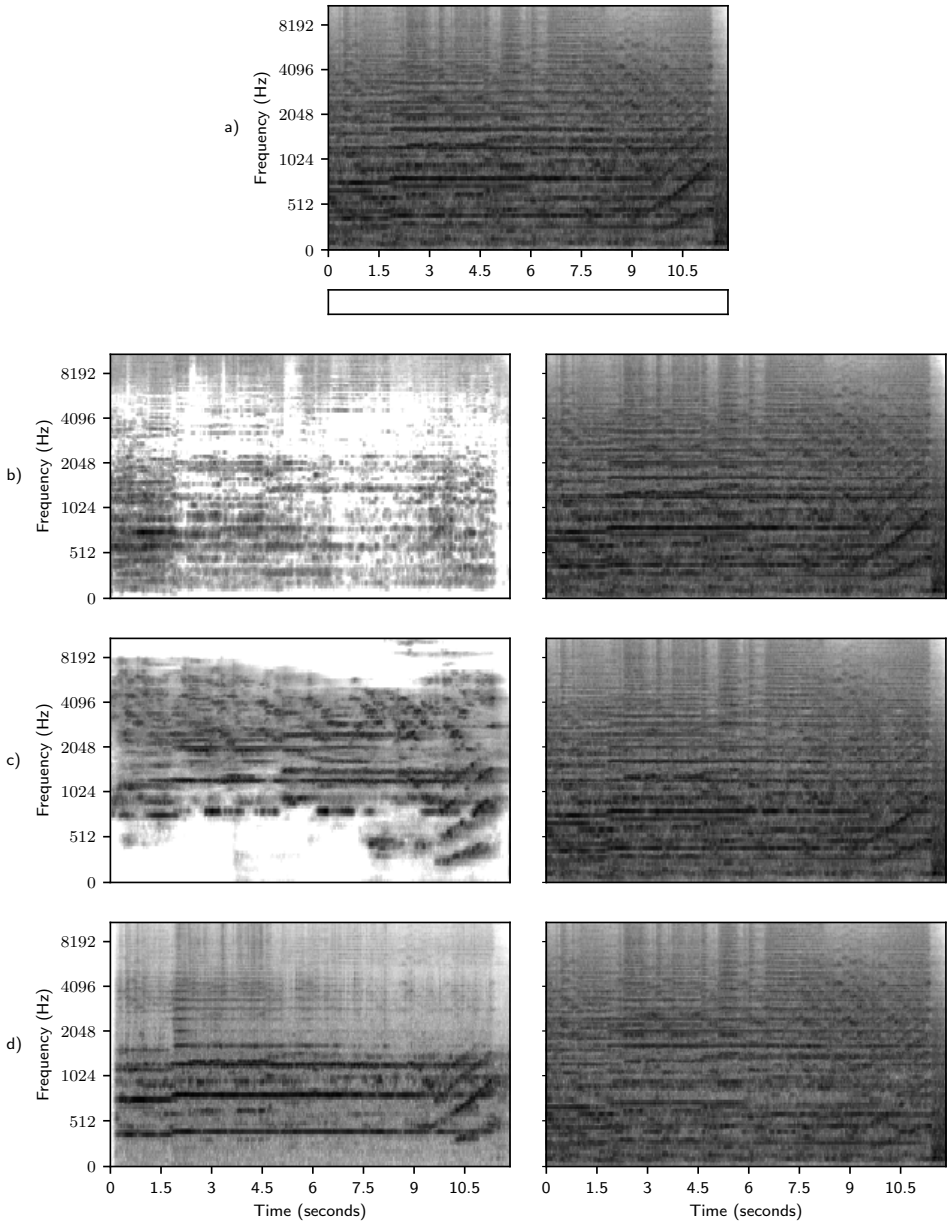


Figure A.1. Qualitative results for an excerpt which is called as *heroic trumpet and very expressive violin sound*, separated into vocal (left) and accompaniment (right) tracks using Open-Unmix (b), Spleeter (c) and Demucs (d), respectively. This is Karajan B1, 00:01:41-00:01:52.

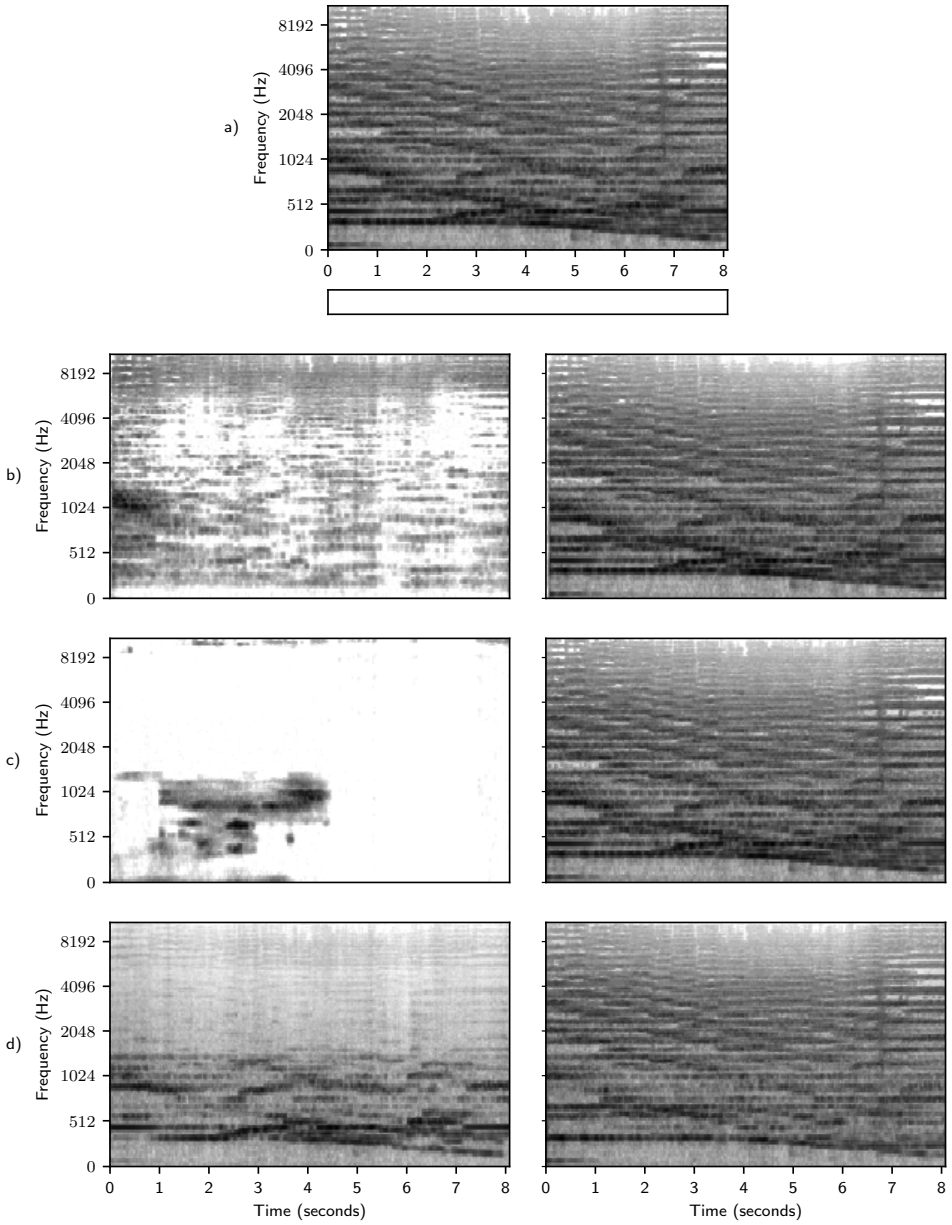


Figure A.2. Qualitative results for an excerpt which is called as *very expressive and carefree violin sound*, separated into vocal (left) and accompaniment (right) tracks using Open-Unmix (b), Spleeter (c) and Demucs (d), respectively. This is Karajan B1, 00:09:48-00:09:56.

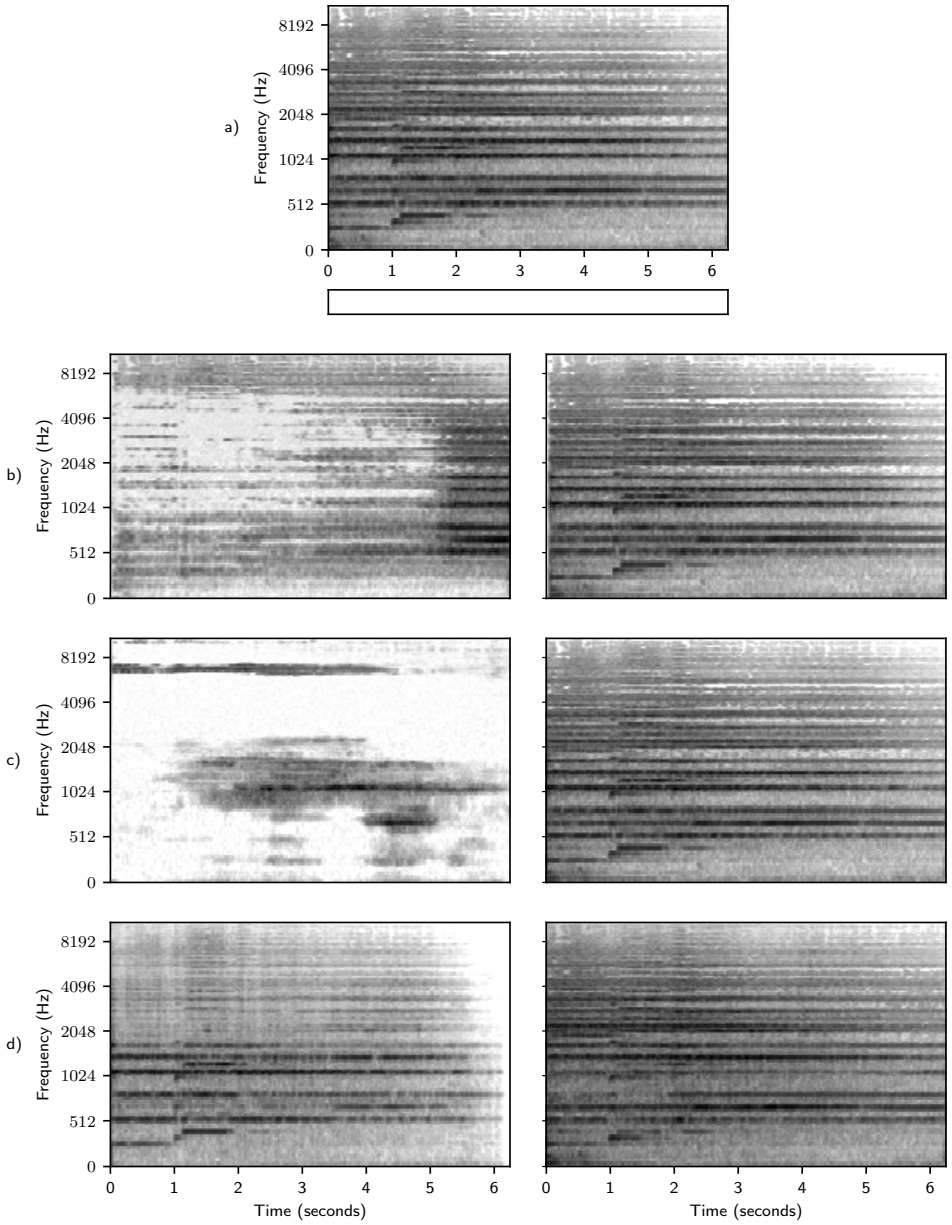


Figure A.3. Qualitative results for an excerpt which is called as *heroic sound with ascending trumpet*, separated into vocal (left) and accompaniment (right) tracks using Open-Unmix (b), Spleeter (c) and Demucs (d), respectively. This is Karajan B1, 00:41:00-00:41:07.

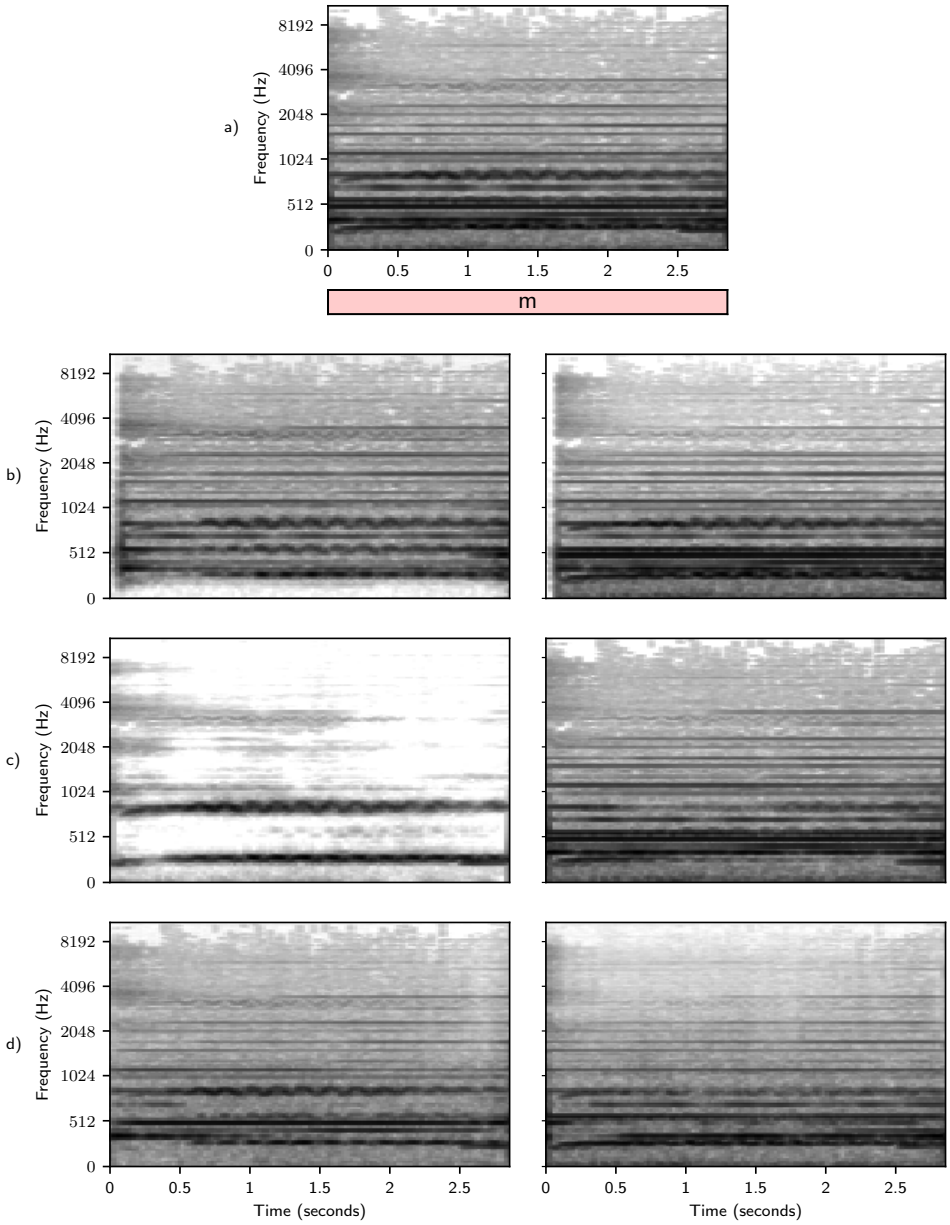


Figure A.4. Qualitative results for an excerpt which is called as *quiet and eerie male vocal sound*, separated into vocal (left) and accompaniment (right) tracks using Open-Unmix (b), Spleeter (c) and Demucs (d), respectively. This is Karajan B1, 00:10:34-00:10:37.

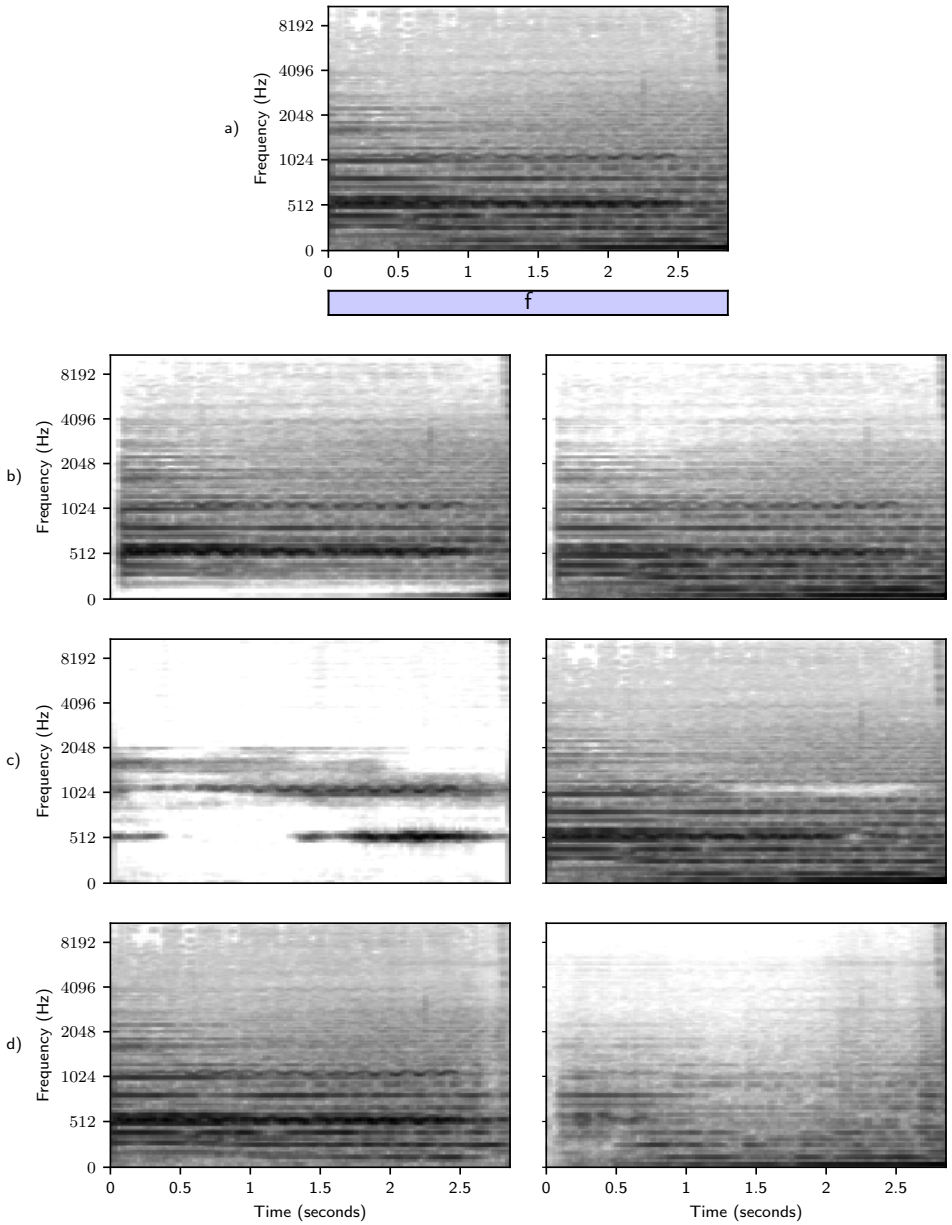


Figure A.5. Qualitative results for an excerpt which is called as *darker female sounds*, separated into vocal (left) and accompaniment (right) tracks using Open-Unmix (b), Spleeter (c) and Demucs (d), respectively. This is Karajan B1, 00:13:41-00:13:44.

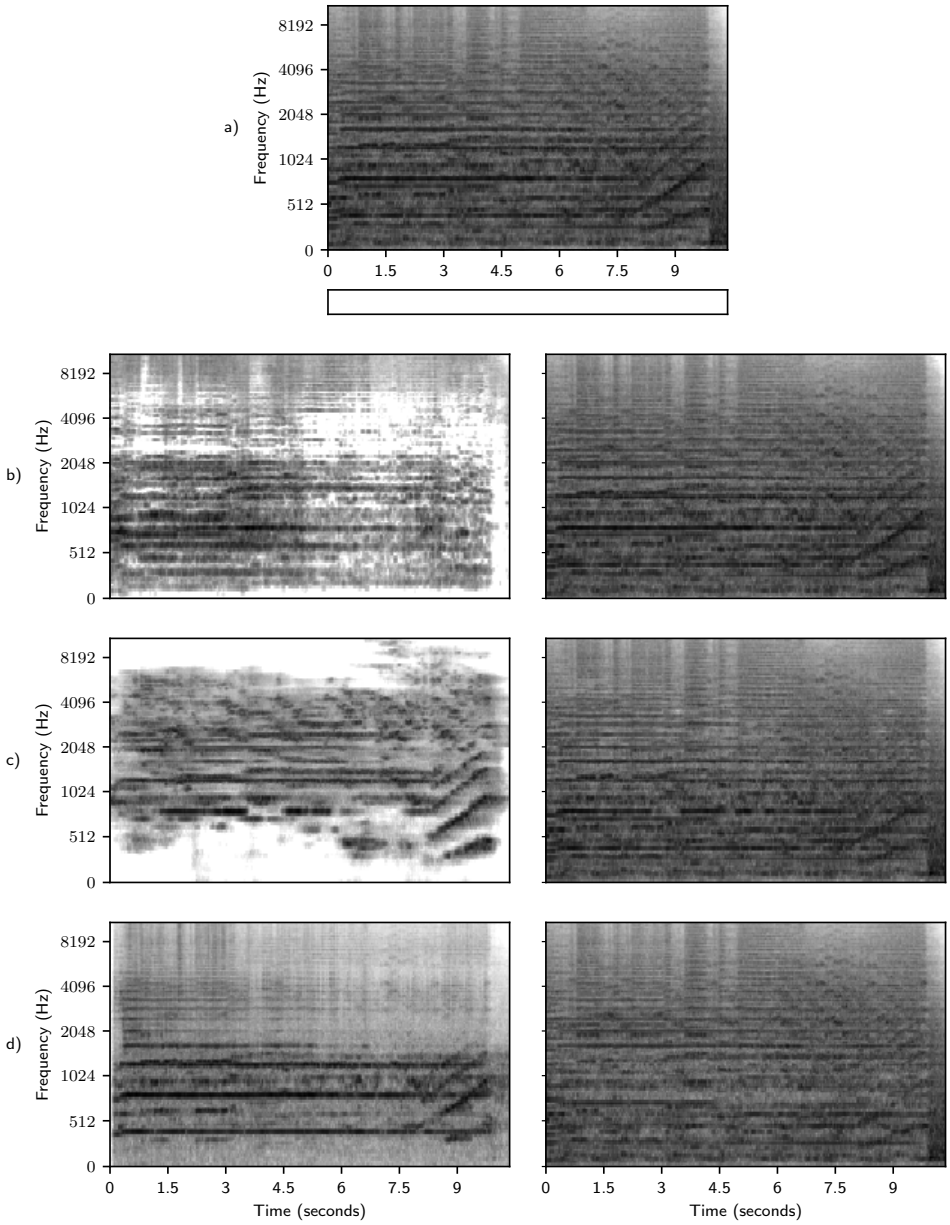


Figure A.6. Qualitative results for an excerpt which is called as *loud orchestral excerpt with trumpet and violin sound*, separated into vocal (left) and accompaniment (right) tracks using Open-Unmix (b), Spleeter (c) and Demucs (d), respectively. This is Karajan B1, 00:01:42-00:01:52.

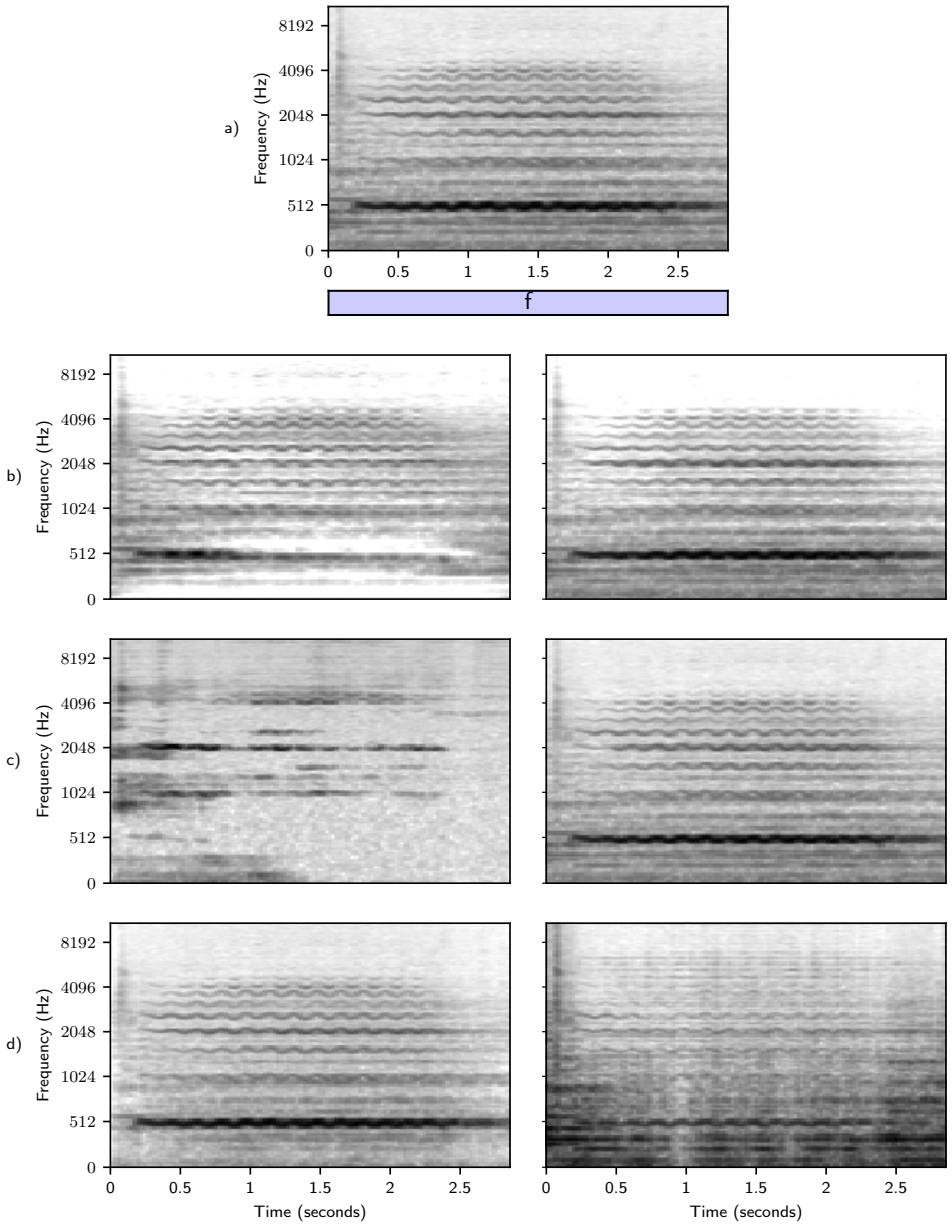


Figure A.7. Qualitative results for an excerpt which is called as *female vibrota*, separated into vocal (left) and accompaniment (right) tracks using Open-Unmix (b), Spleeter (c) and Demucs (d), respectively. This is Karajan B1, 01:00:15-01:00:18.

Bibliography

[1] L. Breiman, *Random forests*, Machine Learning, 45 (2001), pp. 5–32.

[2] P. Burbidge and R. Sutton, *The Wagner companion*, Cambridge University Press, 1979.

[3] V. Chandrasekhar, M. E. Sargin, and D. A. Ross, *Automatic language identification in music videos with low level audio and visual features*, in Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2011, May 22-27, 2011, Prague Congress Center, Prague, Czech Republic, IEEE, 2011, pp. 5724–5727.

[4] H. Chin, J. Kim, Y. Kim, J. Shin, and M. Y. Yi, *Explicit content detection in music lyrics using machine learning*, in 2018 IEEE International Conference on Big Data and Smart Computing, BigComp 2018, Shanghai, China, January 15-17, 2018, IEEE Computer Society, 2018, pp. 517–521.

[5] Y. M. G. Costa, L. Oliveira, and C. N. Silla, *An evaluation of convolutional neural networks for music classification using spectrograms*, Appl. Soft Comput., 52 (2017), pp. 28–38.

[6] R. B. Dannenberg and N. Hu, *Polyphonic audio matching for score following and intelligent audio editors*, in Proceedings of the International Computer Music Conference (ICMC), San Francisco, USA, 2003, pp. 27–34.

[7] A. Défossez, N. Usunier, L. Bottou, and F. Bach, *Music source separation in the waveform domain*, arXiv preprint arXiv:1911.13254, (2019).

[8] R. Delbouys, R. Hennequin, F. Piccoli, J. Royo-Letelier, and M. Moussallam, *Music mood detection based on audio and lyrics with deep neural net*, in Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018, E. Gómez, X. Hu, E. Humphrey, and E. Benetos, eds., 2018, pp. 370–375.

[9] A. M. Demetriou, A. Jansson, A. Kumar, and R. M. Bittner, *Vocals in music matter: the relevance of vocals in the minds of listeners*, in ISMIR, 2018.

[10] C. Dittmar, B. Lehner, T. Prätzlich, M. Müller, and G. Widmer, *Cross-version singing voice detection in classical opera recordings*, in Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), Málaga, Spain, October 2015, pp. 618–624.

[11] A. Faupel and V. Schmutz, *From fallen women to madonnas: Changing gender stereotypes in popular music critical discourse*, Sociologie de l'Art, OPuS 18 (2011), p. 15.

[12] H. M. Fayek", *"speech processing for machine learning: Filter banks, mel-frequency cepstral coefficients (mfccs) and what's in-between"*, "2016".

[13] J. T. Foote, *Content-based retrieval of music and audio*, in Multimedia Storage and Archiving Systems II, C.-C. J. Kuo, S.-F. Chang, and V. N. Gudivada, eds., SPIE, Oct. 1997.

[14] A. Geron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow : concepts, tools, and techniques to build intelligent systems*, O'Reilly Media, Inc, Sebastopol, CA, 2019.

[15] R. Hennequin, A. Khlif, F. Voituret, and M. Moussallam, *Spleeter: a fast and efficient music source separation tool with pre-trained models*, Journal of Open Source Software, 5 (2020), p. 2154. Deezer Research.

[16] Y. Hu, X. Chen, and D. Yang, *Lyric-based song emotion detection with affective lexicon and fuzzy clustering method*, in Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009, Kobe International Conference Center, Kobe, Japan, October 26-30, 2009, K. Hirata, G. Tzanetakis, and K. Yoshii, eds., International Society for Music Information Retrieval, 2009, pp. 123–128.

[17] C. i Wang and G. Tzanetakis, *Singing style investigation by residual siamese convolutional neural networks*, in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, Apr. 2018.

[18] N. Jacoby, E. A. Undurraga, M. J. McPherson, J. Valdés, T. Ossandón, and J. H. McDermott, *Universal and non-universal features of musical pitch perception revealed by singing*, Current Biology, 29 (2019), pp. 3229–3243.e12.

[19] A. Jansson, E. J. Humphrey, N. Montecchio, R. M. Bittner, A. Kumar, and T. Weyde, *Singing voice separation with deep U-net convolutional networks*, in Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), Suzhou, China, 2017, pp. 745–751.

[20] Y. E. Kim and B. Whitman, *Singer identification in popular music recordings using voice coding features*, in Proceedings of the 3rd international conference on music information retrieval, vol. 13, Citeseer, 2002, p. 17.

[21] S. Kum and J. Nam, *Joint detection and classification of singing voice melody using convolutional recurrent neural networks*, Applied Sciences, 9 (2019).

[22] K. Lee, K. Choi, and J. Nam, *Revisiting singing voice detection: a quantitative review and the future outlook*, arXiv:1806.01180 [cs, eess], (2018). arXiv: 1806.01180.

[23] S. Leglaive, R. Hennequin, and R. Badeau, *Singing voice detection with deep recurrent neural networks*, in Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Brisbane, Australia, 2015, pp. 121–125.

[24] B. Lehner, R. Sonnleitner, and G. Widmer, *Towards light-weight, real-time-capable singing voice detection*, in Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), Curitiba, Brazil, 2013, pp. 53–58.

[25] B. Lehner, G. Widmer, and R. Sonnleitner, *On the reduction of false positives in singing voice detection*, in Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Florence, Italy, 2014, pp. 7480–7484.

[26] B. Logan, *Mel frequency cepstral coefficients for music modeling*, in Proceedings of the International Symposium on Music Information Retrieval (ISMIR), Plymouth, Massachusetts, 2000.

[27] B. Logan and S. Chu, *Music summarization using key phrases*, in Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Istanbul, Turkey, 2000.

[28] G. Louppe, *Understanding random forests: From theory to practice*, 2015.

[29] J. Lyons, *Mel frequency cepstral coefficient (mfcc) tutorial*. `http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/`, 2012. [Online; accessed 22-November-2021].

[30] M. I. Mandel and D. Ellis, *Song-level features and support vector machines for music classification*, in ISMIR, 2005.

[31] M. Mauch, H. Fujihara, K. Yoshii, and M. Goto, *Timbre and melody features for the recognition of vocal activity and instrumental solos in polyphonic music*, in Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), Miami, Florida, USA, 2011, pp. 233–238.

[32] B. McFee, A. Metsai, M. McVicar, S. Balke, C. Thomé, C. Raffel, F. Zalkow, A. Malek, Dana, K. Lee, O. Nieto, D. Ellis, J. Mason, E. Battenberg, S. Seyfarth, R. Yamamoto, viktorandreevichmorozov, K. Choi, J. Moore, R. Bittner, S. Hidaka, Z. Wei, nullmightybofo, D. Hereñú, F.-R. Stöter, P. Friesch, A. Weiss, M. Vollrath, T. Kim, and Thassilo, *librosa/librosa: 0.8.1rc2*, may 2021.

[33] A. Mesaros, T. Virtanen, and A. Klapuri, *Singer identification in polyphonic music using vocal separation and pattern recognition methods*, in ISMIR, 2007.

[34] S. I. Mimilakis, C. Weiss, V. Arifi-Müller, J. Abesser, and M. Müller, *Cross-version singing voice detection in opera recordings: Challenges for supervised learning*, in Machine Learning and Knowledge Discovery in Databases – Proceedings of the International Workshops of ECML PKDD 2019, Part II, vol. 1168 of Communications in Computer and Information Science, Würzburg, Germany, 2019, pp. 429–436.

[35] E. Monk-Turner and D. Sylvertooth, *Rap music: Gender difference in derogatory word use*, American Communication Journal, 10 (2008).

[36] A. Oppenheim, *Speech analysis-synthesis system based on homomorphic filtering.*, The Journal of the Acoustical Society of America, 45 2 (1969), pp. 458–65.

[37] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, *Scikit-learn: Machine learning in Python*, Journal of Machine Learning Research, 12 (2011), pp. 2825–2830.

[38] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, *The MUSDB18 corpus for music separation*, dec 2017.

[39] Z. Rafii, A. Liutkus, F. R. Stöter, S. I. Mimilakis, D. FitzGerald, and B. Pardo, *An overview of lead and accompaniment separation in music*, IEEE/ACM Transactions on Audio, Speech, and Language Processing, 26 (2018), pp. 1307–1335.

[40] M. Ramona, G. Richard, and B. David, *Vocal detection in music with support vector machines*, in Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Las Vegas, Nevada, USA, 2008, pp. 1885–1888.

[41] M. Rocamora and P. Herrera, *Comparing audio descriptors for singing voice detection in music audio files*, in Brazilian symposium on computer music, 11th. san pablo, brazil, vol. 26, 2007, p. 27.

[42] J. Salamon, E. Gómez, D. P. W. Ellis, and G. Richard, *Melody extraction from polyphonic music signals: Approaches, applications, and challenges*, IEEE Signal Processing Magazine, 31 (2014), pp. 118–134.

[43] J. Schlüter and T. Grill, *Exploring data augmentation for improved singing voice detection with neural networks*, in Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), Málaga, Spain, 2015, pp. 121–126.

[44] J. Schlüter and B. Lehner, *Zero-mean convolutions for level-invariant singing voice detection*, in Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), Paris, France, 2018, pp. 321–326.

[45] D. Shakespeare, L. Porcaro, E. Gómez, and C. Castillo, *Exploring artist gender bias in music recommendation*, in Proceedings of the Workshops on Recommendation in Complex Scenarios and the Impact of Recommender Systems co-located with 14th ACM Conference on Recommender Systems (RecSys 2020), Online, September 25, 2020, T. Bogers, M. Koolen, C. Petersen, B. Mobasher, A. Tuzhilin, O. S. Shalom, D. Jannach, and J. A. Konstan, eds., vol. 2697 of CEUR Workshop Proceedings, CEUR-WS.org, 2020.

[46] A. Sherstinsky, *Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network*, ArXiv, abs/1808.03314 (2018).

[47] P. Smaragdis and J. C. Brown, *Non-negative matrix factorization for polyphonic music transcription*, in Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), 2003, pp. 177–180.

[48] A. Smola and B. Schölkopf, *A tutorial on support vector regression*, Statistics and Computing, 14 (2004), pp. 199–222.

[49] I. Smule, *Damp-vpb: Digital archive of mobile performances - smule vocal performances balanced*, 2017.

[50] F. R. Stoter, S. Uhlich, A. Liutkus, and Y. Mitsufuji, *Open-unmix - a reference implementation for music source separation*, Journal of Open Source Software, (2019).

[51] N. Takahashi, N. Goswami, and Y. Mitsufuji, *Mmdenselstm: An efficient combination of convolutional and recurrent neural networks for audio source separation*, 2018.

[52] S. Vembu and S. Baumann, *Separation of vocals from polyphonic audio recordings*, in Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), London, UK, 2005, pp. 337–344.

[53] Y. Wang, P. Getreuer, T. Hughes, R. F. Lyon, and R. A. Saurous, *Trainable frontend for robust and far-field keyword spotting*, in Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), New Orleans, USA, 2017, pp. 5670–5674.

[54] C. Xu, N. C. Maddage, and X. Shao, *Automatic music classification and summarization*, IEEE Transactions on Speech and Audio Processing, 13 (2005), pp. 441–450.