Isolating the singing voice from music tracks: a deep neural networks approach to karaoke

Jonathan Deboosere

Supervisors: Prof. dr. Tijl De Bie, Dr. ir. Thomas Demeester
Counsellor: Paolo Simeone

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in Computer Science Engineering

Department of Electronics and Information Systems
Chair: Prof. dr. ir. Koen De Bosschere

Department of Information Technology
Chair: Prof. dr. ir. Bart Dhoedt

Faculty of Engineering and Architecture
Academic year 2017-2018

GHENT
UNIVERSITY

Isolating the singing voice from music tracks: a deep neural networks approach to karaoke

Jonathan Deboosere

Supervisors: Prof. dr. Tijl De Bie, Dr. ir. Thomas Demeester
Counsellor: Paolo Simeone

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in Computer Science Engineering

Department of Electronics and Information Systems
Chair: Prof. dr. ir. Koen De Bosschere

Department of Information Technology
Chair: Prof. dr. ir. Bart Dhoedt

Faculty of Engineering and Architecture
Academic year 2017-2018

GHENT
UNIVERSITY

Permission of usage

Acknowledgements

I would like to thank my supervisors prof. dr. Tijl De Bie and dr. ir. Thomas Demeester, and my counsellor Paolo Simeone for helping me during the process of my master's dissertation. They have given me many tips to improve my model and they really helped me get a better understanding of the different techniques of deep learning and the endless possibilities. Also the idea of focusing on other transformations than the standard short-time Fourier transform, such as the constant-Q transform helped me improve the quality of the predictions and gives a great added value to the thesis and to the Music Information Retrieval (MIR) Community. Further research on the constant-Q transformation e.g. for other MIR problems could be very interesting and should certainly be looked at in more detail in the future.

Abstract

This thesis focuses on a Music Information Retrieval (MIR) problem, i.e. monaural source separation from music tracks. This problem comes down to extracting one particular signal, e.g. the vocals, from a mixture of multiple signals that are overlapping. Recent advances in MIR show how deep neural networks may tackle this problem by using convolutional networks. We make use of both convolutional neural networks and deep convolutional U-nets. The experimental framework considers two approaches for the preprocessing of the music files. The first method to transform the music signals is done by using a short-time Fourier transformation (STFT) which is the standard approach when looking at the state of the art for this problem. The second approach is to use a constant-Q transformation (CQT). This transformation increases the buffer size from high to lower frequencies, compared to the constant buffer size that is used in a STFT. This corresponds to logarithmically spaced frequency bins, which makes it more comparable to how the human hearing intercepts sound waves, since this also works on a logarithmic scale. The CQT is computationally more expensive than the STFT, but the results are promising for future research. Both approaches were used in combination with our deep convolutional U-net and different evaluation metrics were compared. These metrics show that the the model is more consistent in its predictions using CQT, rather than STFT. Furthermore, subjective tests were made to gain new insights in how well the CQT and the STFT perform according to the hearing experience, compared to the objective metrics. The results of the subjective tests show that the instruments predictions using the CQT perform much better compared to the voice predictions, whereas this was not the case for the evaluation with the objective metrics. According to the results of the subjective tests, predictions from songs whose arrangement includes string instruments like guitars seem to improve.

Keywords: deep convolutional U-net, audio source-separation, constant-Q transform, subjective testing

Isolating the singing voice from music tracks: a deep neural networks approach to karaoke

Jonathan Deboosere

Supervisors: Prof. dr. Tijl De Bie, Dr. ir. Thomas Demeester
Counsellor: Paolo Simeone

***Abstract*— Recent advances in Music Information Retrieval (MIR) show how deep neural networks may tackle the source-separation problem by using convolutional networks. A deep convolutional U-net is proposed in combination with the usage of a constant-Q transform (CQT) which is logarithmically scaled and mirrors the human auditory system. A comparison is made between the usage of CQT and the standard short-time Fourier transform (STFT) which has, in contrast to the CQT, equally spaced frequency bins. Objective metrics show that the model is more consistent in its predictions when CQT is used. However, the means of the metrics are very similar. Subjective tests give us new insights and show that in general, a prediction for the instruments with the deep U-net is much more appreciated by the listeners when CQT is used.**

Keywords*— deep convolutional U-net, audio source-separation, constant-Q transform, subjective testing

I. Introduction

The audio source-separation problem is a typical Music Information Retrieval (MIR) problem. When listening to a song, it is not that difficult for human beings to distinguish the vocals from the instruments and vice-versa. Humans are very good at solving these tasks as they are naturally able to focus their attention on the source of the sound they want to listen to. However, for machines there is still a lot of room for improvement. Recent research in deep neural networks has opened different paths towards elegant solutions for this problem, such as the usage of convolutional networks. A deep convolutional U-net is proposed which is based on the paper from Jansson *et al.* [1]. The deep convolutional U-net was initially developed for bio-medical imaging. Instead of using images, the model is now trained on two-dimensional data from the spectrograms.

There are several ways for creating a spectrogram. Two techniques, i.e. short-time Fourier transform (STFT) and constant-Q transform (CQT) are discussed and compared by analyzing both objective metrics and subjective scores. The CQT is known to be logarithmically scaled in the frequency domain and thus mirrors the human auditory system in a better way than STFT does. A disadvantage of the CQT is that it has no exact inverse. This is due to the fact that the transform matrix is no longer square. However, there exist approaches which can construct a CQT that is (approximately) invertible. Three possible implementations for this are discussed by J. Ganseman *et al.* [2].

For the experimentation, the iKala dataset [3] is used. This dataset contains 252 excerpts of 30 seconds each. The tracks have a sampling rate of 44100Hz which makes it useful for testing the final model with more popular songs which mostly have the same sampling rate. The iKala dataset has been used a lot for research in music source-separation and thus has the advantage that the metrics of a new model can be more easily compared with the state-of-the-art.

II. Methodology

A. Data Pre-processing

A first step that is needed before the data is fed into the neural network, is to pre-process the wave files from the iKala dataset. For every song in the iKala dataset, we obtain the groundtruth for the instruments and the vocals by taking the left and right channel from the stereo track, respectively. These wave files will be useful for creating the true labels from which the network will learn. Furthermore, the stereo track also needs to be converted into a mono signal. This way, a monaural waveform is created which contains all the sources. This will be used for creating the input data.

A second step is to split every wave file (from both the mixture and the separate sources) into chunks of 10 seconds. Each chunk is then transformed into a two-dimensional spectrogram which contains now values in both time and frequency domain. This transformation can be done by either using a STFT, or a CQT. A spectrogram consists of complex values which contain both amplitude and phase information. Typically, the power spectrogram is taken, such that we only focus on the amplitude information. An issue with this approach is that it is not possible to rebuild a wave file from a power spectrum, because the phase information is thrown away. To solve this, there exists a technique called Wiener filtering which uses the power spectrograms from all the sources of the mix to create an ideal mask. The mask for the voice is created as follows:

$$mask_{vocals} = \frac{|vocal\ spectrogram|^2}{|vocal\ spectrogram|^2 + |instruments\ spectrogram|^2}$$

With this mask, it is possible to rebuild the full spectrogram by taking the spectrogram of the mixture and multiplying it with the square root of the mask. This way, both amplitude and phase information are regained. With the spectrogram, we are able to apply an inverse STFT or a (pseudo-)inverse CQT for obtaining a wave file again.

Knowing this, we can use the mask as a label when training the neural network. The power spectrum of the mixture will be used as an input. Something to keep in mind is that the mask contains values in the interval [0, 1]. By subtracting 1 with these values, we obtain a mask that could be used for recreating the spectrogram for the instruments.

B. *Training the Model*

By feeding the training data (the input power spectrograms and corresponding labels (masks)) to the model in small batches, the network learns how to distinguish the vocals from the instruments. Both the data pre-processed with the CQT and with the STFT were fed to the deep U-net separately. This way, we will have a model for both transformations from which the metrics can be compared with each other in a later stadium.

C. *Post-processing*

As said before, the power spectrogram of a mixed song is used to feed into the network. However, after training the model, we want to test it and see how well the predicted wave files sound. In order to rebuild these wave files, we first need the full spectrogram again from the mixed song. This is done with either STFT or CQT. Once we have this spectrogram, we can recreate a spectrogram of the predicted source as follows:

$$spectrogram_{vocals} = spectrogram_{mix} \times \sqrt{\widehat{mask}_{vocals}}$$

where $\widehat{mask}_{vocals}$ is the predicted mask for the vocals. The same technique can be applied for the instruments. Once we have the predicted spectrogram, an inverse STFT or (pseudo-) inverse CQT can be applied to obtain a wave file again.

III. The Model

The model that is proposed is a deep convolutional U-net which contains both an encoding and decoding phase. First, the power spectrogram of the mixture is fed into strided convolutional layers. A stride of 2 is used which means that the output of every convolutional layer will be approximately half the size of the input that is fed into it. This way, the input is downsampled a lot. However, the amount of output channels is doubled every subsequent convolutional layer. After every convolution, a leaky relu activation function is used with a leak of 0.2. Also dropout is added with a dropout rate of 0.5 which is used for regularization. After this stage, the network learns how to upsample the downsampled image again. This is done in the decoding phase which consists of 'deconvolutional' layers, each activated with a relu. Although the deconvolutional layers are not the exact inverse operation of the convolutional layers, they do make sure that the image is upsampled again to the exact same size as the initial image that was fed into the network. For every subsequent layer, the amount of output channels are now halved again. Only in the last layer, we keep two output channels to obtain a prediction for both the vocals and instruments. In addition, skip-connections are used to further optimize the model. With these skip-connections, we can concatenate the output of each convolutional layer with its corresponding deconvolutional layer, containing the same amount of output channels. This concatenation is then fed into the next layer.

Since the labels that we feed into the network are only the masks of the vocals, we can calculate the loss as follows. First of all, we calculate the mean square error (MSE) for the voice prediction.

$$MSE_{voice} = \frac{1}{n}\sum_{i=1}^{n}(mask_{voice,i} - \widehat{mask}_{voice,i})^2$$

where $\widehat{mask}_{voice}$ is the predicted mask for the vocals.

The same can be done for the instruments prediction by first calculating the true mask for the instruments. By subtracting an array of ones (with the same dimensions of $mask_{voice}$) with $mask_{voice}$, we obtain $mask_{instruments}$. Now, $MSE_{instruments}$ can be easily calculated. The following equation shows how the total loss function is calculated:

$$RMSE = \sqrt{\alpha \times MSE_{voice} + (1-\alpha) \times MSE_{instruments}}$$

where $\alpha = 0.5$ by default. To optimize this loss, the Adam optimizer is used.

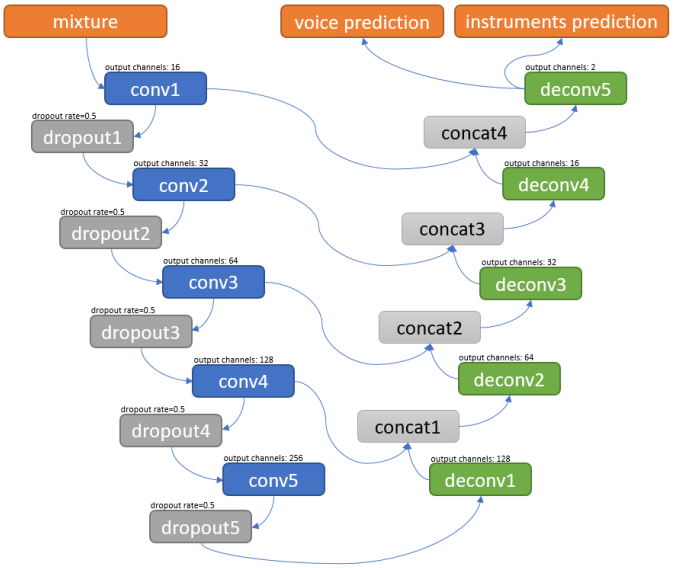

Figure 1 Proposed deep convolutional U-net.

IV. Evaluation

Evaluating the deep convolutional U-net for both STFT and CQT can be done in two ways. First of all, the objective metrics can be calculated by comparing the predicted wave files with the groundtruth of both vocals and instruments. Secondly, we can gain new insights by using subjective tests and focus more on the actual hearing experience from certain a group of people.

A. *Objective metrics*

The most common metrics that are used for the audio source-separation problem are the source-to-distortion ratio (SDR), source-to-interference ratio (SIR) and source-to-artifacts ratio (SAR), and are discussed in detail by Vincent *et al.* [4]. These metrics are expressed in decibels (dB). The higher these metrics are, the better the prediction. Table 1 shows the averages of the calculated metrics for both the STFT and the CQT approach, using the same convolutional deep U-net. Note that for both approaches, the calculated metrics from the vocal predictions are in general more than 5dB higher than for the instruments predictions. In general, the results from STFT and CQT are very similar and there are not a lot of remarkable differences at first sight.

Table 2 Comparison of metrics (average) for both the STFT and CQT experiments.

		Deep Convolutional U-Net	
		STFT	**CQT**
vocals	**SDR**	11.04	11.38
	SIR	17.65	17.23
	SAR	13.62	13.64
instruments	**SDR**	5.34	5.03
	SIR	13.06	10.06
	SAR	6.84	8.30

If we have a closer look at the metrics in Figure 2 and 3, we notice that there are much more outliers when the STFT is used than when CQT is used. We could say that the model is more consistent in its predictions when CQT is applied.

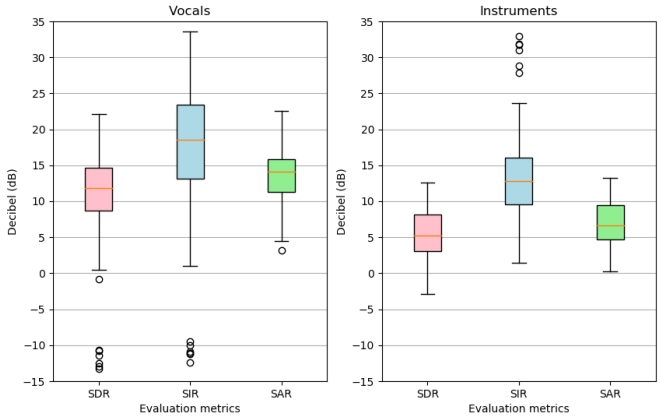

Figure 2 Boxplot of metrics for deep U-net (using STFT).

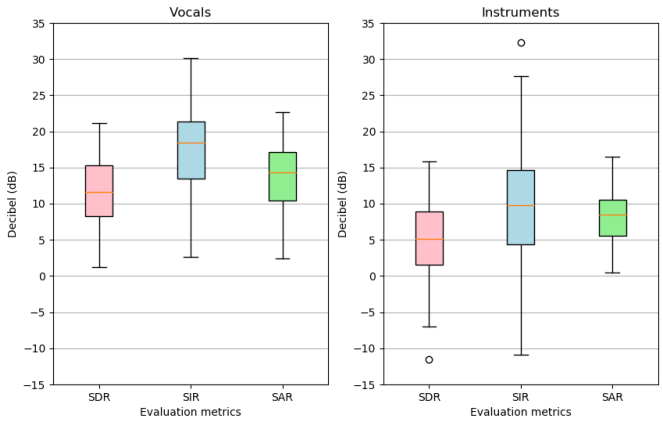

Figure 3 Boxplot of metrics for deep U-net (using CQT).

B. Subjective tests

A second way of evaluating the two approaches, is by letting a group of people listen to a set of samples and by making an analysis of the scores which they have given to the predictions. The surveys contained 18 samples from which the vocal and instruments predictions were made (for both STFT and CQT approach). A score was given from 0 (not isolated at all) to 6 (very well isolated). Figures 4 and 5 show that in general, the model performs better for the instruments predictions when CQT is used. For the vocal predictions, it is less obvious which transformation leads to a better hearing experience. In general, the CQT is a little better for the voice predictions, but the results are very close.

Because the subjective tests were only done with a small group of people, we cannot simply assume that the CQT is better in all cases. It does however give an indication that it could be a better transformation to use.

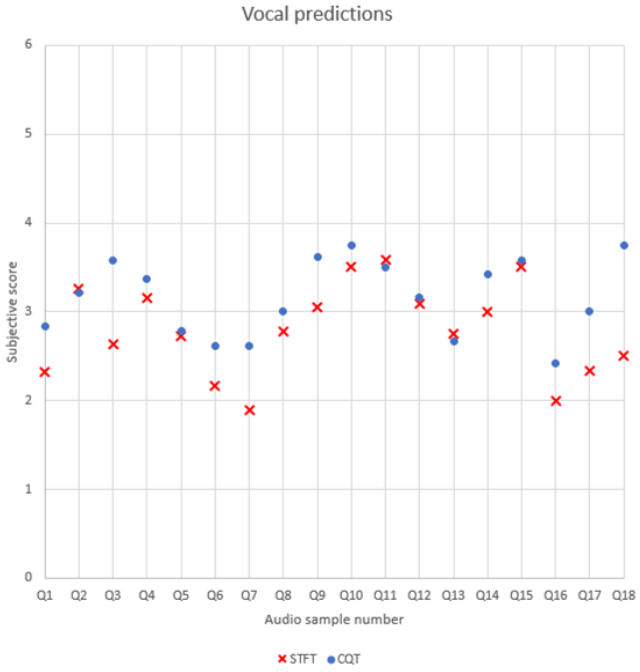

Figure 4 Weighted average of the subjective scores for every sample prediction of the vocals.

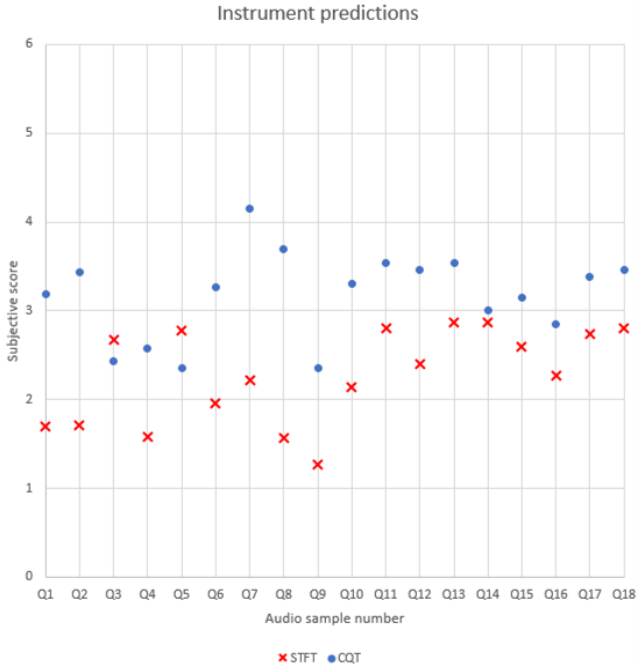

Figure 5 Weighted average of the subjective scores for every sample prediction of the instruments.

V. Conclusions

A lot of research has already been done for the audio source-separation problem. However, most of them only focus on training the parameters from the used neural network as good as possible and simply use a standard short-time Fourier transform (STFT). Instead of a STFT, we propose a constant-Q

transformation (CQT) which could improve the predictions of the neural network, because the CQT has a logarithmic scale and thus mimics the human auditory system in a better way.

By comparing both transformations using the same deep convolutional U-net, we notice that in general, there are not a lot of remarkable differences when looking at the objective metrics. We do however notice that the metrics of the predictions with STFT contain much more outliers than when CQT is used. This could imply that the model is more consistent in its predictions when a CQT is used.

Evaluating the two approaches in a more subjective way, we learn that instruments predictions are almost always improved with CQT. Especially when a lot of (distorted) guitars are contained within the song. For the vocal predictions, it is hard to say which transformation should be used. Both objective and subjective tests give approximately the same results. For the vocal predictions, we notice that it is much harder for the model to filter out everything from the drums. These are still very noticeable in both STFT and CQT approaches. This could however be improved by applying some extra filters during pre-processing.

REFERENCES

[1] A. Jansson, E. Humphrey, N. Montecchio, R. 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)*, 2017, pp. 323-332.

[2] J. Ganseman, P. Scheunders, and S. Dixon, "Improving plca-based score-informed source separation with invertible constant-q transforms," in *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European.* IEEE, 2012, pp. 2634-2638.

[3] MACLab, "The ikala dataset," 2017, [Online; accessed April 30, 2018]. [Online]. Available: http://mac.citi.sinica.edu.tw/ikala/

[4] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," *IEEE transactions on audio, speech, and language processing*, vol. 14, no. 4, pp. 1462-1469, 2006.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

When listening to a song, it is not that difficult for a human being to distinguish the vocals from the instruments and vice-versa. Even when we are at a party, we are able to focus on one particular person's voice while all the other people in the room are talking as well, or when there is a lot of background noise. Humans are very good at solving these tasks as they are naturally able to focus their attention on the source of the sound they want to listen to. However, for machines there is still a lot of room for improvement. These problems are often called 'Cocktail party problems'. In this paper, the focus will be on one specific sub-problem, i.e. the isolation of a singing voice from music. This can be useful in many software tools aimed to private end-users interested in karaoke-like apps as well as to professionals of the music industry interested in recording, post production, and (re)mastering. Recent research in deep neural networks has opened different paths towards elegant solutions for this problem. This requires a high computational cost.

In this thesis, some of the existing approaches for this particular problem will be discussed, followed by an explanation of the methodology. During the experimentation, two main models have been used. The first model that will be discussed is a deep convolutional network. The progress of how this model was improved will be explained, together with all the lessons learned. Different techniques for improving the model, such as dropout and data augmentation were used during this process. Even after doing all the optimizations, this model still did not perform very well in its predictions. For this reason, a second model called a deep U-net is introduced where deconvolutional layers are added to the model. These layers do the inverse operation of what happens in a convolutional layer. This model is often represented in the shape of a U, hence the name U-net. The idea for using

this architecture for the audio source separation problem was introduced by Jansson *et al.* [3] and is currently one of the best known performing approaches for this purpose. The deep U-net used for this thesis is not tuned as much as is done by Jansson *et al.*, but the metrics are close to their results and the predictions are acceptable. All the improvements that were used in the first model are again applied to this model. Also batch normalization and a newer activation function, i.e., the leaky ReLu are applied.

So far, a lot of research can be found where only short-time Fourier transformations (STFT) are used in order to create a two-dimensional dataset. However, other transformations exist, such as the constant-Q transform (CQT). CQT has the advantage that it mimics the human hearing in a better way than STFT does, because of its logarithmic scale. We will take a closer look at the CQT and also use this transformation for preprocessing the data. A comparison will be made between the predictions of the final model using CQT and the predictions of the exact same model, but using STFT instead. The main purpose of this thesis is not just about finding a model that surpasses every other existing approach, but rather to investigate how much influence the used transformation can have on the predictions of the model.

The used framework which is written in Python will be briefly explained. For the implementation of the neural network, the TensorFlow library is used, as this is widely supported by industry. Furthermore, the results will be compared with the state of the art, and several evaluation metrics will be discussed. Also, results of subjective tests will be discussed, in order to get different insights in how well the model performs when using CQT and STFT.

Chapter 2

State of the art

2.1 Source Separation

In this master's thesis, the term *source separation* should be interpreted in the context of audio, where we have a mixture of signals, i.e. multiple instruments and vocals, and we want to extract or isolate one certain component from it. The mixed sources are assumed to be monaural, so it is not possible to just split multiple channels and assume that one source will be more likely to be isolated from the other sources in one channel.

Using just the raw audio files for separating sources is not the way to go[1], because all of the different harmonics (different frequencies) are overlapping in this case and we would only have data in the time domain. Only if the different sources never appeared at the same time, it would be possible to separate them. Otherwise, there would be too much overlap.

The short-time Fourier transform (STFT) is a commonly used approach to obtain data not only in the time domain, but also in the frequency domain. STFT determines the sinusoidal frequency and phase information of local intervals of the signal as it changes over time. In other words, by using this transformation, we obtain (complex) values in both the time and frequency domain. In this case, all of the different harmonics will thus be separated. A visualization of this transformation is shown in figure 2.2 and is called the *spectrogram*. Another possibillity to transform a signal to the time-frequency domain is to use a Constant-Q Transformation

[1]There exists an approach by Google, called WaveNet [10] which is a deep neural network that does make use of the raw audio files during training. This is however for another MIR problem, i.e. for the generation of audio waveforms.

(CQT). The CQT can be understood as a STFT, but with logarithmically spaced frequency bins. This is accomplished by varying the length of the analysis window, instead of using a fixed window as is done in a STFT. Because the human hearing also works on a logarithmic scale, using this transformation could have a better influence on the predictions than the STFT does. Computationally, this transformation is more expensive compared to the STFT. The biggest disadvantage of the CQT is that it is not invertible, since a DC component cannot be calculated due to the logarithmic frequency spacing, and minimum and maximum frequency bands need to be defined outside the signal which will not be analyzed. However, there exist some methods which can construct a CQT that is (approximately) invertible. Three possible implementations for this are discussed by J. Ganseman *et al.* [11].

For source separation, spectrograms are much more useful than the raw audio files, because there are a lot more informative features that could be extracted from this. Normally, we take the absolute value of the spectrogram in order to obtain the amplitude values and get rid of the complex values. We call this the *magnitude spectrogram*. However, one disadvantage is that a magnitude spectrogram cannot be inverted, i.e. it is not possible to derive the estimate itself, because the phase information is lost. Nevertheless, there exists an approach called *Wiener filtering* which creates a mask for one certain source and then multiplies this mask with the spectrogram of the original mixture. This way, the phase information can (approximately) be recovered from the original signal. The mask is calculated as follows:

$$m_k = \frac{|S_k|^2}{\sum_{i=1}^{n} |S_i|^2}, \ where \ 1 \leq k \leq n \tag{2.1}$$

with m_k the mask for source k (e.g. the vocals), n the amount of sources in the mixture, and S_i the spectrogram of source i, where $1 \leq i \leq n$. This mask contains normalized values between zero and one for one specific source. By multiplying the original mixture spectrogram with the square root of this mask, the full spectrogram for the isolated source can again be obtained:

$$\hat{S}_k = S_m \cdot \sqrt{m_k} \tag{2.2}$$

with $\hat{S}_k$ the reconstructed spectrogram for source k, S_m the original spectrogram of the mixture, and m_k the prediction of the mask from source k. Finally, the audio file can be reconstructed using the inverse transformation, i.e. inverse short-time Fourier or inverse constant-Q transformation.

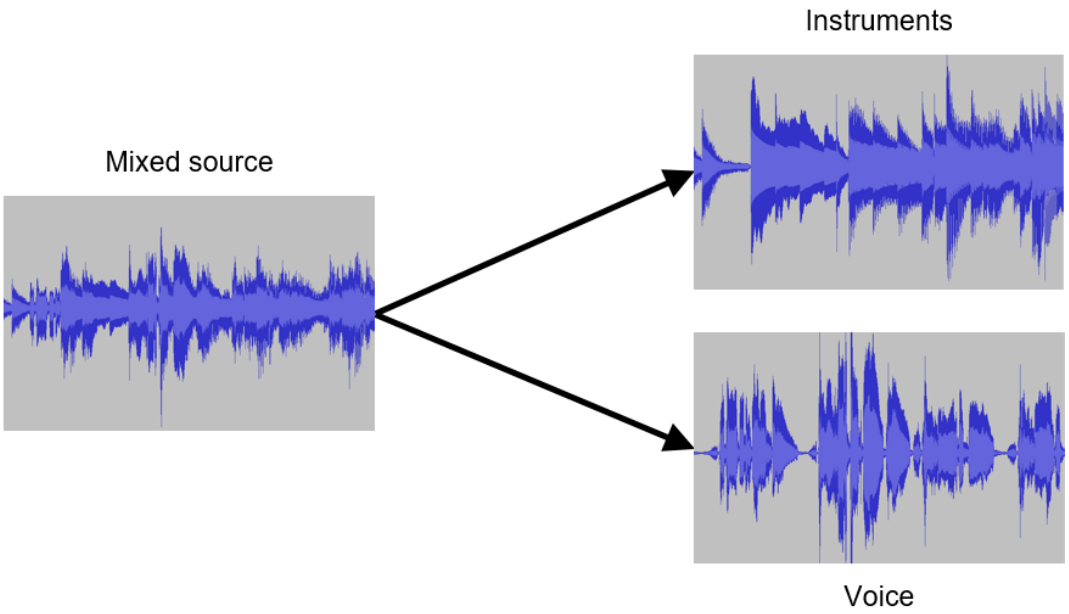

Figure 2.1: Sources in the time domain (overlapped harmonics)

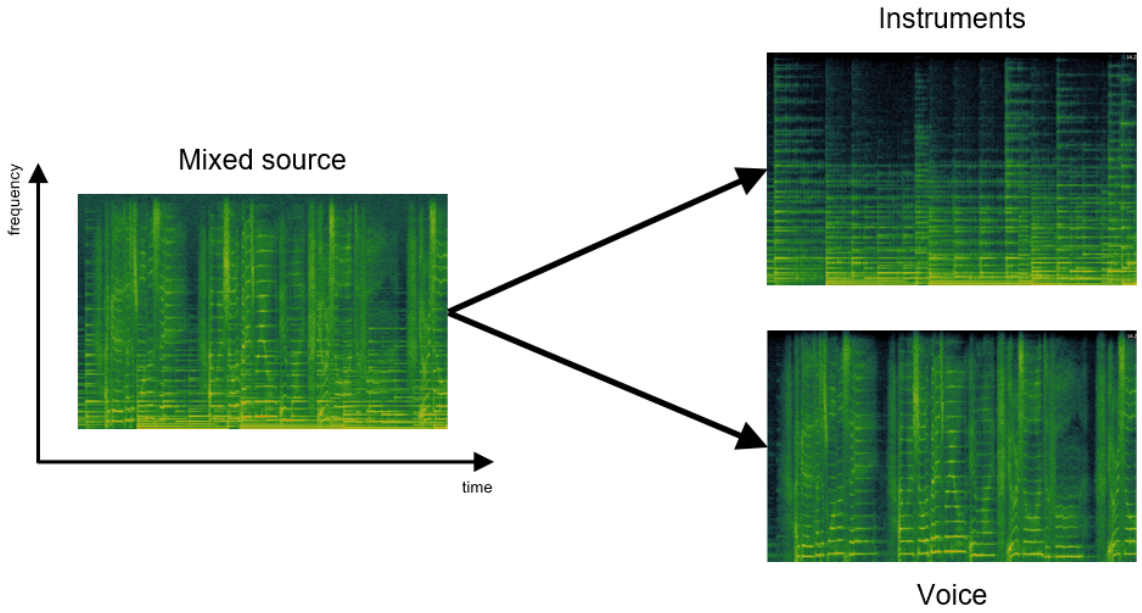

Figure 2.2: Sources in both time and frequency domain (separated harmonics)

2.1.1 Evaluation measures

Some evaluation metrics that are used a lot in research for audio source separation are discussed by E. Vincent *et al.* [12]. There are different kinds of distortion we can distinguish. When a source is estimated, it can be decomposed into a true source part and three error terms, i.e. interference, additive noise, and artifacts.

$$\hat{s}_j = s_{target} + e_{interference} + e_{noise} + e_{artifacts} \tag{2.3}$$

where $\hat{s}_j$ is the estimated source.

The decomposition of this signal is based on orthogonal projections. For a subspace spanned by the vectors $y_1, ..., y_k$, we denote $\Pi\{y_1, ..., y_k\}$ as the orthogonal projector onto this subspace. Three orthogonal projectors are defined:

$$P_{s_j} := \Pi\{s_j\}, \tag{2.4}$$

$$P_{\mathbf{s}} := \Pi\{(s_{j'})_{1 \leq j' \leq n}\}, \tag{2.5}$$

$$P_{\mathbf{s,n}} := \Pi\{(s_{j'})_{1 \leq j' \leq n}, (n_i)_{1 \leq i \leq m}\}. \tag{2.6}$$

The four terms of the decomposed source $\hat{s}_j$ can then be defined as follows:

$$s_{target} := P_{s_j}\hat{s}_j, \tag{2.7}$$

$$s_{interference} := P_{\mathbf{s}}\hat{s}_j - P_{s_j}\hat{s}_j, \tag{2.8}$$

$$s_{noise} := P_{\mathbf{s,n}}\hat{s}_j - P_{\mathbf{s}}\hat{s}_j, \tag{2.9}$$

$$s_{artifacts} := \hat{s}_j - P_{\mathbf{s,n}}\hat{s}_j. \tag{2.10}$$

The computation of s_{target} is straightforward and only involves an inner product: $s_{target} = \langle \hat{s}_j, s_j \rangle s_j / ||s_j||^2$. If the sources are mutually orthogonal, then $s_{interference} = \sum_{j' \neq j} \langle \hat{s}_j, s_{j'} \rangle s_{j'} / ||s_{j'}||^2$. Otherwise, we need to define a vector $\mathbf{c}$ of coefficients: $\mathbf{c} = \mathbf{R}_{\mathbf{ss}}^{-1}[\langle \hat{s}_j, s_1 \rangle, ..., \langle \hat{s}_j, s_n \rangle]^H$, where $\mathbf{R}_{\mathbf{ss}}$ is the Gram matrix and $(.)^H$ denotes the Hermitian transposition. We now can define $P_{\mathbf{s}}\hat{s}_j$ as follows: $P_{\mathbf{s}}\hat{s}_j = \sum_{j'=1}^{n} \bar{c}_{j'} s_{j'} = \mathbf{c}^H \mathbf{s}$. $P_{\mathbf{s,n}}$ can be computed in a similar fashion. We can however assume that the noise signals are mutually orthogonal so that $P_{\mathbf{s,n}}\hat{s}_j \approx P_{\mathbf{s}}\hat{s}_j + \sum_{i=1}^{m} \langle \hat{s}_j, n_i \rangle n_i / ||n_i||^2$.

Based on the decomposition of $\hat{s}_j$, we can now define several performance measures, i.e. the Source to Distortion Ratio (SDR), the Source to Interference Ratio (SIR), the Source to Noise Ratio (SNR), and the Source to Artifacts Ratio (SAR). These

metrics represent energy ratios and are expressed in decibels (dB). The higher these values are, the better the performances.

- Source to Distortion Ratio

$$SDR := 10 \cdot log_{10}\Big(\frac{||s_{target}||^2}{||e_{interference} + e_{noise} + e_{artifacts}||^2}\Big) \tag{2.11}$$

- Source to Interference Ratio

$$SIR := 10 \cdot log_{10}\Big(\frac{||s_{target}||^2}{||e_{interference}||^2}\Big) \tag{2.12}$$

- Sources to Noise Ratio

$$SNR := 10 \cdot log_{10}\Big(\frac{||s_{target} + e_{interference}||^2}{||e_{noise}||^2}\Big) \tag{2.13}$$

- Sources to Artifacts Ratio

$$SAR := 10 \cdot log_{10}\Big(\frac{||s_{target} + e_{interference} + e_{noise}||^2}{||e_{artifacts}||^2}\Big) \tag{2.14}$$

2.2 Existing Approaches

2.2.1 Neural Networks

Since the computational power for computers has increased a lot, different machine learning approaches have become more successful in all kind of fields, such as speech and object recognition. One machine learning technique that has increased a lot in popularity these days is deep neural networks. These are neural networks with more than one hidden layer. For every layer, an activation function is used which captures non-linear relationships between the inputs. This way, the network is able to learn more complex relationships. Knowing that most real-life problems can be pretty complex, this is eventually what we want to achieve.

In the next sections, a brief introduction of the basic concepts of neural networks is provided. Some well-known approaches in deep learning are briefly discussed for this particular problem, i.e. audio source separation. The most common ones are convolutional and recurrent neural networks, but some other techniques are introduced as well.

2.2.1.1 Convolutional Neural network

A first example of an existing architecture for the monaural source separation problem is a convolutional neural network. This is a specialized kind of neural network that uses convolution instead of general matrix multiplication and is known best in image processing problems, such as object recognition and classification. The model sees every input as some sort of grid, e.g. an image can be considered as a two-dimensional grid of pixel values. Suppose we have a small image of 100 by 100 pixels. When using a fully connected layer, we would have 10000 weights for each neuron in the second layer. When using a convolutional layer, the number of parameters can be reduced a lot. We could for example divide the image in regions of 5 by 5 pixels which would only require 25 learnable parameters (times the amount of output channels) knowing that each region uses the same shared weights. This approach also helps to allow much deeper networks, with fewer parameters to train.

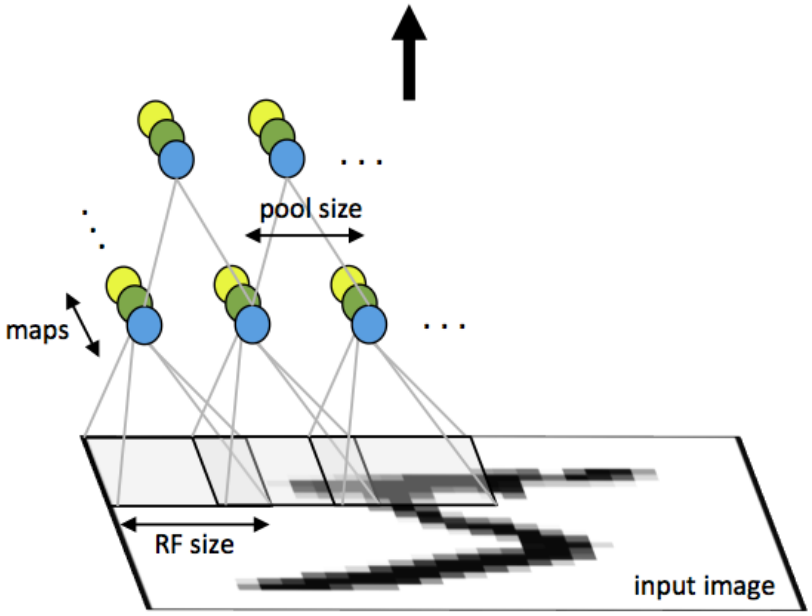

Figure 2.3: Convolutional layer [1]

Convolutional neural networks (CNNs) have been very successful in the identification of faces and objects in images, but M. Espi *et al.* [2] thought this model could be useful in monaural source separation problems as well. By applying a short-time Fourier transform on a music signal, a spectrogram can be obtained. This spectrogram can then be seen as a two-dimensional grid (i.e. in both time and frequency domain). The rest can be done in a similar way as in image recognition.

By using a convolutional neural network, the local characteristics of a spectrogram can be exploited. A common use in CNNs is alternating convolutional layers with pooling layers. Pooling layers are used for reducing the spatial size of the data, which makes the computations less expensive. It can also be useful to prevent overfitting. Although pooling can be very useful in image recognition, its use is less effective in music source separation, because when too much detail in musical data is reduced, it can be easily noticed. M. Espi *et al.* [2] experimented with different pooling layers both along frequency and along time, and their results showed that the performance degrades visibly when pooling is included along the frequency domain.

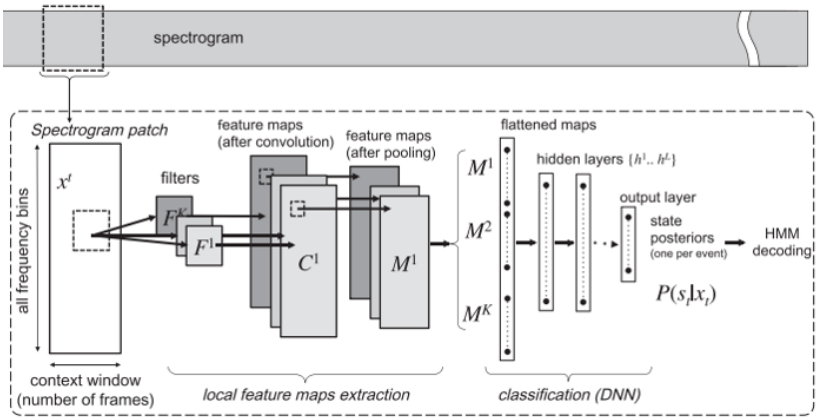

Figure 2.4: Proposed convolutional network architecture of M. Espi *et al.* [2]

2.2.1.2 Vector Product Neural Network

A less obvious choice for solving this problem was proposed by Z.-C. Fan *et al.* [13]. They propose a novel neural network model for music signal processing which uses vector product neurons and dimensionality transformations. They map the inputs from real values into three-dimensional vectors. These vectors are then fed into a three-dimensional vector product neural network. The final outputs are then mapped back into the real values. Two models for dimensionality transformation are proposed: context window (context-Window Vector Product Neural Network or WVPNN) and spectral coloring (Colored Vector Produc Neural Network or CVPNN). For their experiments, they used the iKala dataset [14]. 189 clips from this dataset were used for testing and only 63 clips for training. This was due to the limitation of their GPU memory. To reduce computation, all the clips were downsampled to 16000Hz. For evaluation they used the global NSDR (GNSDR), global SIR (GSIR), and global SAR (GSAR). These are the same metrics as already

mentioned before in section 2.1.1, except that the weightings are now proportional to the length of each clip. Results in table 2.1 show that both their proposed models WVPNN and CVPNN perform better than traditional DNNs.

Neural networks	**Arch.**	**Context Window Size**	**GNSDR**	**GSIR**	**GSAR**
DNN_1	512x3	1	8.16	11.88	12.11
DNN_2	1536x3	1	8.37	12.64	11.82
CVPNN	512x3	1	8.87	13.38	11.37
DNN_3	1536x3	3	8.85	12.59	12.52
WVPNN	512x3	3	9.01	13.82	11.97

Table 2.1: Comparison of ordinary DNN with WVPNN and CVPNN

2.2.1.3 Deep U-Net Convolutional Network

Another possibility for solving the singing voice separation problem, is to use a deep U-net convolutional network. This means that the network consists of several convolutional layers, followed by deconvolutional layers as can be seen in figure 2.5. Deconvolution is used to reverse the effects of the convolutions in order to go back to the original size of the input. A deconvolution can be seen as the inverse operation of a convolution. The U-net architecture was initially developed for (bio-medical) imaging, but A. Jansson *et al.* thought it could be useful in musical applications as well. Instead of using images [15], spectrograms from music signals are used. Evaluation is done by comparing the evaluation metrics signal-to-distortion ratio (SDR), signal-to-interference ratio (SIR), and signal-to-artifacts ratio (SAR) with state of the art (shown in table 2.2). As can be seen in the table below, their U-net model has the highest scores which means that it performs better than the ChimeraNet, which was proposed by Y. Luo *et al.* [16].

		U-Net	**Chimera**
vocals	**NSDR**	11.094	8.749
	SIR	23.960	21.301
	SAR	17.715	15.642
instruments	**NSDR**	14.435	11.626
	SIR	21.832	20.481
	SAR	14.120	11.539

Table 2.2: iKala mean scores for the U-net model from Jansson *et al.* and the ChimeraNet from Luo *et al.*

After evaluating the metrics, they also made use of subjective tests to see if the observations from the metrics were the same as for the hearing experience of the predictions. For their survey, they asked CrowdFlower users to listen to three clips of the isolated audio, generated by their U-net model, a baseline model and Chimera. The order of these three clips was randomized. Another thing they did, was adding "control questions" to the survey. These contained an audio file without music or a prediction but with a person's voice asking the listeners to fill in a certain rating. Whenever such a control question was answered incorrectly, the user was disqualified from the task. For the survey, they used 25 clips from IKala and 42 from MedleyDB.

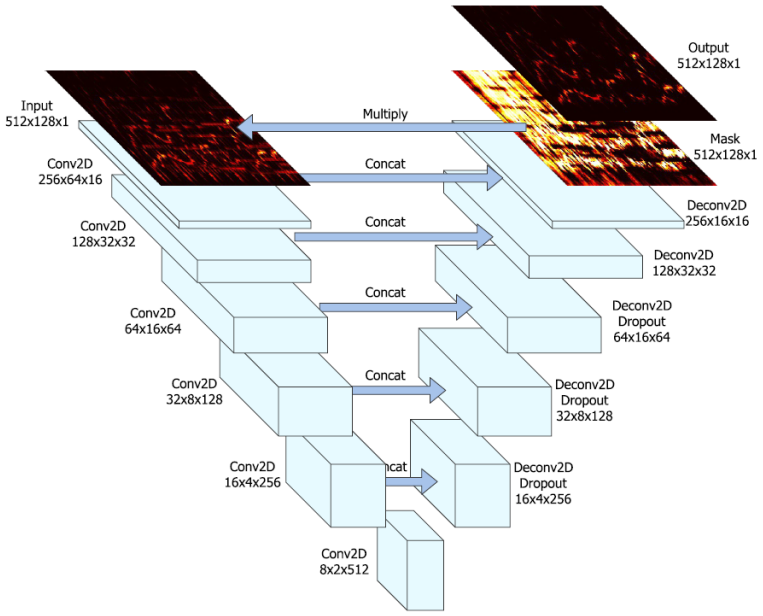

Figure 2.5: Proposed deep U-net convolutional network architecture by A. Jansson *et al.* [3]

A less deep network but with the same principle, is proposed by P. Chandna *et al.* [4]. Their architecture contains an encoding stage and a decoding stage just like in the previous example, but both stages only contain two (de)convolutional layers. The difference is that they focus more on one dimension per layer, i.e. in the first convolutional layer, the model will try to search for local timbre information which are features in the frequency domain, and in the second convolutional layer there is more focus on features in the time domain. As can be seen in figure 2.6 the encoding stage also contains a fully connected layer. This layers acts as a bottleneck and is used for dimensionality reduction. The decoding stage is then used for doing

the inverse operations of the encoding stage, such that the output gets back the original size. In their research, they do not only focus on separating the voice from music tracks but also every instrument separately. For these experiments, the Mixing Secrets Dataset 100 (MSD100) and the Demixing Secrets Dataset 100 (DSD100) were used. Both these datasets consist of 100 professionally produced full track songs. They contain separate tracks for drums, bass, vocals, and other instruments for each song in the set, which makes it ideal for cases where specific instruments need to be separated.

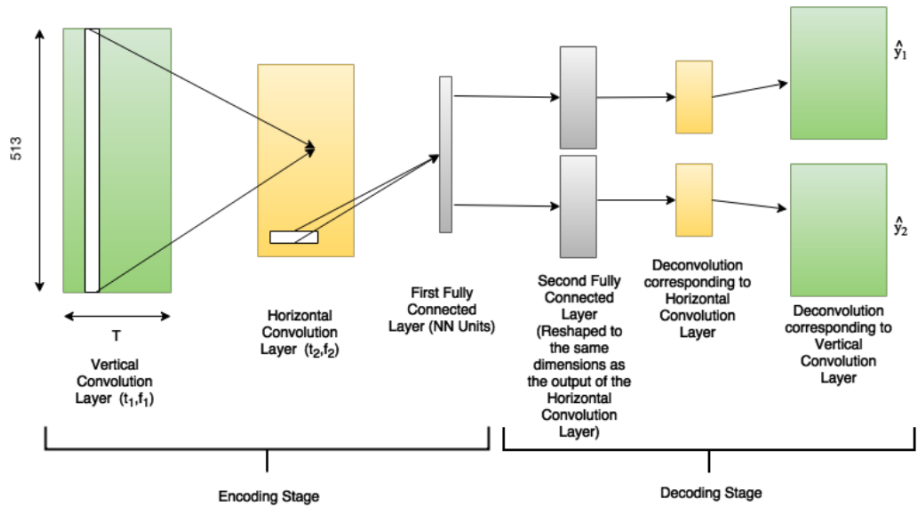

Figure 2.6: Proposed network architecture by P. Chandna *et al.* [4]

2.2.1.4 Recurrent Neural Network

Another example of an existing architecture for the monaural source separation problem is that from P.-S. Huang *et al.* [5]. They propose a deep recurrent neural network (RNN) consisting of three stacked layers, a dense layer (one for each source), and an extra normalization layer, which they call the time-frequency masking layer (also one for each source). A figure of the model architecture is shown in figure 2.8. The main idea behind a recurrent neural network is to make use of sequences of information where consecutive elements (e.g. words in a sentence, musical tones from a melody) are dependent on each other, compared to a traditional neural network where it is assumed that all inputs and outputs are independent of each other. The term recurrence indicates the hidden state that is brought along (and updated) every time step.
The diagram in figure 2.7 shows a RNN being unfolded into a full network, i.e. we write out the network for the complete sequence of information.

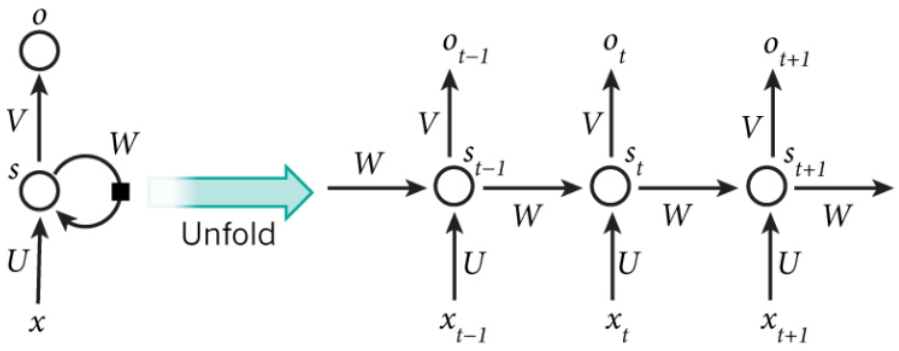

Figure 2.7: A recurrent neural network and the unfolding in time

P.-S. Huang *et al.* used their RNN to focus on singing voice separation, but speech separation and speech denoising were tested as well. For these tests, they used the MIR-1K, TSP, and TIMIT datasets, respectively. The MIR1-1K dataset is most relevant for this thesis. In order to evaluate the proposed model, the evaluation metrics global NSDR (GNSDR), global SIR (GSIR), and global SAR (GSAR) were used (as previously discussed in section 2.2.1.2) and compared to other approaches from the state of the art as shown in table 2.3. Their proposed RNN model performs better than the RPCA and RNMF approaches, which will be discussed briefly in section 2.2.2. Huang *et al.* also further explored different discriminative training objectives to enhance the source to interference ratio. Their approach achieves 2.30∼2.48 dB GNSDR gain and 4.32∼5.42 dB GSIR gain compared to the other models.

Unsupervised			
Model	GNSDR	GSIR	GSAR
RPCA [17]	3.15	4.43	11.09
RPCAh [18]	3.25	4.52	11.10
RPCAh + FASST [18]	3.84	6.22	9.19
Supervised			
Model	GNSDR	GSIR	GSAR
MLRR [19]	3.85	5.63	10.70
RNMF [20]	4.97	7.66	10.03
DRNN-2	**7.27**	**11.98**	**9.99**
DRNN-2 + discrim	**7.45**	**13.08**	**9.68**

Table 2.3: Comparison between the models proposed by Huang *et al.* [5] and other proposed approaches. The "discrim" denotes the models with discriminative training

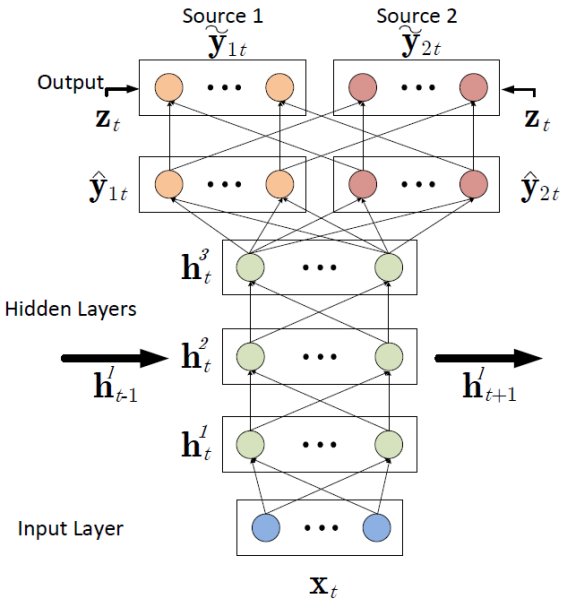

Figure 2.8: Proposed neural network architecture by P.-S. Huang *et al.* [5]

Another example of a recurrent neural network approach is that from Hershey *et al.* [21]. They introduced the term 'Deep clustering' which means they use a Bi-directional Long Short-Term Memory (BLSTM) neural network to learn useful embeddings of time-frequency bins of a mixture. To estimate the partition of one source, they use an objective function which encourages these embeddings to cluster according to their source such that K-means can be applied [21]. For these experiments, the focus was more to separate speech from multiple speakers. LSTMs are a commonly used type of RNNs and are know to be much better at capturing long-term dependencies. The fact that Hershey *et al.* use a bi-directional LSTM, basically just means that two independent LSTMs are put together. The input sequence is fed in normal time order for one network and in reverse time order for the other one. BLSTMs allow the network to have both forward as well a backward information about a given sequence at every time step.

So far, most of the existing approaches rely on a post processing step where Wiener filtering is applied, i.e. after training a mask is generated in order to rebuild the spectrogram of the source that needs to be predicted. P.-S. Huang *et al.* [5] made this post processing step unnecessary by adding an extra 'time-frequency layer' to their model itself which immediately returns a mask output. S. I. Mimilakis *et al.* [7] propose another method where the model trains directly on a source-dependent mask which makes the post processing step unnecessary. Furthermore, they apply

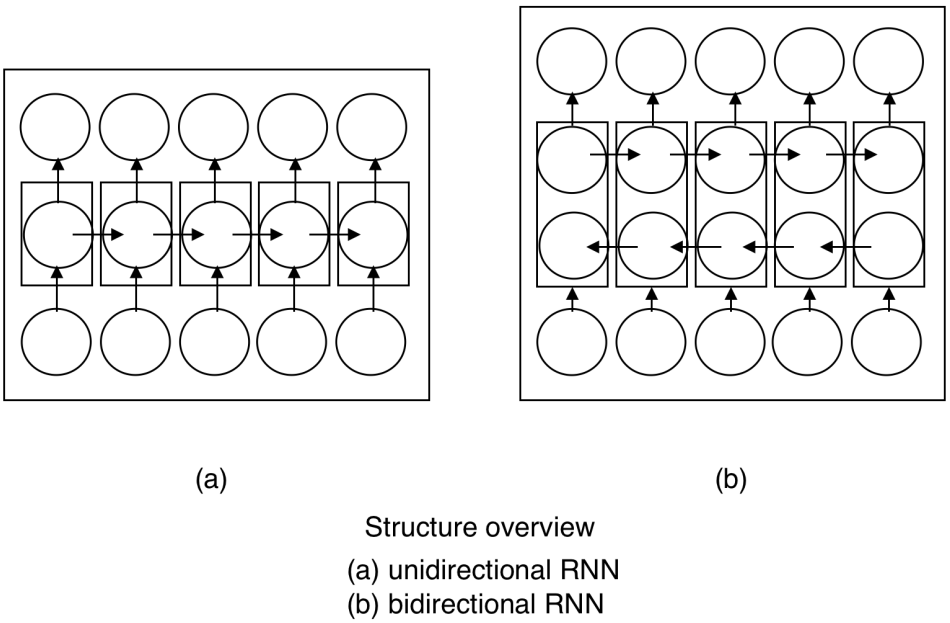

Figure 2.9: Basic idea of bi-directional RNN compared to uni-directional RNN [6]

a denoising filter to the results obtained from the time-frequency masking which enhances the result even more. Compared to previous state-of-the-art methods for music source separation from Grais *et al.* [22], Chandna *et al.* [4], and Mimilakis *et al.* [23, 24], they obtained results with an increase of 0.49 dB for the signal to distortion ratio and 0.30 dB for the signal to interference ratio.

2.2.1.5 Combining Audio Features with Visual Features

There are a lot of cases where not only an audio signal is provided, but also a visual representation of the person who is talking or singing. Google's research group recently experimented with this idea. Their focus was not on extracting vocals from music tracks but rather on isolating voices within a crowd or a noisy environment [25].

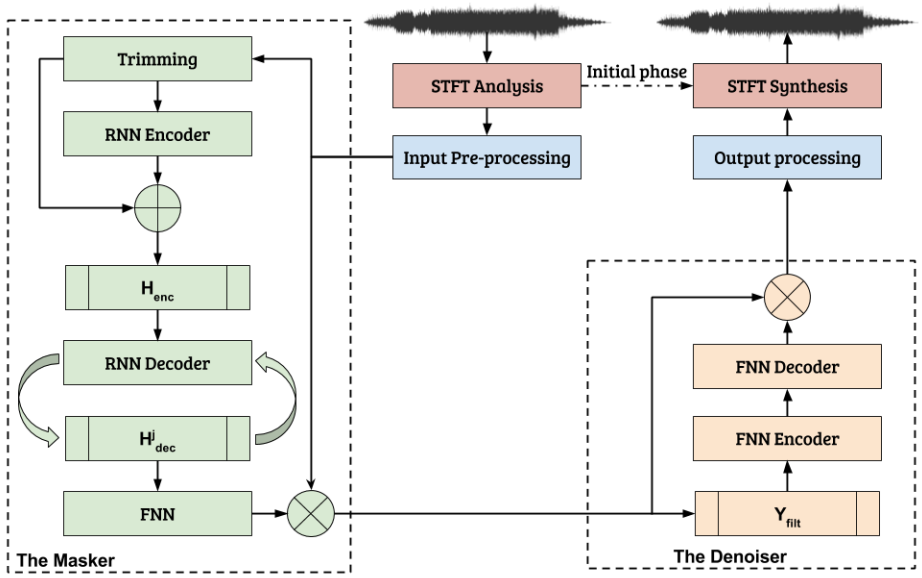

Figure 2.10: Proposed architecture by S. I. Mimilakis *et al.* [7]

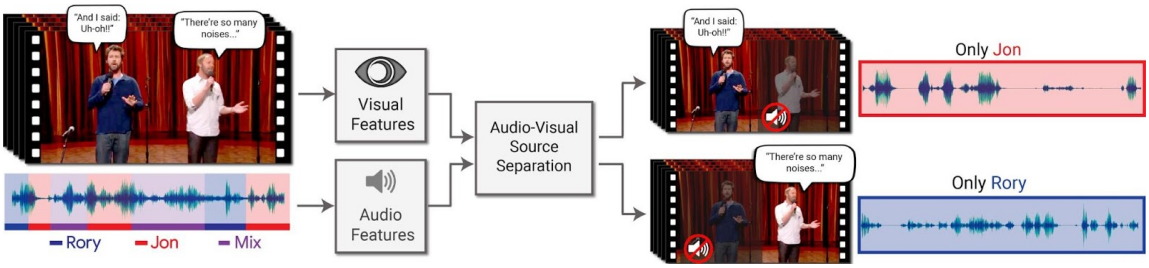

Figure 2.11: Visual explanation of Google's idea [8]

By looking closer to the model that was used in Figure 2.12, it can be seen that again a convolutional network has been used for both the visual and audio features. After the convolutional network, both streams are added together and subsequently passed through a bidirectional recurrent network. Finally, the masks can be obtained through the dense (or fully connected) layers, from which the spectrograms can be recalculated. For creating the spectrograms and reconstructing the wave file from the spectrogram the short-time Fourier transform (STFT) and the inverse STFT were used, respectively.

Google states that the visual component is the key here to make the predictions more accurate, because by watching a person's mouth, it is easier to identify who is talking and identify which voices to focus on.

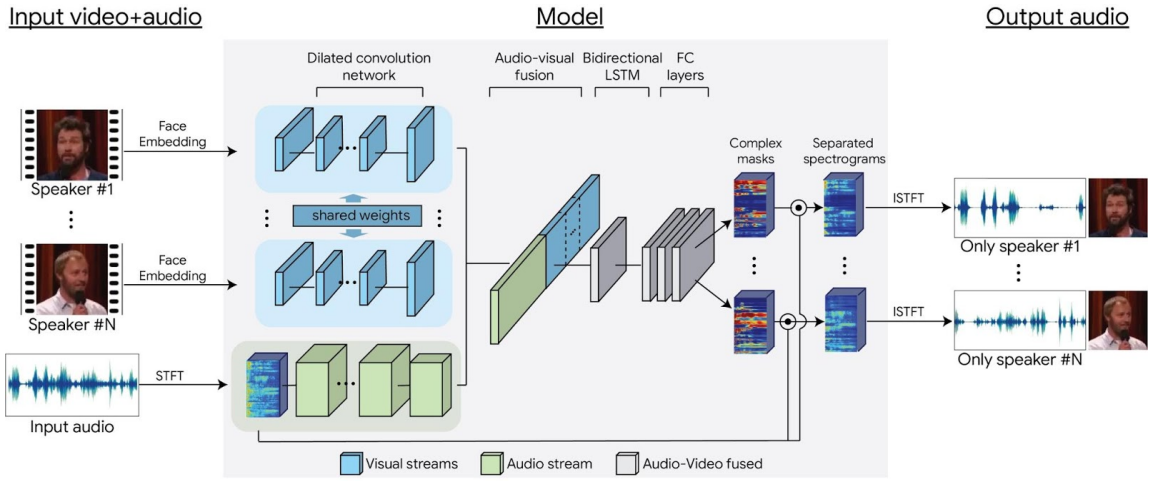

Figure 2.12: Proposed model by Google for combining both audio and visual features [8]

2.2.2 Other Techniques

In the next part, techniques other than neural networks are briefly discussed that have been used a lot in the past for audio source separation. As mentioned before, Huang *et al.* have shown that neural networks can already surpass these techniques a lot (table 2.3). Because of this reason, we will not look at them too much in detail.

2.2.2.1 (Robust) Non-Negative Matrix Factorization

Before deep learning was becoming more popular, Non-Negative Matrix Factorization (NMF) was one of the most used techniques for supervised source separation. The idea behind this approach is to represent a matrix $Y = BG$, where B is the basis vector and represents the frequency response of a source at a given time and G represents the gain of the frequency response at any point along the time axis. Thus, if we have two sources, S_1 and S_2, that are mixed into the mixture Y, i.e. $Y = S_1 + S_2$, and we know that the basis vectors for S_1 and S_2 are computed as B_1 and B_2, then we can write the mixture Y as follows:

$$Y = B_1G_1 + B_2G_2 \tag{2.15}$$

where G_1 and G_2 are the activation gains for the sources S_1 and S_2, respectively. One thing to keep in mind is that NMF assumes that the mixture can be represented as a linear combination of the basis dictionaries. The example in equation

2.15 can be generalized for multiple sources:

$$Y_{i,j} = \sum_{k=1}^{K} B_{i,k} G_{k,j} \tag{2.16}$$

where K is the amount of sources.

Sprechmann *et al.* [20] proposed a more robust version of this approach, i.e. *robust* NMF (RNMF). This model extends the standard NMF to consider the low rank and the outlier terms to be non-negative.

Despite the fact that the results from deep neural network approaches are already a lot better than when a NMF approach is used, NMF can still be very useful. Grais *et al.* [26] propose a deep neural network (DNN) for single channel source separation where NMF is used to initialize the DNN estimate for each source. The results of their experiments show that using a DNN initialized by NMF improves the quality of the separated signal compared with using only NMF for source separation.

2.2.2.2 (Robust) Principal Component Analysis

Compared to the techniques that were discussed earlier, principal component analysis (PCA) is an unsupervised approach, which means that it learns without knowing the groundtruth of the training data. PCA is a widely used technique for dimensionality reduction. However, its performance can be very sensitive to outliers. To remedy this issue, a technique has been developed called robust PCA (RPCA) [17, 18, 27]. The algorithm decomposes a spectrogram X into a low-rank matrix A plus a sparse matrix $X - A$ and can be formulated as follows:

$$\min_{A} ||A||_* + \lambda ||X - A||_1 \tag{2.17}$$

where $||A||_* = tr(\sqrt{A^T A})$ is the trace norm of A, $||X - A||_1$ is the entrywise l_1-norm, and λ is a positive constant which can be set to $\frac{1}{\sqrt{max(m,n)}}$.

The l_1-norm is used to characterize sparse corruptions and thus makes it more robust against outliers.

2.3 Datasets

There are a number of datasets available that have been created for research on music information retrieval (MIR) tasks. In this section, several datasets will be discussed and a comparison of their advantages and disadvantages will be made.

2.3.1 MIR-1K

The MIR-1K dataset [28] was created especially for singing voice separation and contains 1000 song clips. The vocals and instrumental accompaniment are recorded at the right and left channel, respectively. The duration of the clips can have a variable length from 4 up to 13 seconds. The songs are Chinese pop songs and sung by 8 female and 11 male singers. Most of the singers are amateur and do not have professional music training. The songs have a sampling rate of 16000Hz so it is perfect for small experiments and to do research on, but for experimenting with 'real-world examples', the dataset is not representative.

2.3.2 iKala

The iKala dataset [14] is currently the most used dataset for source separation in audio. A lot of papers that were discussed in the previous section [3, 13, 18, 27] used this dataset for their experiments. This has the advantage that the metrics of a new model can be more easily compared with the state of the art.

This dataset contains 252 clips of 30 seconds each. The same principle is used as for the MIR-1K dataset, i.e. the voice is recorded at the right channel and the music accompaniment at the left channel. An advantage from this dataset compared to the previous one is that the tracks now have a fixed length. This makes it more easy to work with. Another advantage is that the sampling rate is now higher (44100Hz) so a model that is trained on this data could then be tested more easily on songs that we hear on the radio for example. The iKala dataset is not publicly available, but can be requested by signing a license agreement. A download link is then sent by e-mail within five working days.

2.3.3 DSD100

The demixing secrets dataset (DSD) [29] is a multi-track dataset, which means that it not only contains a groundtruth for the vocals, but also for every instrument individually. This makes it possible to go further than only isolating the voice

from a music track. The set consists of 100 tracks and the supported targets are 'vocals', 'accompaniment', 'drums', 'bass' and 'other'. The sampling rate of the songs is 44100Hz.

2.3.4 MedleyDB

Another example of a multi-track audio dataset is MedleyDB [30]. It contains 122 songs, each with a sampling rate of 44100Hz. The duration of the songs is mostly around 4 minutes, but there are a few excerpts that are about 7 hours in total. MedleyDB consists of different genres, a.o. singer/songwriter, classical, rock, jazz, pop, rap, etc.

Dataset	**Contents**	**Hours in total**	**Metadata**	**Sampling rate**
MIR-1K	1000 excerpts	2.22	Vocals and background	16000Hz
iKala	252 excerpts (30s)	2.10	Vocals and background	44100Hz
DSD100	100 songs	±6.5	Multitrack	44100Hz
MedleyDB	122 songs	±14.2	Multitrack	44100Hz

Table 2.4: Comparison datasets

By comparing these datasets, there are huge differences that can be noticed. First of all, the sampling rate from MIR-1K is much lower than from the other datasets. Not only the sampling rate is low, but also the quality of how the recordings were made is not optimal, because it was made by amateur singers. This is one of the main reasons this dataset has not been used for our model. Secondly, we notice a big difference in total hours when we compare the MIR-1K and iKala with the two multitrack datasets. Despite the fact that the recordings from the DSD100 and MedleyDB datasets sound much better than the tracks from iKala and contain much more data, the choice was made to use iKala for our experiments. First of all because a lot of the approaches from the state of the art used this dataset, which makes it easy to compare our approach with. Secondly, this thesis focuses on the constant-Q transformation which causes the transformed training data to be much larger than when the standard STFT is used. This already makes it harder to load everything into memory. Using an even larger dataset would probably overload the machine.

Chapter 3

Methodology & Implementation

3.1 Proposed framework

The framework consists of three different main parts. First of all, the dataset needs to be preprocessed before it can be used for the training of the model. As discussed earlier, the music signals should be transformed into a two-dimensional space, such that all the different frequencies can be distinguished from each other at every time instance. This can be done by using a short-time Fourier transformation (STFT) or a constant-Q transformation (CQT). After this process, these transformations are used to create the ideal masks needed to reconstruct the original source's spectrograms. Then the transformed dataset should be split into a training set, a developer set and a testing set. The second part will be the training of the model. By feeding the training data to the model in small batches, the network learns how to distinguish the vocals from the instruments. The developer set can then be used to finetune the hyperparameters to make sure that the model performs as good as possible when unseen data is fed to the network. Finally, the trained model can be used to predict unseen batches of data (from the test set), and for all these estimated sources, the evaluation metrics can be calculated. In the following sections, these three main steps will be discussed.

3.1.1 Preprocessing

Since this master's thesis mostly focuses on the comparison between STFT and CQT, there is a preprocessing script provided for both these transformations. For loading the wave files and transforming them, the *Librosa* [31] library is used. In order to get the magnitudes, the absolute value is taken from the spectrograms. This way, we get rid of the complex values and the phase information is thrown

away. However, if the magnitude spectrograms would be used for training (in case only the groundtruth of one source is fed to the network), it would not be possible to recover the full spectrograms afterwards, i.e. the spectrograms with complex values which contain both amplitude and phase information. To solve this problem, a mask is created for every song. As already discussed in section 2.1 (equation 2.1), this mask represents the ratio between the power spectrogram of one source, e.g. the vocals, and the sum of the powerspectra of all the sources from the mix. With this mask the full spectrogram can be recreated by multiplying it with the spectrogram of the mix (equation 2.2). This way, both the amplitude and phase information are regained from the input information. For the final model, the samples from the dataset are not downsampled and kept at their original sampling rate, i.e. 44100Hz.

Furthermore, there is a third preprocessing script which basically just splits the left and right channel from the original mix, which contain the voice and instruments respectively. These represent the original sources and can be used for subjective testing. A comparison can then be made with the estimated sources.

As will be explained in section 3.2, every sample is split into three chunks of 10 seconds each. This can be easily done with the *Librosa* library.

Before we can start training the network, the data needs to be split into a separate training set, developer set, and testing set. That way, we can make sure that a certain subset of the data remains untouched and we can evaluate the model using the data that the network has not seen yet. For this, the 3 chunks that were extracted from one sample are always contained within one set (this is only applicable for the final deep U-net model which will be explained in section 3.2.3).

3.1.2 Training the model

Before actually training the model, a configuration file can be adjusted and the following parameters can be changed:

- *epochs*: the amount of times the model needs to go over the full training set
- *epoch_save_checkpoint*: after how many epochs a checkpoint is saved
- *learning_rate*: learning rate of the Adam optimizer
- *dropout_rate*: the probability that a unit (neuron) is dropped out of the network

- *batch_size*: the size of the batches that are used for training based on mini-batch gradient descent-like optimization
- *loss_type*: the loss function that needs to be minimized
- *alpha*: factor in the loss function (equation 3.6). The higher alpha, the more the model focuses on the loss of the vocals. The lower alpha, the more the model focuses on the loss of the instruments ($0.0 \leqslant alpha \leqslant 1.0$). This parameter is only used for the U-net model.
- *transf*: the used transformation (STFT or CQT)
- *trainset, devset and testset*: the path to the location of the data

When the network is learning, it writes data to an *events*-file every epoch. This file can be loaded into *TensorBoard* and the course on the loss curves for both the training and developer set can be followed. This way, it can be monitored when the model starts overfitting, i.e. when the loss on the developer set stops reducing. The decision of when to stop training the model was made manually after an analysis was made of the final loss curves. Every X amount of epochs (see *epoch_save_checkpoint* parameter in the configuration file), a checkpoint is saved to the machine. After training, these checkpoints can then be used for testing the model. Another useful feature of *TensorBoard* is that it makes it possible for the user to see a graphical representation of the model that is implemented.

3.1.3 Testing the model

The model can be tested in different ways. First of all, the test set will be fed into the model. If the model is already trained, the framework will load the most recent checkpoint to make predictions on the test set. If it would be the case that the model was already overfitting at this point (which can be monitored using TensorBoard), it is possible to use a checkpoint from earlier in time. This can be done by adjusting the epoch in the checkpoints file. During the testing process, the evaluation metrics (SDR, SIR, and SAR) are stored per predicted test sample. If all samples are processed, a boxplot is automatically generated and stored in PNG format to evaluate the results.

Another possibility to test the model and its predictions is to make use of subjective testing. For this approach, we do not look at the strict evaluation metrics that were calculated for every sample, but rather test how much the predictions are

appreciated by the people. Since this is not a part of the framework, this approach will be discussed more in detail in section 4.3.

3.2 Experimentation

In this section, the different steps of the progress are explained and the evolution of the used models are discussed.

3.2.1 Comparison Convolutional and Recurrent Network

The first small experiment was to implement a simple network and see if it was able to learn a very small training set by heart. This is a first step to be sure that the model is capable of learning the data. The next step is then to check if the network is capable of learning new data that it has not seen before. Both a convolutional and a recurrent network were implemented. For these experiments, the iKala dataset was used which contains audio files of 30 seconds each and a sampling rate of 44100Hz. Since this was only a small first test, all the files were downsampled to 16000Hz to reduce the computation. Also only short-time Fourier transformations were used during preprocessing for these experiments. It soon became clear that the recurrent neural network needs a lot more computation time than the convolutional one and thus converges much slower. This was a first reason to give more attention to convolutional networks. A second reason for focusing more on convolutions is that recurrent networks are known to incorporate feedback. For the audio source separation problem, this feedback can have the impact that the model is less stable when tested under novel conditions that were not seen during the training process [32]. Also by using convolutional neural networks, smaller, robust features of the timbres can be learned across smaller subsections of the spectrogram.

3.2.2 Deep Convolutional Network

After the decision was made to focus on convolutional neural networks, the simple model with only two convolutional layers was transformed into a deeper network. Several experiments were done to improve the model and its predictions. Also for these experiments, the samples from the dataset were kept at their original sampling rate, i.e. 44100Hz. This way, the trained model could also be tested on commercial songs which have almost always this sampling rate (CD quality).

The first version of the model contained only convolutional layers and at the end one dense layer. Several experiments were done using different channel sizes for each layer. One approach was to increase the channel size up until the last layer, another test was to first increase the channel size for a few layers and then gradually decrease the output channels again. As the latter gave more promising results, this was the model to focus on. In the following section the evolution of this model will be discussed as well as the improvements that were made.

3.2.2.1 Evolution of the model

As can be seen in figure 3.1, the loss curve on the developer set is going down at the beginning but compared to the training loss, its minimum is still way too high and tends to go up again really fast. To improve this loss, different methods can be applied. First of all, we should search for an optimal size for both the train and test sets. In case the test set is small and there is one (or more) outlier(s), the evaluation metrics will not be representative. Several ratios were tested for splitting the dataset into train and test sets and then a comparison was made in order to find the best division. When splitting the dataset into two equal partitions for both the training set and the two evaluation sets (one for developing and one for testing), the minimum from the loss curve on the developer set went from a RMSE of 0.39 to about 0.37 (Figure 3.2). The loss function that was used for this model is defined in equation 3.3.

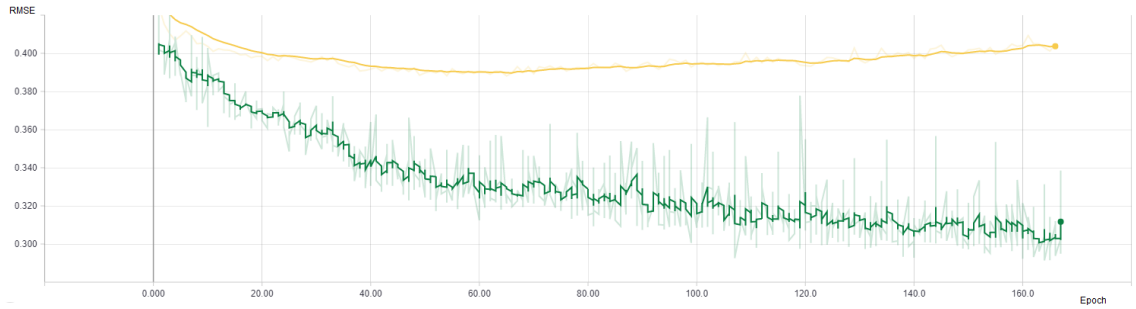

Figure 3.1: First RMSE loss curves (training loss: green, evaluating loss: yellow)

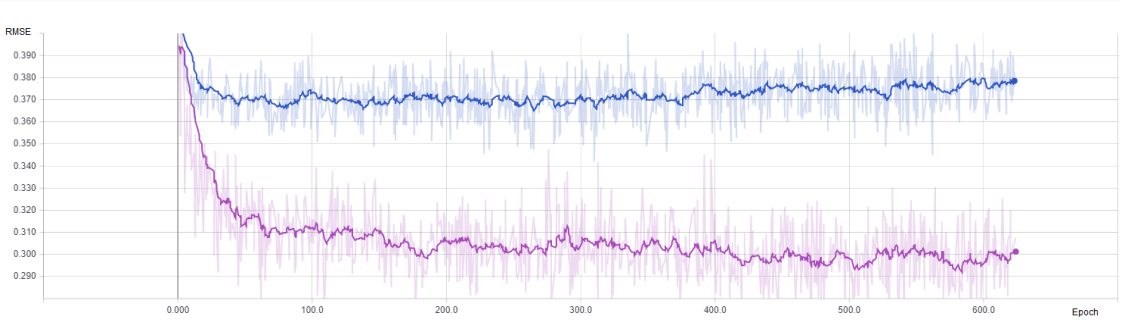

Figure 3.2: RMSE Loss curves after making the evaluation set bigger (training loss: purple, evaluating loss: blue)

As can be inferred from figure 3.2, the model is still not doing well when it tries to predict unseen data. There are still some options to enhance the predictions. First of all the data can be augmented, because the more data, the more the network can learn. By slicing every audio file into different chunks and slightly allowing some overlap, we could improve the model. Another possibility is to add several dropout layers [33] which is a regularization technique that will make sure that the model does not overfit too fast on the training set. As shown in figure 3.3, it can be clearly seen that the developer loss tends to go lower when a higher dropout rate is used. When looking at the curves for the training loss, indeed the model does learn much slower when the dropout rate is higher. In other words, the higher the dropout rate, the more the loss for the train and developer set tends to become closer to each other.

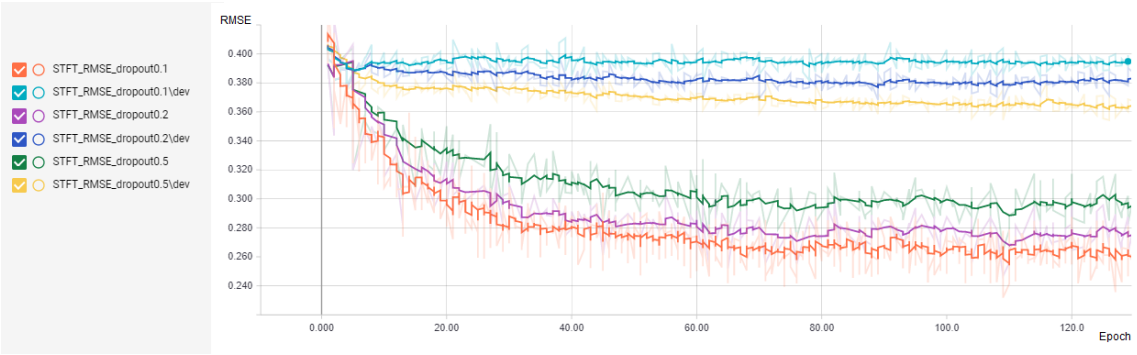

Figure 3.3: Comparison RMSE loss curves for different dropout rates

3.2.2.2 The model

The final version of this model consists of seven convolution layers, each followed by a dropout layer and is shown in figure 3.5. The layer after the last dropout

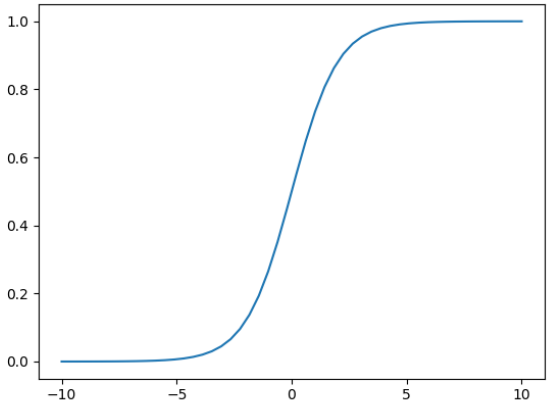

Figure 3.4: Sigmoid function

layer is a fully connected layer which predicts the final output in the same format as the input. For the convolutional layers a filter was used of three by three and the output channels have the following order: 16, 32, 64, 32, 16, 16, 1. Every convolution is followed by a rectified linear unit (ReLu) activation function (see figure 3.6). This function keeps all values that are positive, but sets all negative values equal to 0. The biggest advantage of this activation function is that it works better for gradient propagation. It has fewer vanishing gradient problems if we compare it to e.g. sigmoidal activation functions. The ReLu activation function can be expressed as follows:

$$ReLu(x) = x^+ = max(0, x) \tag{3.1}$$

For the dense layer a sigmoid activation function is used which is visualized in Figure 3.4 and defined as follows:

$$Sigmoid(x) = \frac{1}{1 - \mathrm{e}^{-x}} \tag{3.2}$$

What this formula actually does, is transforming values between $-\infty$ and ∞ into values that are in the interval $[0, 1]$. Whenever x is a big positive value, e^{-x} will become very close to 0 so the output of the sigmoid function will be (close to) 1. If x is a very small value (negative), e^{-x} will be much higher, which causes the output to be 1 divided by a very big number. The output will thus be much closer to 0 in this case.

For all the above figures where the losses are visualized, the root mean square

error (RMSE) is used as a loss function. Another possibility would have been to simply use the mean square error (MSE). This is a more common alternative which makes the additional square root unnecessary. However, for all our experiments with both this model and the U-net model, the RMSE was used.

This deep convolutional model only predicts the isolated vocals, which implies that the loss is only based on this prediction. We can formulate the loss function as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_{voice,i} - \hat{Y}_{voice,i})^2} \tag{3.3}$$

with Y_{voice} the target value for the voice and $\hat{Y}_{voice}$ the value that is predicted by the model.

To optimize this loss, the Adam optimizer is used. Adam is a derivation from 'adaptive moment estimation'. The Adam algorithmn tries to combine the best properties of two other optimization algorithms, i.e. Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp). These are both extensions of the classical stochastic gradient descent. AdaGrad maintains a per-parameter learning rate and improves the performance on problems with sparse gradients such as natural language problems. RMSProp also maintains per-parameter learning rates but here, they are adapted based on the average of recent magnitudes of the gradients for the weight. The advantage of RMSProp is that it works well for online and non-stationary problems. The Adam optimizer only requires first-order gradients with little memory requirement and computes individual adaptive learning rates for different parameters from estimates of the first and second moments of the gradients. The decaying averages of past and past squared gradients (m_t and v_t, respectively) are computed as follows:

$$m_t = \beta_1 m_{t-1} + (1 - \beta) g_t \tag{3.4}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta) g_t^2 \tag{3.5}$$

where g_t denotes the gradient at time step t. m_t represents the estimate of the first moment (the mean) of the gradients and v_t represents the second moment (the uncentered variance).
The Adam optimizer was introduced by Kingma *et al.* [34]. They made a comparison with other optimization algorithms and showed that this optimizer has a

lot of advantages. Adam can handle sparse gradients on noisy problems very well and it is relatively easy to configure. The default configuration parameters are recommended for most problems so for the experiments with the deep convolutional network, these were left untouched (learning_rate=0.001, beta1=0.9, beta2=0.999, epsilon=1e-08).

3.2.3 Deep U-net

Despite the improvements that were made for the deep convolutional neural network to reduce the loss, the predictions on the unseen data with that model were still not performing well on unseen data, given the evaluating loss compared to the training loss. There was need for a better model that actually learned some useful features. Based on the model of A. Jansson *et al.* [3], a U-shaped network was implemented which builds upon a fully convolutional network and is similar to the deconvolutional network [15]. In a deconvolutional network, a stack of convolutional layers encodes the image (in this case the mask of a source) into a small and deep representation. In each convolutional layer, the size of the image is halved (by using strides of 2x2) and the output channels are doubled. In the second part of the network, the encoded image is then decoded again back to the original size of the input by a stack of upsampling layers.

3.2.3.1 Evolution of the model

The initial U-net model was implemented with four convolutional layers and four deconvolutional layers. For the convolutions, the channel size started from 16 and was doubled every next layer up till 128. For the deconvolutions, the same channel sizes were used but in reversed order. Before every deconvolution (except the first one) a skip-connection is made such that the output of the convolutional layer and the output of their corresponding deconvolutional layer can be concatenated and fed into the next layer. This allows low-level information to flow directly from the high-resolution input to the high-resolution output. All the lessons learned from the previous model, such as dropout, the use of ReLu activation functions, and equally divided train and evaluation sets, were now again applied to the model.

During the experiments with this model, leaky ReLu activations were used as well for some layers with a default leak of 0.2. They allow a small, positive gradient when the unit is not active. They are only used in the convolutional layers. For the deconvolutional layers, normal ReLu's are used. Adding these leaky ReLu activations had a significant impact on the predictions.

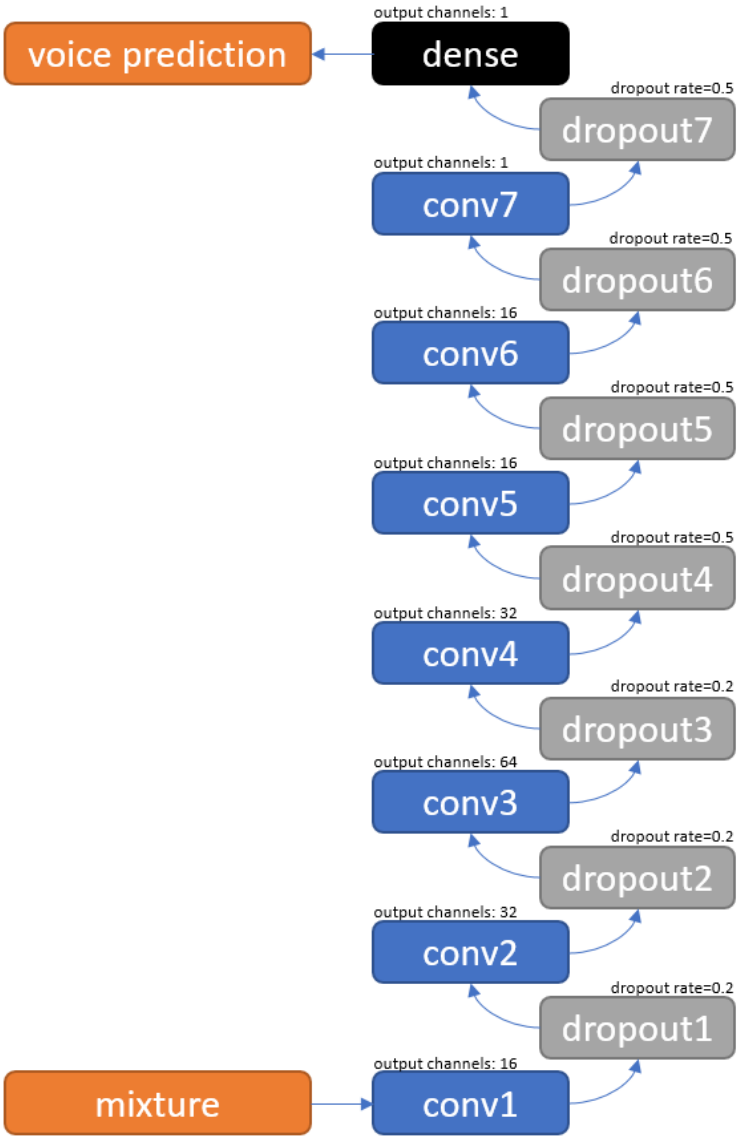

Figure 3.5: Proposed deep convolutional network architecture.

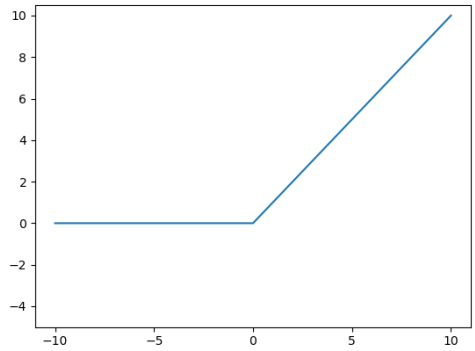

Figure 3.6: ReLu function

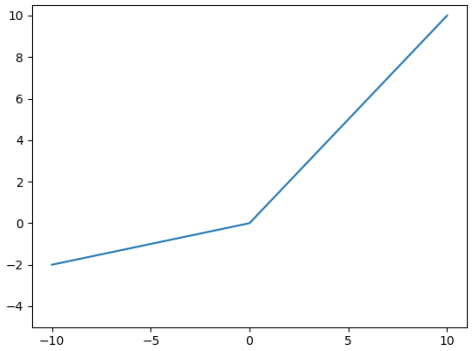

Figure 3.7: Leaky ReLu function

Finally, batch normalization is added to the model which also acts as a regularizer (sometimes this approach even eliminates the need for dropout, but in this case a combination of both is used). Batch normalization allows us to use higher learning rates and be less careful about initialization. It allows each layer of the network to learn more by itself and make it more independent of the other layers [35].

As discussed in the section of the previous model (section 3.2.2.1), we introduced some overlap in the preprocessed data to create a bigger dataset. With the U-net model, this augmented data could still be used for the experiments with the STFT, but for the CQT experiments, this was impossible because of the much bigger file sizes. To make sure the machine did not go out of memory, the overlapping parts were removed again from the dataset for these experiments, and the batch size was reduced from 10 to 8. Nevertheless, the curves for the developer loss compared to the training loss for this model were still a lot more promising than for the deep convolutional network.

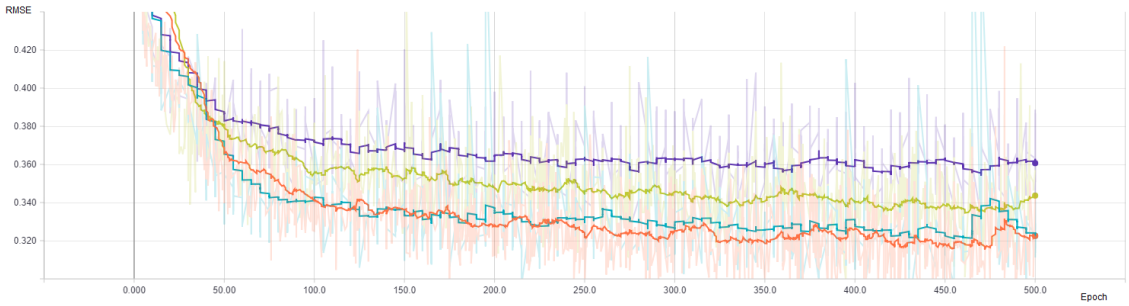

Figure 3.8: Loss curves for the U-net model (STFT training loss: green, STFT evaluating loss: purple, CQT training loss: orange, CQT evaluating loss: blue)

By looking at the loss curves in figure 3.8, a few things stand out. First of all, it can be noticed that the CQT training loss decreases to a RMSE of 0.32 while the STFT training loss only decreases to a value of 0.34. A second, and maybe more important aspect, is that the loss on the developer set for the CQT follows almost exactly the same trend as the CQT training loss, whereas for the STFT experiment, there is a big gap between the training and developer loss. By looking at the losses on the 500th epoch, the STFT developer loss has decreased to a RMSE of only 0.36, while the CQT developer loss is close to 0.32.

3.2.3.2 The model

For the final version of this model, an extra convolutional and deconvolutional layer were added to the network, both with an output channel of size 256 (again the channel size is doubled from the previous layer). Compared to the deep convolutional model, the final layer now gives an output with two channels, one for the voice prediction and one for the instruments prediction. In this case, both predictions are used in the loss function. A parameter α can be adjusted in the configuration file to emphasis more on the voice loss or the instruments loss. By default both the voice and instruments loss are threated equally, so this value is set to 0.5. The loss function is defined as follows:

$$L_{total} = \sqrt{\alpha \cdot MSE_{voice} + (1 - \alpha) \cdot MSE_{instruments}} \tag{3.6}$$

where MSE is the mean squared error for each source:

$$MSE_{source} = \frac{1}{n} \sum_{i=1}^{n} (Y_{source,i} - \hat{Y}_{source,i})^2 \tag{3.7}$$

A graphical representation of this loss function can also be seen below in figure 3.10. Again, the same algorithm was used as for the previous model to optimize this loss, i.e. AdamOptimizer. A visualization of the model itself is shown in figure 3.9.

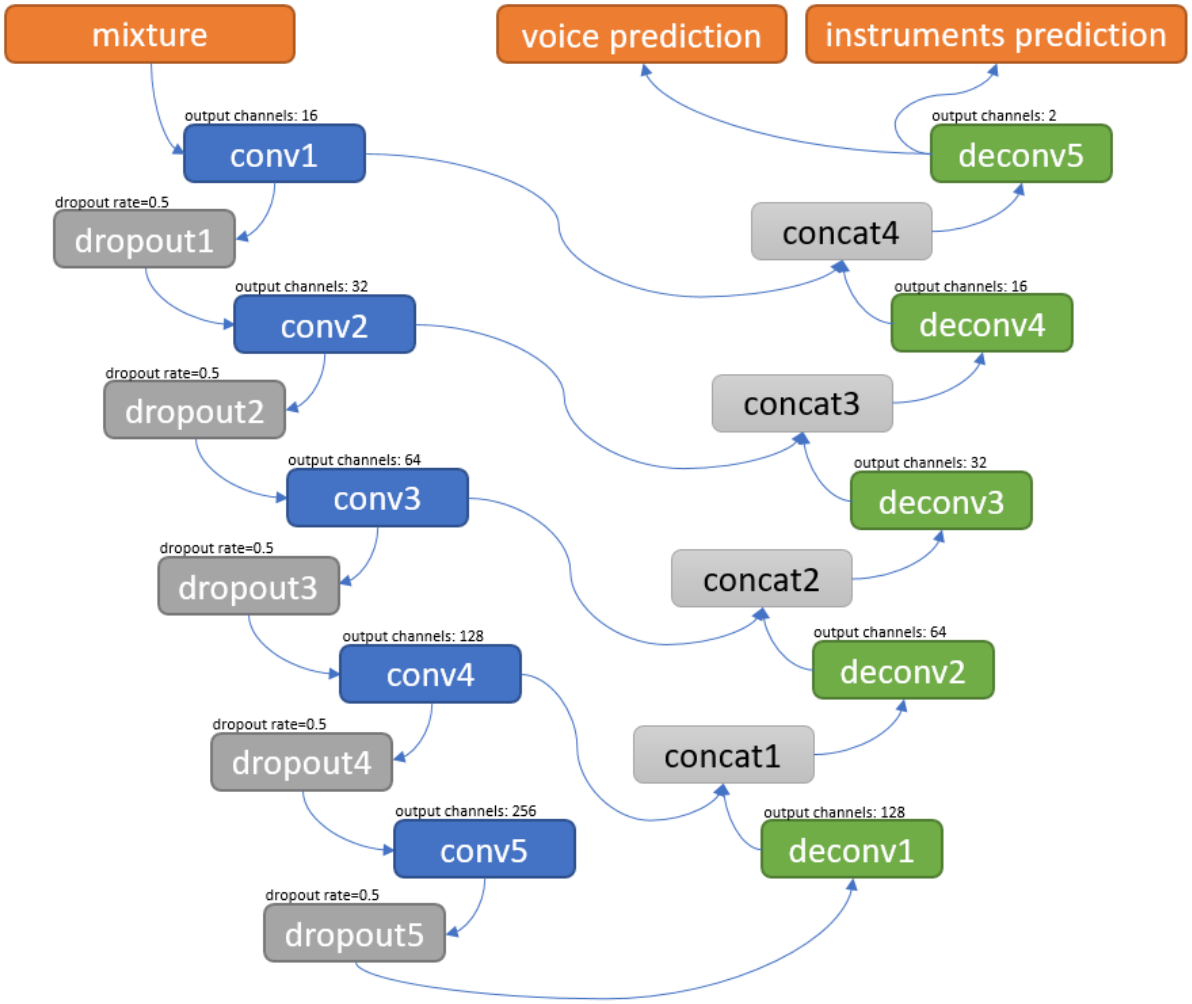

Figure 3.9: Proposed deep U-net convolutional network architecture.

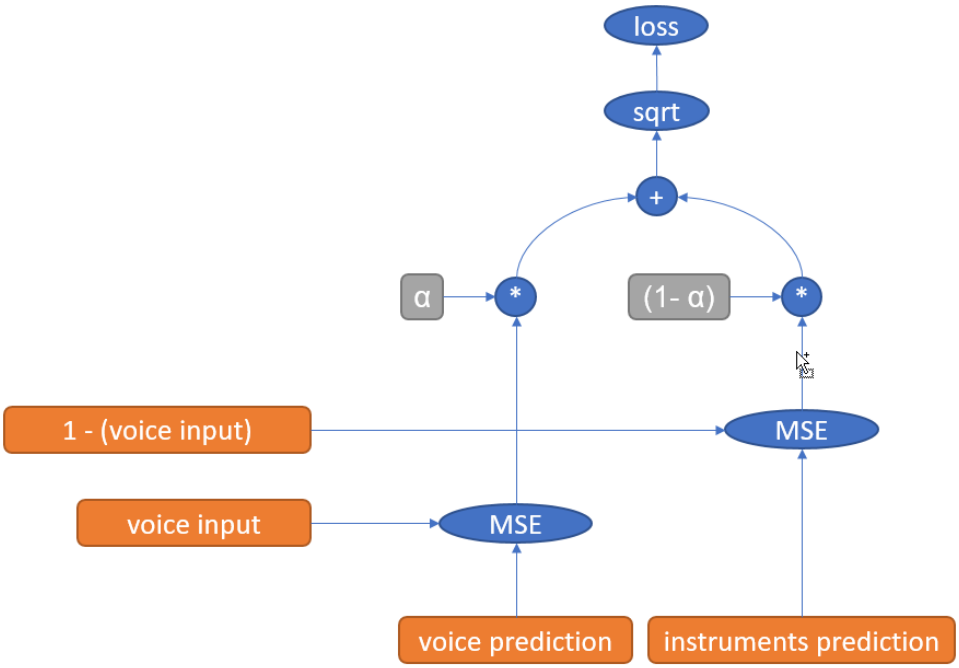

Figure 3.10: Loss function for U-net convolutional network architecture.

Chapter 4

Evaluation

4.1 Results

During the experimentation, the *mir_eval* library [36] was used for calculating the evaluation metrics SDR, SIR and SAR. These were calculated for every file in the test set from which then the average was taken to compare it with the state of the art. However, only using the average can sometimes give a wrong impression if there are a lot of outliers. Because of this reason, boxplots were generated as well to get a more general idea of the metrics. Since the predictions of the deep convolutional model were not acceptable enough compared to the state of the art, the evaluation metrics were only calculated for the predictions of the second model, i.e. the deep U-net convolutional model. As can be seen in both the boxplots for the STFT (figure 4.1) and CQT experiments (figure 4.2), the metrics are very similar. This can also be noticed in table 4.1. We could say that the SIR and SAR scores for the isolated instruments using STFT are a little bit better than when using CQT, but the difference is very small and could differ when using another test set.
Something else that can be noticed in the boxplots is that the predictions from the STFT experiment contain more outliers than the ones from the CQT experiment. Both experiments were tested on the same testing set, so we can imply that the model is more consistent in its predictions when using CQT. The same evaluation metrics were calculated on the developer set and indeed, the same findings could be made. There were a lot of (bad) outliers for the vocals using STFT, a lot of (good) outliers for the instruments using STFT and only a few outliers using CQT. And even if there are outliers when using CQT, they are mostly close to the mean, compared to the big gaps that can be noticed between them in the STFT predictions.

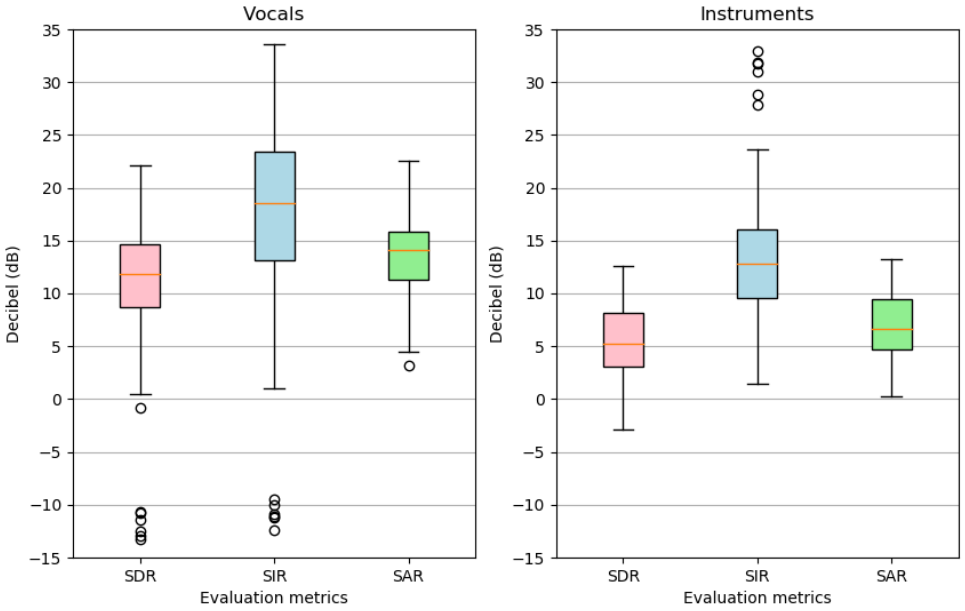

Figure 4.1: Boxplot of metrics for deep U-Net (using STFT)

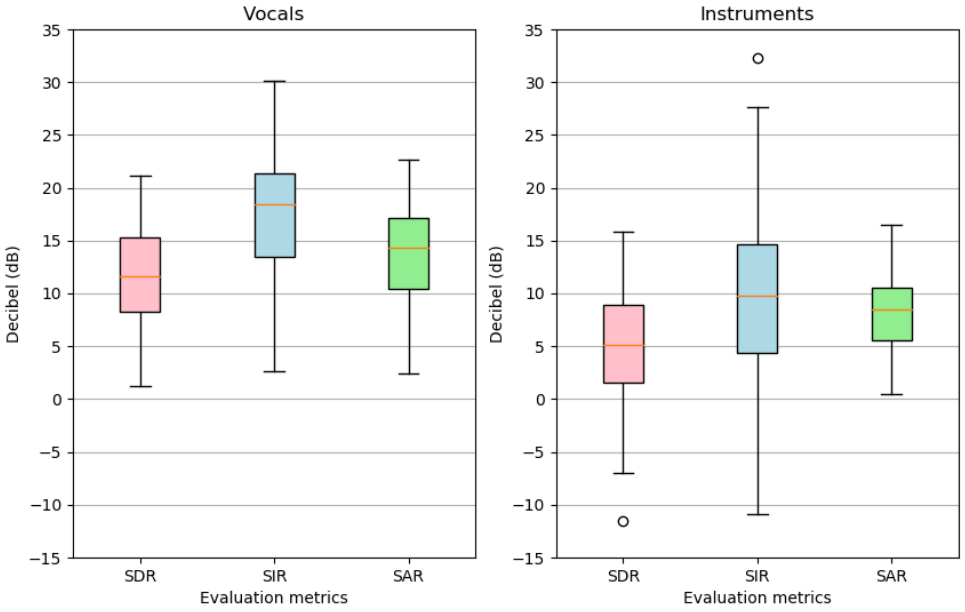

Figure 4.2: Boxplots of metrics for deep U-Net (using CQT)

		Deep Convolutional U-Net	
		STFT	**CQT**
vocals	**SDR**	11.04	11.38
	SIR	17.65	17.23
	SAR	13.62	13.64
instruments	**SDR**	5.34	5.03
	SIR	13.06	10.06
	SAR	6.84	8.30

Table 4.1: Comparison of metrics (average) for both the STFT and CQT experiments

4.2 Comparison with state of the art

If we take a look at the results of the evaluation metrics and compare it with the state of the art, it can be noticed that the deep U-net achieves much better results for the isolation of the vocals than the proposed recurrent network by Huang *et al.* [5] when comparing the metrics from table 4.1 with table 2.3. The deep U-net also surpasses the models based on RPCA [17, 18] and RNMF [20] easily. However, comparing it with the results from Jansson *et al.* [3] (Table 2.2), is more difficult. The results for the predictions on the vocals are close to each other when looking at the signal to distortion ratio. The predictions on the instruments are a lot better when their model is used and also the SIR and SAR metrics are much higher in general. The reason for this difference is probably because they had the resources to feed both voice and instruments data into the network, which makes it for them possible to train the voice and instruments predictions separately. In our approach, we had to find a way not to overload the machine and thus only make a separation in the last layer of our model. However, the main goal of this thesis is not so much to surpass these highly trained models, but rather to get better insights in the use of constant-Q transformations for this purpose, as stated before. Our deep U-net does however achieve better SDR results for the vocal predictions than their baseline model.

Evaluating the predictions based on the results of the metrics is a first good step and already gives some insights into the performance of the model for both the STFT and CQT approach. The second step is to listen to the results and try to observe the differences. What can be easily noticed is that the predictions for

songs with a lot of guitars and heavy arrangement perform very good when using CQT. What sticks in both approaches are the drums. These are very hard to get rid of in the vocal predictions. The bass on the other hand is almost always isolated completely. This is relatively easy to declare, since these frequencies are at a whole other level than the vocals.

A possible way to improve the predictions for the vocals could be to do some preprocessing on the input files. E.g. equalizing the songs and applying some filtering techniques could already bring the drums more to the background of the track. This preprocessing would probably do the trick to obtain higher quality predictions.

4.3 Subjective testing

Evaluating the predictions of the deep U-net for the STFT and CQT experiments by only comparing the evaluation metrics (SDR, SIR, SAR) does give an idea of how well both approaches perform, but the human hearing is quite complex and can be more sensitive to some frequencies than to others. To get a better idea of how well the model predicts on the STFT data and the CQT data, some subjective tests were created and a group of about 20 people were asked to fill in these surveys.

From the final predictions of the U-net model, 18 samples were chosen at random. These were separated into two parts (part A and part B) each using 9 samples of the set. Each of these parts was then again separated into 4 different parts:

- Predictions on the vocals using STFT
- Predictions on the vocals using CQT
- Predictions on the instruments using STFT
- Predictions on the instruments using CQT

Each part contained 9 questions where both the original mix as well as the prediction can be heard. For each part the same 9 samples were used in combination with their prediction. The respondents were asked to rate how well they thought the source (voice or instruments) was separated from the original mix on a scale from 0 (not isolated at all) to 6 (very well isolated).

If we take a look at figure 4.3, we see a boxplot for every set of scores that was given to the samples. From every sample, a weighted average was taken and added as a data point for the boxplot visualization. Something that is immediately noticeable is that the subjective scores of the STFT and CQT predictions for the vocals are quite comparable while the scores for the STFT predictions of the instruments are in general much lower than for the CQT predictions.

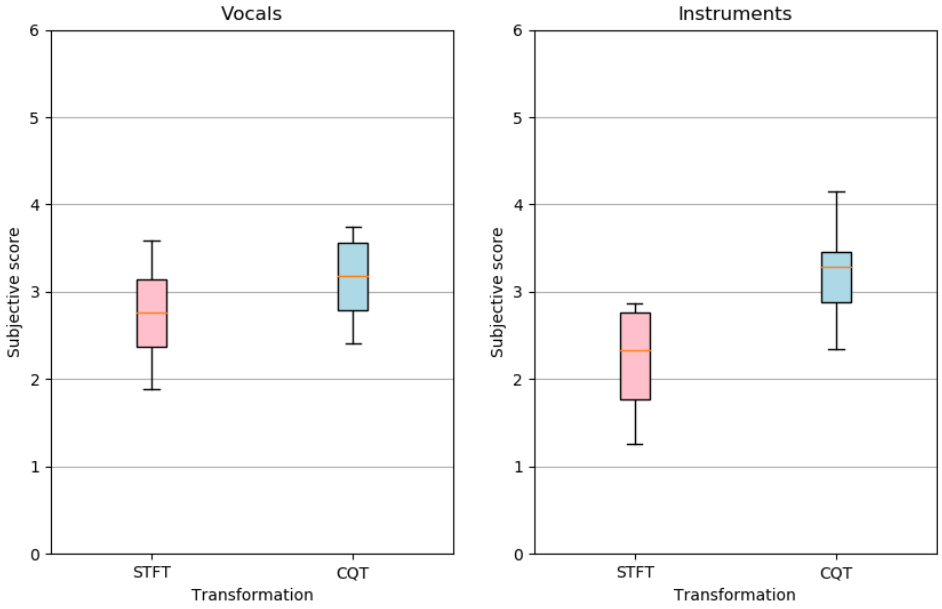

Figure 4.3: Boxplots of the subjective scores (comparison STFT and CQT)

Based on the subjective tests, we could infer that the CQT predictions are more preferred over the SFTF predictions, but let's have a closer look at the different samples. If we compare the weighted average for every STFT and CQT sample individually, we can see that in most cases the scores for the CQT samples are indeed better than for STFT. For the instruments predictions, this is very clear in figure 4.5. Figure 4.4 however shows that it was much harder for the listeners to determine which one of the voice predictions (using STFT or CQT) was better isolated from the original mix. Most of the time, the scores are very close together, but there are a few exceptions. E.g. the sample for question 6 gives a score of about 1.88 for the vocal prediction using STFT while the score for the CQT approach is 2.61. By listening to this sample, it can be noticed that for the STFT prediction, there are still some small parts of the bass that are not completely

isolated. This creates some extra noise which might be less pleasant for a person to listen to. For the CQT prediction of sample 6, the bass has (almost) completely become unnoticeable. The same observation can be made for the sample predictions of question 7, 14, and 18. For the other exceptions, there are only very small differences noticeable, and it is hard to say why people could have preferred the one over the other.

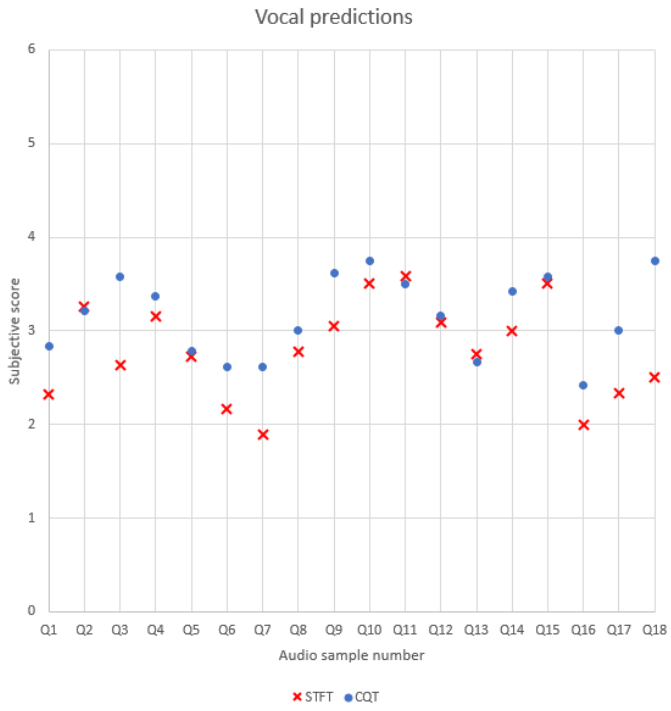

Figure 4.4: Weighted average of the subjective scores for every sample prediction of the vocals

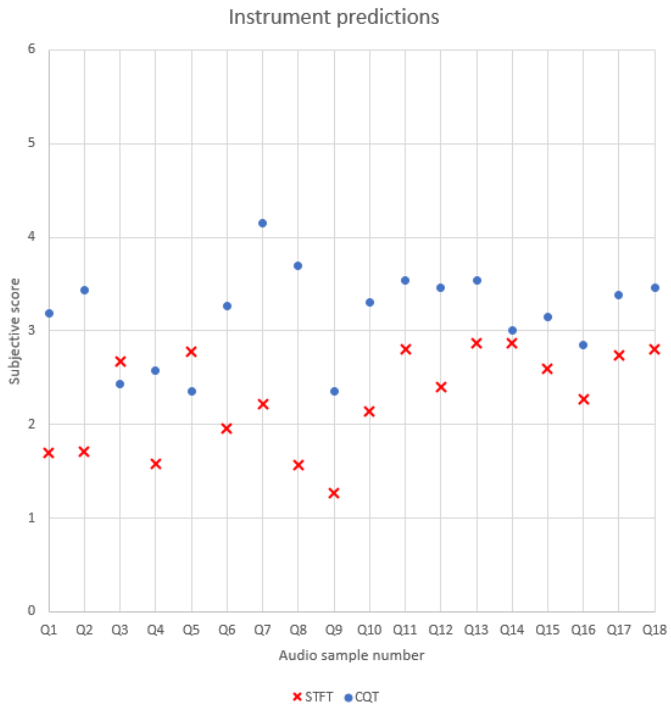

Figure 4.5: Weighted average of the subjective scores for every sample prediction of the instruments

If we look at figure 4.5, we see a much bigger gap between the subjective scores of the STFT and CQT approaches. CQT clearly outperforms the STFT in almost all the samples. By listening to the songs that were used for the tests, we notice that the CQT performs better than STFT if the sample contains guitars and a more heavy arrangement. The exceptions where STFT is more appreciated by the listeners than CQT are the samples of question 3 and 5. For these samples there are (almost) no guitars in the original mix and the main instrument is a piano in these cases.

			Subjective score (average)	
	Question	**Sex of singer**	**STFT**	**CQT**
Part A	**Q1**	Male	2.32	2.84
	Q2	Female	3.26	3.21
	Q3	Male	2.63	3.58
	Q4	Female	3.16	3.37
	Q5	Female	2.72	2.78
	Q6	Male	2.17	2.61
	Q7	Male	1.89	2.61
	Q8	Male	2.78	3.00
	Q9	Male	3.06	2.61
Part B	**Q10**	Male	3.50	3.75
	Q11	Female	3.58	3.50
	Q12	Female	3.08	3.17
	Q13	Female	2.75	2.67
	Q14	Female	3.00	3.42
	Q15	Female	3.50	3.58
	Q16	Male	2.00	2.42
	Q17	Male	2.33	3.00
	Q18	Male	2.50	3.75

Table 4.2: Subjective score (average) for the vocal prediction of every sample + sex of singer

Table 4.2 shows the average subjective scores for the voice prediction of every sample in more detail. This table also contains a column that shows the sex of the singer. With this data we can compare the average scores of the voice predictions for both male and female singers, and see if there are any differences (Table 4.3). Let us first focus on the STFT approach. By taking the average over all the scores from samples with a female singer, we get a score of 3.13. The average score for samples with a male singer is only 2.52. If we calculate the averages for the CQT

approach, we obtain a slightly higher score for the female voice predictions, i.e. 3.21, and for the predictions of the male singers 3.12, which is a great improvement compared to the STFT predictions. For both the STFT and CQT approaches, the predictions with a female singer get in general a higher rating than the ones with a male singer. A second observation is that the difference in score between the STFT and CQT approach for the female singers is only 0.08 which is not a significant difference, while the difference for samples with male singers is 0.6. This brings new insight when looking again at the plot in figure 4.4. If we take a look at the samples where the difference is bigger between the blue and red dots, these are indeed always samples with a male singer, with only one exception (the sample of question 14).

	Average score	
	STFT	**CQT**
Female	3.13	3.21
Male	2.52	3.12

Table 4.3: Average subjective scores: comparison male and female singers

Chapter 5

Conclusions

A lot of research has already been done in music information retrieval problems using deep neural networks. Currently, from most of the proposed models that try to solve the audio source separation problem, the focus is on the parameters that need to be tuned. Preprocessing is mostly done by using a standard short-time Fourier transformation (STFT) which divides the spectrogram in equally spaced frequency bins. The human hearing however works on a logarithmic scale, so this is where the idea of the constant-Q transformation (CQT) comes in. The CQT uses logarithmically spaced frequency bins and thus should mimic the human hearing in a better way.

During the experimentation, a deep convolutional neural network was used for this problem. Improvements were made by adding several aspects that are important in the field of deep learning such as dropout layers, batch normalization, the use of ReLu activation functions, etc. Furthermore we have learned that a better model was needed in order to get the network perform well on unseen data as well. For this, a deep convolutional U-net was implemented using the same ideas as the first model, but with deconvolutional layers added to the network. After analyzing the learning curves, we learned that the U-net managed to make much better predictions on unseen data than the deep convolutional model.

In the next step, the STFT and CQT have been compared by feeding the preprocessed data into the network and comparing the evaluation metrics, i.e. signal-to-distortion ratio (SDR), signal-to-interference ratio (SIR) and the signal-to-artifacts ratio (SAR). The metrics for both experiments were very similar and the differences between the means of the SDR, SIR and SAR were mostly not more than 1dB. However, when looking at the metrics more extensively, the predictions using

the STFT seemed to have more outliers than than the ones using CQT. In both cases the metrics for the vocal predictions scored significantly higher than the instruments predictions with differences of more than 5dB.

When evaluating the experiments based on human assessments, we gained some new insights from the subjective scores. While the evaluation metrics in the objective tests for both the STFT and CQT approach are in general much lower for the instruments than for the vocals, this difference is less noticeable in the subjective scores when looking at the CQT approach. In this case the average score for the instrument predictions is even a little bit higher than for the vocal predictions. For the objective testing, it was quite hard to compare the metrics for the STFT with the CQT, but if we look at the subjective scores of the instruments predictions, we see a clear difference between both approaches. The instrument predictions where the CQT is used clearly have a higher rating in general, especially when there are a lot of guitars in the song. For the vocal predictions it is less obvious, but the CQT still performs a little bit better than the STFT. Furthermore, we could also conclude from the subjective scores that in general, for the vocal predictions a higher rating was given with a female singer than to the ones with a male singer. This applies to both the STFT and CQT approach.

Chapter 6

Future work

The results of the predictions when using a constant-Q transformation instead of a short-time Fourier transformation are promising, so further research should be very interesting. Using the CQT works best for predictions of the instruments where a lot of guitars and a more heavy arrangement are included in the song. The voice predictions however still need some more improvements. The drums are still noticeable in every voice prediction, both for the CQT and STFT experiments. A possible way to improve the quality is to do some preprocessing on the data. Equalization and filtering would probably do the trick to obtain better results.

Right now, the deep U-net is only tested on the iKala dataset which means that either only the voice or the instruments can be predicted. In the future, this model should be tested on a multi-track dataset as well (e.g. DSD100 dataset) in order to see if the CQT helps to predict certain instruments better than others. The DSD100 dataset is also more representative than the iKala dataset if we compare it with the daily music we hear on the radio, because these songs are in English and the quality of how the recordings were made is much better (the sampling rate is however the same, i.e. 44100Hz).

While the instruments predictions could be used for karaoke-like applications, the voice predictions could be very useful as well. E.g. when looking at the automatic subtitle generation from YouTube, it can be noticed that their predictions are already very good when a speaker's pronunciation is correct and the sound quality is not too low. However, from the moment that there is some background noise or background music, their model fails to predict or just stops predicting. As discussed previously in section 2.2.1.5, Google is already working on a model that predicts voice based on both audio and visual aspects, from which they state that

the visual component is the key to predict more accurately. By first applying their methodology to the videos on YouTube, the automatic subtitle generation should improve as well. A last step is then to try combining this approach with the CQT instead of the standard STFT and see if the predictions can be improved even more.

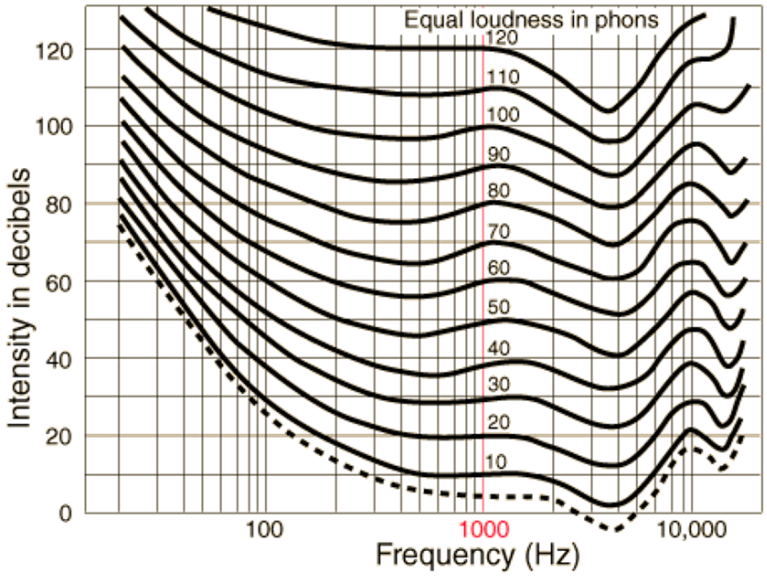

Figure 6.1: Equal loudness (Fletcher-Munson) curves [9]

Using a CQT for the transformations of the dataset gives a better representation of the human hearing than when the songs are transformed using a STFT, because CQT uses logarithmically spaced frequency bins and our hearing works on a logarithmic scale as well. Another possibility would be to find a way to transform the input and target data in the frequency domain to something that has a higher chance of leading to the best hearing experience. For this approach, we can make use of the so called *equal loudness curves* or *Fletcher-Munson curves* [9] as shown in figure 6.1, where the y-axis shows the measure of sound pressure (dB SPL) and the 'phon' is the unit of measurement for loudness level. The idea would be to focus more on the frequencies that the human ear is more sensible to. E.g. at 60 dB, we do not hear a sound at a frequency of 4kHz at the same loudness as a sound at a frequency of 30Hz. Something to keep in mind is that it would not be possible to evaluate this approach objectively and subjective tests will be needed.

Bibliography

[1] UFLDL, Stanford, "Convolutional neural network," [Online; accessed May 16, 2018]. [Online]. Available: http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/

[2] M. Espi, M. Fujimoto, K. Kinoshita, and T. Nakatani, "Exploiting spectro-temporal locality in deep learning based acoustic event detection," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2015, no. 1, p. 26, 2015.

[3] A. Jansson, E. Humphrey, N. Montecchio, R. 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)*, 2017, pp. 323–332.

[4] P. Chandna, M. Miron, J. Janer, and E. Gómez, "Monoaural audio source separation using deep convolutional neural networks," in *International Conference on Latent Variable Analysis and Signal Separation*. Springer, 2017, pp. 258–266.

[5] P.-S. Huang, M. Kim, M. Hasegawa-Johnson, and P. Smaragdis, "Joint optimization of masks and deep recurrent neural networks for monaural source separation," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 23, no. 12, pp. 2136–2147, 2015.

[6] Wikipedia, the free encyclopedia, "Bidirectional recurrent neural networks," 2015, [Online; accessed May 10, 2018]. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/c/c7/RNN_BRNN.png

[7] S. I. Mimilakis, K. Drossos, J. F. Santos, G. Schuller, T. Virtanen, and Y. Bengio, "Monaural singing voice separation with skip-filtering connections and recurrent inference of time-frequency mask," *arXiv preprint arXiv:1711.01437*, 2017.

[8] I. Mosseri and O. Lang. (2018) Looking to listen: Audio-visual speech separation. [Online]. Available: https://research.googleblog.com/2018/04/looking-to-listen-audio-visual-speech.html

[9] HyperPhysics, R. Nave, "Equal loudness curves," 2016, [Online; accessed May 12, 2018]. [Online]. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/Sound/eqloud.html

[10] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[11] J. Ganseman, P. Scheunders, and S. Dixon, "Improving plca-based score-informed source separation with invertible constant-q transforms," in *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*. IEEE, 2012, pp. 2634–2638.

[12] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," *IEEE transactions on audio, speech, and language processing*, vol. 14, no. 4, pp. 1462–1469, 2006.

[13] Z.-C. Fan, T. Chan, Y.-H. Yang, and J.-S. R. Jang, "Music signal processing using vector product neural networks," *arXiv preprint arXiv:1706.09555*, 2017.

[14] MACLab, "The ikala dataset," 2017, [Online; accessed April 30, 2018]. [Online]. Available: http://mac.citi.sinica.edu.tw/ikala/

[15] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1520–1528.

[16] Y. Luo, Z. Chen, and D. P. Ellis, "Deep clustering for singing voice separation."

[17] P.-S. Huang, S. D. Chen, P. Smaragdis, and M. Hasegawa-Johnson, "Singing-voice separation from monaural recordings using robust principal component analysis," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, 2012, pp. 57–60.

[18] Y.-H. Yang, "On sparse and low-rank matrix decomposition for singing voice separation," in *Proceedings of the 20th ACM international conference on Multimedia*. ACM, 2012, pp. 757–760.

[19] ——, "Low-rank representation of both singing voice and music accompaniment via learned dictionaries." in *ISMIR*, 2013, pp. 427–432.

[20] P. Sprechmann, A. M. Bronstein, and G. Sapiro, "Real-time online singing voice separation from monaural recordings using robust low-rank modeling." in *ISMIR*, 2012, pp. 67–72.

[21] J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe, "Deep clustering: Discriminative embeddings for segmentation and separation," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on.* IEEE, 2016, pp. 31–35.

[22] E. M. Grais, G. Roma, A. J. Simpson, and M. D. Plumbley, "Single-channel audio source separation using deep neural network ensembles," in *Audio Engineering Society Convention 140.* Audio Engineering Society, 2016.

[23] S. I. Mimilakis, E. Cano, J. Abeßer, and G. Schuller, "New sonorities for jazz recordings: Separation and mixing using deep neural networks," in *2nd AES Workshop on Intelligent Music Production*, vol. 13, 2016.

[24] S. I. Mimilakis, K. Drossos, T. Virtanen, and G. Schuller, "A recurrent encoder-decoder approach with skip-filtering connections for monaural singing voice separation," *arXiv*, vol. 1709, 2017.

[25] J. Dunn. (2018) Google works out a fascinating, slightly scary way for ai to isolate voices in a crowd. [Online]. Available: https://arstechnica.com/gadgets/2018/04/google-works-out-a-fascinating-slightly-scary-way-for-ai-to-isolate-voices-in-a-crowd/

[26] E. M. Grais, M. U. Sen, and H. Erdogan, "Deep neural networks for single channel source separation," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on.* IEEE, 2014, pp. 3734–3738.

[27] T.-S. Chan, T.-C. Yeh, Z.-C. Fan, H.-W. Chen, L. Su, Y.-H. Yang, and R. Jang, "Vocal activity informed singing voice separation with the ikala dataset," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on.* IEEE, 2015, pp. 718–722.

[28] Multimedia Information Retrieval lab, "Mir-1k dataset," 2009, [Online; accessed April 30, 2018]. [Online]. Available: https://sites.google.com/site/unvoicedsoundseparation/mir-1k/

[29] Zafar R., Fabian S. and Antoine L., "Professionally-produced music recordings," 2016, [Online; accessed April 30, 2018]. [Online]. Available: https://sisec.inria.fr/sisec-2016/2016-professionally-produced-music-recordings/

[30] R. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam and J. P. Bello, "Medleydb: A dataset of multitrack audio for music research," 2014, [Online; accessed April 30, 2018]. [Online]. Available: http://medleydb.weebly.com/

[31] B. McFee, "Librosa," https://github.com/librosa/librosa, 2018.

[32] S. Mobin, B. Cheung, and B. Olshausen, "Convolutional vs. recurrent neural networks for audio source separation," *arXiv preprint arXiv:1803.08629*, 2018.

[33] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[35] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[36] C. Raffel, "Mir eval," https://github.com/craffel/mir_eval, 2017.

Isolating the singing voice from music tracks: a deep neural networks approach to karaoke

Jonathan Deboosere

Supervisors: Prof. dr. Tijl De Bie, Dr. ir. Thomas Demeester
Counsellor: Paolo Simeone

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in Computer Science Engineering

Department of Electronics and Information Systems
Chair: Prof. dr. ir. Koen De Bosschere

Department of Information Technology
Chair: Prof. dr. ir. Bart Dhoedt

Faculty of Engineering and Architecture
Academic year 2017-2018

GHENT
UNIVERSITY