# The Effect of Deep Learning-based Source Separation on Predominant Instrument Classification in Polyphonic Music

**Diego Ligtenberg**

Under the supervision of Prof. Peter Kranenburg

Second examiner: Prof. Anja Volk

A thesis presented for the degree of
M.Sc. Artificial Intelligence

Universiteit Utrecht

Department of Natural Sciences
Utrecht, the Netherlands
15-09-2022

## Abstract

Music instrument classification is the task of detecting individual music instruments in music tracks. It remains a challenging task, particularly in polyphonic music. Prior research has shown that analytical-based music source separation can increase the performance of instrument classification. Music source separation is the art of extracting isolated instrument groups called stems from music tracks. We propose a novel deep learning-based source separation model in the time-frequency domain that learns to generate a combination of the 'vocals and other' stems. Additionally, we develop a postprocessing algorithm that increases the subjective performance of these stems. We also compare the objective performance between these raw and postprocessed stems and measure which of the stems positively impacts instrument classification. We find that only the postprocessed stems positively impact the performance of instrument classification. In addition, we perform instrument-wise analysis to examine which classified instruments are most affected by music source separation. We find that the cello, clarinet, piano and violin were the only instruments that were positively impacted. This research confirms the importance of the source-separated stem's quality. The instrument-wise analysis gives insights into which instruments benefit most from source separation and what source separation quality improvements are needed to increase the performance of instrument classification.

To my family, Wim, Erica, Diego, Jhon Jairo and Juliana.

Even in difficult times, all of you have helped, guided and supported me.

Thanks

# ACKNOWLEDGEMENTS

I am extremely grateful to everyone who helped me through my studies. Foremost, I would like to express my deepest appreciation to my supervisor Prof. Peter Kranenburg. His support through all the meetings and his positive reactions to my new ideas made the past year an incredible experience. You guided me throughout the research process, and I could not have wished for a better advisor and mentor. I would also like to show my gratitude towards Rob de Mooij, a constant friend who has helped me transition into this field and guided me with all the machine learning models. Your help and support were indispensable. Last but not least, I would like to thank Thomas van den Broek for his emotional support throughout difficult times, the late-night drinks, and the guidance through parts of my life where I had almost lost hope.

# Contents

# List of Figures

# Chapter 1

# Introduction

In this chapter, we introduce our work, instrument classification, with music source separation as preprocessing step; for both methods, deep learning-based approaches are utilized. First, the motivation and usefulness of the project are given. We then show the literature research on the related work and discuss existing methods and their results. Afterwards, the objective of the thesis is given. The last section lists the organization of the thesis.

## 1.1 Motivation

Music instrument classification (MIC) is a fundamental task of music information retrieval (MIR) that attempts to identify single instruments used to compose a music track. Accurate MIC has been useful as a conditional tool for various MIR tasks such as music genre classification, music similarity computation, and automatic music transcription [37]. Recently, MIC has gained a lot of interest from commercial firms in the music recommendation industry [33]. In their recommendation applications, MIC can function as a standalone product, increasing the depth of user queries; users could search for music tracks containing specific instruments. It could also be used as a content-based descriptor that helps their recommendation algorithm to provide users with songs they prefer listening to. Content-based descriptors are features in music tracks (e.g., instrumentation, tempo, pitch) that can be extracted solely from the music track itself [22]. Historically, these descriptors were mostly avoided as analytical feature extractors provided insufficient predictive power or were computationally too expensive compared to collaborative or context-based descriptors [37]. However, with the rise of deep learning-based prediction models that automatically extract features, and a yearly increase in computational power, the use of content-based descriptors has become more appealing [29].

The quality of MIC has dramatically improved over the past two decades. Most improvements in MIC can be attributed to improved datasets, data preprocessing, feature extraction and classifier algorithms [37]. The main focus in older MIC research was improving data preprocessing and feature extraction [4, 23, 25, 27, 39]. Specifically, using analytical music source separation algorithms as preprocessing step had shown variable but promising results [3, 25]. With the rise of deep learning in the early 2010s, most MIC researchers have shifted their focus towards building deep learning-based instrument classifiers, which perform mostly better than their traditional machine learning counterparts [15, 38]. Deep learning has also revolutionized the field of music source separation, with deep learning-based models outperforming analytical approaches [45]. To our knowledge, no recent research has focused on improving MIC by applying improved source separation algorithms as preprocessing steps for their instrument classifiers. In this research, we aim to investigate whether deep learning-based music source separation has the potential to increase the performance of MIC.

To summarize, deep learning-based music source separation can potentially increase the performance of MIC algorithms. Better MIC algorithms have several use cases; first, they can improve the performance of recommendation systems by serving as a standalone product or as content-based descriptor. Second, it could help in other MIR tasks by serving as a conditional tool.

## 1.2 Related work

Humans are reasonably good at recognising musical instruments in polyphonic music (music where at least two or more instruments play simultaneously). Building a system that allows computers to do this is still an active area of research in MIR [37]. This section chronologically discusses the scientific literature for MIC, showcasing how deep learning methods took over feature-based approaches. Traditionally the main focus of MIC algorithms was content-based feature engineering [4, 23, 25, 27, 39]. Researchers would extract task-specific features (e.g., amplitude envelopes, spectral flux, 0-crossing rate) from music tracks and then use traditional machine learning methods, such as logistic regression or support vector machines to classify the instruments. These task-specific features were limited as they could not capture all characteristics inherent to the music track. Furthermore, there was information overlap between features, whereas other information about the original music track that was not captured by any of the features was lost [42].

Starting as early as 2006, one of the first pipelines was built for polyphonic MIC [39]. This research aimed to identify which of the four classical instruments (flute, piano, trumpet and violin) were playing in an audio segment from concerti and sonatas (IOWA collection dataset). To do this, the authors extracted mel-frequency cepstrum coefficients (MFCCs) to compute mel spectrograms from each note sample in the dataset. Due to computational limitations, they had to reduce the dimensionality of these mel spectrograms using principal component analysis (PCA). Finally, the PCA-reduced mel spectrograms were used as input to a k-nearest neighbour classifier. This classifier did work, but performed significantly worse than human annotators did. The authors mentioned that training on monophonic data, while using a polyphonic test set could be the cause of the poor performance. After that, several studies found that training on polyphonic data produced better results [25, 27].

In the late 2000s, most MIC improvements were due to better classifier algorithms or prior transformations to the input audio [12, 23]. The first researchers started applying music source separation as preprocessing step to their MIC frameworks [25]. Music source separation is the art of separating specific sources from an audio file (e.g., extracting isolated drums, vocals or bass stems from a music track). The idea originates from the cocktail party effect [6], which refers to the ability of humans to focus one's attention in a crowded area on a particular auditory stimulus, while filtering out a range of other stimuli (i.e., crowd noise). In their research they attempted to separate foreground recorded instruments (monaurally recorded instruments, e.g, vocals, bass, drums) and background instruments (stereo recorded instruments, e.g., strings, brass, guitar) from the original input music track. The extracted backward stems, mainly containing the instruments to be recognised, were then used to compute wavelet transforms. These transforms were the input to an evolutionary classifier algorithm that could detect the presence of the instruments, producing equal or superior performance compared to other methods at that time.

Hereafter, another researcher investigated whether music source separation could improve the performance in MIC [3]. In his PhD thesis, he used the flexible audio source separation toolbox (FASST), to split mel spectrogram representations of the original music tracks into drums, bass and other stems (groups of instruments that belong together). Combinations of these mel spectrogram representations were the input to a support vector machine-based classifier. He then examined whether the separated sources would increase the performance of MIC. He found that performance increased the most when source separation was applied on both train and test sets. Some significant limitations of this study were the low quality of MFCCs; the use of 38 mel bands is relatively low for today's standards. Moreover, the FASST algorithm introduced many auditory artefacts that further harmed the quality of the music tracks. Next, the support vector machine used as a classifier is not an ideal algorithm for spectrogram-based MIC [15]. Finally, the separation step was computationally expensive (taking up to two minutes of processing time per second of audio), making the process not feasible for practical use. Yet, considering these limitations, performance on MIC still increased, setting the stage for further research.

Starting in the early 2010s, more and more researchers began shifting their focus from feature-based engineering and source separation toward building better deep learning-based classifiers. Convolutional neural networks (CNNs) were the first deep learning-based classifiers that started dominating the MIC field, resulting in significant improvements in the state of the art at that time [37]. The success of the convolutional neural network can not only be explained by its architecture.

Increases in computing power enabled the use of higher resolution representations of sound, thus, spectrograms or mel spectrograms could be directly used as input for the classifier [37]. These representations contain practically all characteristics of a digital audio signal; thus no information is lost. Since their emergence, several architectural changes have proved to be robust in improving convolutional model performance; residual connections, inception modules and batch normalisation [18]. However, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners in 2015 showed that by adding more layers and thus creating deeper CNNs that contain more parameters, model performance significantly increased [37]. This led to a still-going race for bigger models. Large models take longer to train, the inference is slower and they require computer specialisations that are sometimes unavailable for consumer products [40].

To our knowledge, only [15] followed up on the research of [12, 25] and used source separation as preprocessing step for MIC. In their research,[15] used a convolutional classifier which took as input source separated mel spectrogram representations of an original jazz music dataset. They evaluated the performance of two analytical-based separation algorithms; phase-based harmonic/percussive and pitch-based solo/accompaniment separation. The phase-based harmonic/percussive separation is an analytic solution [15]. It works under the assumption that harmonic music instruments have stable phase contours when differentiating the phase spectrogram in the time domain. In contrast, given percussive instruments' broadband and transient-like characteristics, stability in phase cannot be expected [15]. The pitch-based solo/accompaniment separation is separated by extracting pitch information from the solo instrument, and then closely tracking its harmonic components to create a spectral mask [5]. Pitch information is extracted by performing a pair-wise evaluation of spectral peaks and finding partials with well-defined frequency ratios. The pitch information extracted is then used to track the harmonic components, which can differentiate between solo and accompaniment stems using amplitude modulation, inharmonicity, attack length, and saliency as underlying concepts [5]. The phase-based harmonic-percussive separation algorithm produced low-quality separation stems, resulting in worse performance on the MIC task [15]. The pitch-based solo/accompaniment separation algorithm performed better. The separated stems were of higher quality, especially for instruments with clear and stable partials (woodwind and string instruments) [15]. However, the separation was noisier for instruments with less stable spectral behaviour, such as the electric guitar or music instruments that are distorted due to sound effects [15]. Despite the quality issue's on specific instruments, this separation algorithm produced slightly better results on the MIC task.

Lastly, a framework was proposed for automatic singer identification in pop songs utilising music source separation [38]. The authors compared the performance of an analytic and two deep learning-based source separation algorithms; harmonic/percussive separation, time-frequency-based convolutional separation and time-based Wave-U-Net separation [43]. The algorithms extracted harmonic or only vocal stems from a dataset containing pop songs. Mel spectrograms of these stems were computed and used as input for a Gaussian mixture model. They found that the deep learning-based source separation significantly outperformed the analytical approaches in the singer identification task, with Wave-U-Net as the clear winner [38].

Most research has shown that music source separation can boost the performance of MIC, given that the extracted source stems are of high quality. We argue that the clever use of feature engineering which was often performed in the early stages of MIC, particularly music source separation, could potentially increase the performance of current MIC models. To our knowledge, no recent work applied deep learning-based source separation on instrument classification datasets. The next session discusses the progress of music source separation algorithms in more detail, showcasing how deep learning-based separation algorithms produce higher-quality stems compared to analytical methods.

## 1.3 Source Separation

When producing music, recordings of individual instruments are arranged together (called stems) and mastered (merged) into the final music track (mixture). Music source separation aims to recover or generate those individual stems from the mixed signal [7]. Most recent research in source separation divides the stems into four broad categories: drums, bass, other and vocals. The other stems contain all the leftover instruments that do not fall into the more precisely defined stems. Given a music track which is a mixture of these four stems, the goal is to generate waveforms that correspond to one or more (combinations) of the original stems [11].

We have discussed how older analytical source separation algorithms are prone to artefacts and highly dependent on the harmonics of the instruments played in the original music track [3, 15, 25]. Deep learning-based separation models, constructed with the revolutionary U-net architecture, suffer less from these problems [40, 42]. U-net is an encoder-decoder convolutional generator architecture, initially developed as an image segmentation tool for biomedical images [35]. It owes its success to the use of skip connections between convolutional layers. The skip connections preserve important information from layers in the encoder so that the decoder can accurately reconstruct spatial-related multi-dimensional target data from the input data. Since the emergence of U-net-based generator models, which is the backbone for almost all modern deep learning-based source separation algorithms [11], approaches to extract stems from the original music tracks (mixtures) can be split into three categories: spectrogram-based (time-frequency) generator models, waveform-based (time) generator models, and combinations of these two (hybrid generator models).

Spectrogram-based models attempt to extract isolated stems from mixtures by first converting the waveform of the input mixture to spectrogram representations [8]. The generator model is then supposed to learn a mapping from the spectrogram of the mixture towards a specific target spectrogram. The predicted target spectrogram is then converted to waveform using some short-time Fourier transform inversion algorithm (e.g., istft, Griffinlim [16]). Different spectrogram models vary primarily in the input data, either represented by its amplitude or as the concatenation of its real and imaginary parts [11]. Similarly, the output can be either a mask on the input spectrogram, modulation of the input spectrogram, or the complex-as-channel representation of the input spectrogram [11]. We will now discuss some architectures using several different configurations.

Spleeter by Deezer [20] uses the original U-net framework with the addition of dilated convolutions and a mask-based final layer. The input of their model is the amplitude of the mixture spectrogram. Their final layer generates a spectrogram mask with values bounded between zero and one, using a sigmoid activation function. This mask is multiplied with the input layer to produce the target source. The predicted target source's phase is then restored to its waveform using the Griffinlim algorithm. Finally, Wiener filtering [50] was used on the output spectrogram to reduce the noise produced by the algorithm, resulting in better-perceived quality of the stems. The addition of dilated convolutions increased the receptive field of the convolutional kernels by inserting holes between the consecutive elements of the input data. The holes help expand the area covered by the kernels, but detailed local information is lost. The Wiener filter, reduces the overall resolution of the sound, to reduce perceived noise [13, 31, 50].

D3Net [46] is another spectrogram-based source separation architecture that exploited dilated convolutions. The difference with Spleeter is that its final layer directly predicts the target spectrogram allowing the model to learn non-linear relations with respect to the input layer. This is better, as mask-based output layers have to be linear combinations of the input layer, which is not ideal when harmonics of different sources overlap [8]. Directly predicting the output layer made D3Net's predicted stems sound less 'hollow'. Furthermore, they examined the positive effect of adding 1500 songs to their training data. Their standard and extra data model outperformed Spleeter in SDR for all stems.

LaSAFT [8] is a spectrogram-based source separation architecture that uses Complex-As-Channels as its input and output. They directly learn the magnitude and phase of the short-time-Fourier transform (STFT) representation of the audio removing the Griffin-lim algorithm's need. They argue that learning the separation model's phase increases the source separation algorithms' upper bound. However, their models are more unstable to train, since it needs to not only learn twice as much data, but the phase data should also be consistent with the magnitude spectrograms not to cause audio artefacts. Lastly, [24] experimented with generative adversarial neural networks but found the training very unstable. They even measured worse performance than regular convolutional-based approaches.

Waveform-based source separation methods such as Wave-U-Net[43] and Conv-Tasnet [28], always had underperformed their spectrogram-based counterparts. This changed when the Demucs architecture won the signal separation evaluation campaign (SISEC) in 2018 using waveforms directly as input [11]. The architecture of Demucs is quite similar to spectrogram-based methods as it still uses U-Net as its backbone. The main difference is using 1-dimensional convolutions and a larger stride to downsample the waveform data. Furthermore, two bidirectional long-short term memory (BLSTM) are added inside the latent space. The last layer of the model directly predicts the waveform without the use of masking. The benefit of waveform-based methods is that the loss function is directly based on the raw target waveform instead of the magnitude of the spectrogram. This fixes phase reconstruction artefacts that occur in spectrogram-based methods [41]. A problem of the waveform approach is that due to the 1-dimensional data, even after downsampling with larger strides, the memory required to train such models is still significantly larger than spectrogram-based approaches. This results in longer and more unstable training time, given that the same amount of computational resources are available [11].

Theoretically, there should be no difference between spectrogram and waveform models. Especially when considering CaC (complex as channels), which is only a linear change of base for the input and output space [10]. However, the MUSDB-18 dataset, containing only 150 songs, is relatively small which makes it susceptible to inductive bias of either the spectrogram or waveform models. Additionally, the test errors measured in both model types remain high, indicating the models produce artefacts in the generated target stems. Spectrogram models often suffer from phase inconsistency problems that make the attack of drums sound less punchy, whereas waveform models produce crunchy static noises, especially in the vocal stems [11]. Because of this, the latest trend in music source separation is to use hybrid models that take the best information from both waveform and spectrogram-based models. Some examples are; hybrid Demucs and HTMD-net [10, 14].

Although waveform-based and hybrid separation methods produce promising results, they are more unstable to train. Additionally, both train and inference times take significantly longer than their spectrogram-based counterparts. For this reason, we implement a spectrogram-based source separation model that can extract input mixtures' vocals and other stems. These stems are the input for the final MIC classifiers that are used to investigate whether using the deep learning-based separated stems increases the performance.

## 1.4 Objective

To our knowledge, the last research that investigated whether music source separation could improve the performance on instrument classification was [15]. Their instrument classification datasets were transformed into source-separated stems using analytical separation algorithms. The main goal of this thesis is to use deep learning-based source separation, which has shown to produce higher quality stems than the analytical-based separation algorithms [38], and measure its effect on MIC for polyphonic music. To do this, we develop a spectrogram-based source separation algorithm that removes the 'drum and bass' stems from music tracks. The remaining 'vocals and other' stems contain all the instruments present in the IRMAS dataset. These stems will be used to train and test instrument classifiers whose performances are compared with a base model that uses no source separation.

Secondly, several other research has shown that the quality of source separation algorithms affects the influence of source separation on MIC [3, 15, 25, 38]. This research will investigate whether this still holds for deep learning-based separation algorithms. To do this, we evaluate the objective and subjective quality of the raw and postprocessed output of the source separation model and examine whether the differences in separation quality impact the performance of the final instrument classifier.

Lastly, previous research has shown that the effect of source separation depends on the characteristics of the instruments that are classified [15, 25]. However, the evaluation metrics for the instrument classifiers are averaged over all instruments. This does not allow the reader to examine which instruments benefit most from source separation. Therefore, we provide instrument-wise performance metrics to get more detailed insights into the effect of source separation on instrument classification.

## 1.5 Outline

The rest of the thesis is organized as follows. Chapter 2 introduces the data that is used in this research. An introduction to sound is given, and relevant feature representations of audio, including their advantages and disadvantages, are discussed. We then showcase the datasets used for both source separation and instrument classification. Chapter 3 showcases the method that was used to perform this research. A description of each model's pre and postprocessing pipeline is given. The model architectures are explained, and the evaluation metrics used to test the hypothesis of this research are presented. Chapter 4 showcases the results of the experiments. The discussion and the limitations of this study are covered in Chapter 5. Chapter 6 presents the final conclusion of the research.

# Chapter 2

# Data

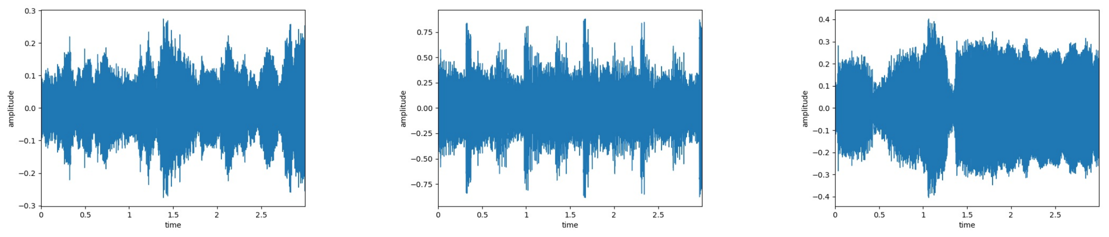
In this chapter, the necessary theory about audio is explained. Afterwards, the datasets used in this thesis are shown.

## 2.1 Sound

Sound is the auditory sensation produced by the vibration of an object in a medium (e.g., air, water). Vibrations determine the oscillation of molecules, creating an alternation of high and low pressure in a medium [48]. The human ear can detect a fixed range of different pressure alternations, which we perceive as sound.

### 2.1.1 Waveform

A waveform is a function of a sound signal's amplitude over time which can be visualized in a graph. Example waveforms can be seen in fig. 2.3. Two important perceptual properties are embedded in waveforms; amplitude and period, that correspond to the human hearing perception of loudness and pitch (fig. 2.2).



(a) Violin (b) Guitar (c) Singing voice

Figure 2.1: Examples of three waveforms

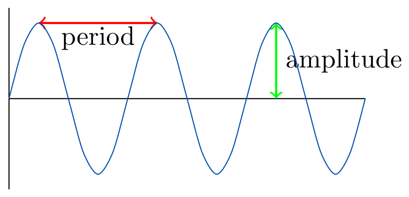


Figure 2.2: Amplitude and period of a sinusoidal waveform

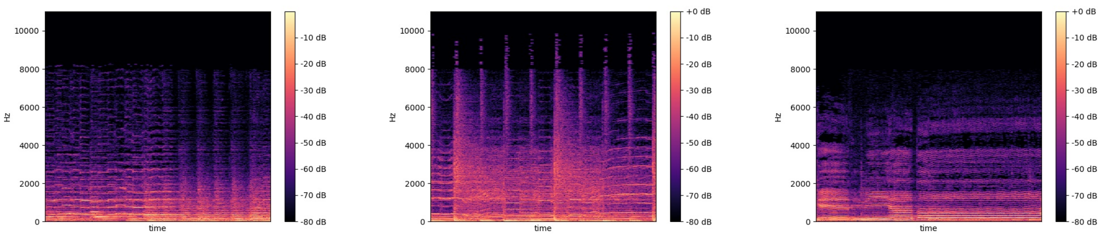
The amplitude of a waveform corresponds to the distance between its high and low-pressure points and the equilibrium position. Amplitude directly relates to loudness; how ”loud” or ”soft” we perceive a sound. Louder signals have a higher amplitude. The basil membrane in the human ear creates a stronger ’push’ when the amplitude is high, resulting in a louder perceived audio signal [37]. In graphs, the amplitude is often transformed to a base ten logarithmic scale as the ratio between the raw sound intensity that leads to permanent damage and the limit of the smallest audible sound is close to 1 trillion ($10^{12}$) [9]. The unit for amplitude in the logarithmic scale is called decibel (dB) [37].

Pitch relates to the perception of tone height and is determined by the period of the waveform. Larger distances between each amplitude peak indicate lower sounds, and smaller distances indicate higher sounds [48]. It is a convention that pitch is measured in frequencies (Hz), which is the inverse of the period. Studies have shown that humans do not perceive frequencies on a linear scale [21]. We are better at detecting differences in lower frequencies than in higher frequencies [47]. For example, differences between 100 Hz and 200 Hz are easily noticeable, whereas the difference between 10.000 Hz and 10.100Hz is significantly harder to detect. Furthermore, for instruments, which are composed of several repeating frequency patterns, perceived pitch relates to the loudest frequency in the lowest frequency pattern (fundamental frequency). The other frequencies patterns are overtones or partials which form the timbre of the sound. Timbre helps us tell the difference between two musical instruments [37].

Waveforms found in nature are continuous. Digital waveforms are sampled discretized representations of continuous waveforms. Computers can handle these discretized representations [37]. The resolution or sampling rate of the discretized waveform is measured in hertz (Hz). It indicates how many data points per second of the continuous waveform are processed by the computer. In this thesis, we will be using a sample rate of 44.100 Hz.

### 2.1.2 Spectrogram

A spectrogram is a visual representation of an audio signal’s amplitude and pitch over time [22]. Spectrograms are usually visualized as coloured images, where the horizontal and vertical axes represent time in seconds, and frequency in Hz, while the colour represents amplitude in dB. Examples of spectrograms can be seen in fig. 2.3.



(a) Violin (b) Guitar (c) Singing voice

Figure 2.3: Examples of three spectrograms

The short-time Fourier transform (STFT) generates a spectrogram from a waveform [48]. First, it splits a waveform into short-time windowed frames. Discrete Fourier transforms are applied to each frame, resulting in a frequency spectrum. Next, the power spectrum from the amplitude part of the frequency spectrum is computed. Finally, the spectrogram is obtained by transforming the amplitude scale of the power spectrum to decibels. A more detailed explanation is given below.

1. First, the original waveform x(n) is split into overlapping short-time frames with equal length. Each frame contains $N$ samples which corresponds to the frame length. The number of samples between each overlapping frame is denoted by the hop size $H$ (fig. 2.4). In this research, we use a frame length of 4096 and a hop size of 1050. The following formula represents the samples in each frame

$$x_\alpha(n) = \begin{cases} x(n + (\alpha - 1) \cdot H), & n \in [0, N-1] \\ 0, & \text{otherwise} \end{cases} \tag{2.1}$$

where $\alpha$ is the index number of one short-time frame, and $x_\alpha(n)$ denotes samples in the $\alpha^{th}$ frame taken from original signal $x(n)$.

2. Each frame is then applied with a window function w(n) to overcome spectral leakage [17]. We used the Hann window function (fig. 2.5).

$$x_{\alpha w}(n) = x_\alpha(n) \cdot w(n) \tag{2.2}$$

3. The short-time Fourier transform is then defined as applying discrete Fourier transforms to each windowed overlapping short-time frame. This transforms each windowed audio signal into a complex frequency spectrum containing the amplitude $\hat{x}_a(k)$ and phase $\hat{x}_\rho(k)$ matrices for each k$^{\text{th}}$ frequency.

$$\hat{x}(k) = \sum_{n=0}^{N=1} x_{\alpha w}(n) \cdot e^{-2\pi n \frac{k}{N}} \qquad k = 1, 2 ... \frac{N}{2} \tag{2.3}$$

4. We then omit the phase $\hat{x}_\rho(k)$ and compute the power spectrum density by taking the element-wise square of the amplitude matrix from the frequency spectrum.

$$P(k) = \|\mathbf{\hat{x}_a(k)}\|^2 \tag{2.4}$$

5. Finally, the spectrogram is obtained by converting the power into the decibel scale.

$$P^{dB}(k) = 10 \log_{10}(P(k)) \tag{2.5}$$

The benefit of the spectrogram representation is its two-dimensional spatial-related information. These properties are similar to images, allowing for the use of convolution algorithms that efficiently extract features from two-dimensional data [20]. Therefore we will be using spectrogram representations of waveforms for both music source separation and instrument classification.

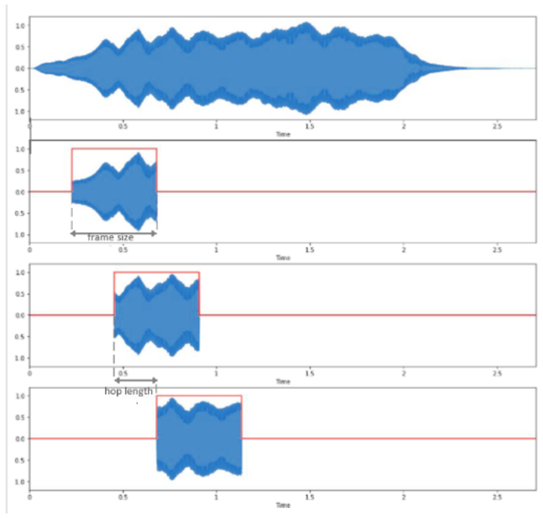


Figure 2.4: Waveform split in overlapping frames

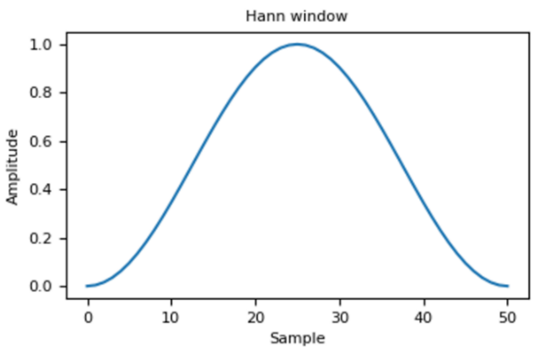


Figure 2.5: Hann window function

## 2.2 Audio Dataset

Two datasets are used in the overall pipeline of this thesis: the MUSDB-18 dataset for source separation and the IRMAS dataset for music instrument classification. Both of these datasets are benchmarks in their respective fields and are used in most of the recent source separation and MIC articles considered in this thesis.

### 2.2.1 MUSDB-18 dataset

The MUSDB-18 is the benchmark dataset for music source separation [34]. It was used for the 2018 community-based Signal Separation Evaluation Campaign (SISEC) competition, where incredible improvements were made in the field of source separation [44]. The dataset contains 150 full-length music tracks (10h duration) with individual track lengths ranging from two to eight minutes. All tracks are stereophonic, encoded at 44.1kHz and were downloaded in the waveform format. The tracks cover a variety of music genres such as pop, rock, rap, EDM, jazz, folk and classical music to name a few. It comes with a formerly divided training set composed of 86 music tracks, a validation set of 14 tracks and a test set of 50 tracks. The 100 train and validation tracks are taken from the DSD100 dataset, which itself was derived from The' Mixing Secrets' Free Multitrack Download Library [34]. The test set contains 46 tracks taken from MedleyDB, Native Instruments provided two tracks and the other two tracks came from the Canadian rock band The Easton Ellises. Each music track contains isolated drums, bass, vocals and other stems. Together they form the track mixture that is used as input for our separation model. Section 3.1 will discuss the creation of the target stems used in this research.

### 2.2.2 IRMAS dataset

The IRMAS dataset is developed for polyphonic instrument classification and was derived from the dataset in Ferdinand Fuhrmann's PhD thesis [12]. The dataset contains 8772 audio segments, which are stereophonic and encoded at 44.1kHz. All audio segments were downloaded in the waveform format. The dataset comes with a formerly divided train and test set. The train data contains 6700 three-second segments with monophonic instruments. Genres include; country-folk, classical, pop, rock, latin and soul [12]. In contrast, the test dataset contains 2072 polyphonic segments of varying lengths (five to twenty seconds). For the whole duration of the segment, the predominant instruments stay consistent [3]. Genres include; rock, pop, classical, jazz, electronic and folk music. The eleven instruments included in both train and test dataset are: cello (cel), clarinet (cla), flute (flu), acoustic guitar (gac), electric guitar (gel), organ (org), piano (pia), saxophone (sax), trumpet (tru), violin (vio) and human singing voice (voi). We chose to customise the predetermined train and test data splits and adjust the length of the segments in the test data for the following two reasons. First, the train data consists of only monophonic music, which has shown to result in poor performance for polyphonic music classification [25, 27]. Second, Convolutional neural networks have difficulty dealing with the varying length audio segments between train and test data. A detailed explanation of the data customisation can be seen in section 3.1.

# Chapter 3

# Method

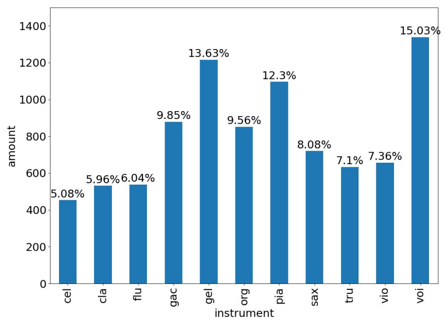
This chapter explains the methods used to answer the research questions. First, an explanation is given about the data preprocessing required to convert the waveforms of both datasets into model-ready spectrogram representations. We then describe how the spectrogram-based source separation model, which extracts the combination of vocals and other stems from music tracks, was built. Next, the postprocessing algorithm that is used to remove audio artefacts of the separation model's output is covered. The source separation section ends with the commonly used SDR metric that evaluates the performance of the raw and postprocessed stems produced by the model. The instrument classification starts with a description of the workings of the three CNN-based instrument classifiers; base, raw and postprocessed. The parameters that were used for the models are then discussed. Afterwards, an overview of the evaluation metrics that are used to measure the performance of the instrument classifier is shown. Lastly, the validation of the statistical methods used to compare the performance of the source separation and instrument classification models is given.

## 3.1 Data preprocessing

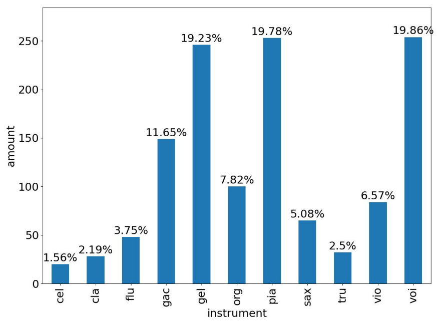
This section will discuss the preprocessing steps taken for both music source separation and instrument classification datasets. First, we explain how the target waveforms for the spectrogram-based autoencoder were created from the MUSDB-18 dataset. Next, we elaborate on the steps taken to create a custom IRMAS dataset with new train and test splits. Finally, we specify the preprocessing steps required to convert the waveforms from the MUSDB-18 and IRMAS datasets toward model-ready spectrogram representations.

Starting with music source separation, the goal is to train the spectrogram-based autoencoder such that it learns to convert the track mixture towards waveforms containing only the target stem. The track mixture is a combination of all the stems; drums, bass, vocals and other. The target stem's waveform should contain all the instruments that need to be identified in the IRMAS dataset. To do this, we removed the drum and bass stems from the music tracks. The final target stem is then created by merging the remaining' vocals and other' stems using addition. We now have two waveforms; the track mixture and the target waveform containing only the combination of the vocals and other stems.

Next, instrument classification; we customised the original IRMAS dataset as the predetermined train split contained only monophonic music and the test set contained varying length audio segments. First, we added a part of the polyphonic test set to the monophonic train dataset as training on polyphonic music has shown to provide better results [25, 27]. Second, to get matching length segments in train and test data, the segments in the test dataset were padded such that only the first three seconds remained. The padding does not influence the results because of the following property belonging to the IRMAS test dataset; the predominant instruments are playing for the whole duration of the segments [3]. The final customised train and test datasets contain respectively 8006 (6700 monophonic, 1306 polyphonic) annotated train segments for a total of 8911 instruments and 766 polyphonic annotated test segments with a total of 1279 instruments. 20% of the train data is used for validation purposes. The instrument distributions for the train and test dataset can be seen in fig. 2.3. We used class weights to counteract the slightly uneven instrument distribution in the train dataset. The test dataset shows an even more skewed distribution (e.g., cello and clarinet). The IRMAS dataset simply contained no more polyphonic data for the instruments with low presence. When interpreting instrument-wise evaluation metrics for the MIC classifier, we will be mindful of this.



(a) Train data distribution



(b) Test data distribution

We will now discuss the preprocessing steps required to convert waveforms from the MUSDB-18 and IRMAS datasets toward model-ready spectrogram representations. This consists of the following steps; converting the stereo waveforms to mono, splitting or padding the waveforms into segments of three seconds, converting the segments into spectrograms, and normalising the spectrograms. First, the 44.1kHz stereophonic waveforms are converted to mono as it significantly reduces the data required to train the models. This conversion does not change the key characteristics of the instrumentation inside the audio signal, making its effect on the performance of the models small. Second, we split the MUSDB-18 tracks into three-second audio segments for computational purposes; the end-of-song segments are zero-padded. The train data in the IRMAS dataset already has a duration of three seconds per segment. As explained before, we only took the first three seconds for the test dataset. Third, the three-second segments are converted to logarithmically-scaled amplitude spectrograms using the STFT (hop size = 1050, frame size = 4096) described in section 2.1.2. The initial dimensionality of these spectrograms is 2049 (frequency) by 126 (time). However, these dimensions would result in resolution problems, since autoencoders repeatedly downsample the data by factors of two. To prevent this, we removed the highest frequency bin and added two time bins, resulting in 2048 by 128 sized spectrograms. These dimensions are powers of two; thus, downsampling can be applied repeatedly without running into resolution problems. Finally, we normalise the logarithmic amplitude spectrograms by dividing the logarithmic amplitude values by 150. This normalisation bounded the amplitude values to a new range between -1 and 1. This range is ideal because neural networks are better at handling smaller valued data [11]. These normalised logarithmic amplitude spectrograms are used as input for both source separation and instrument classification models

## 3.2 Spectrogram-based source separation

In this section, we explain the workings of our spectrogram-based autoencoder and showcase the parameters used in this research. It is inspired by several other source separation algorithms in both waveform and time frequency domain [11, 20, 41, 45]. The spectrogram-based autoencoder is a U-net-based generator model [35] that learns a mapping from the spectrogram of the input mixture towards the target spectrogram; the combination of vocals and other stems. Its architecture consists of an encoder, bottleneck and decoder. In an unsupervised manner, the encoder learns to create data encodings from the input mixture which are stored in the bottleneck. The decoder decodes these encodings towards newly generated target spectrograms. Additionally, the encoder layers are able to send relevant information to the decoder using the skip connections that concatenate the encoder layers with their corresponding decoder layers. We propose the following architecture for our spectrogram-based autoencoder fig. 3.2.

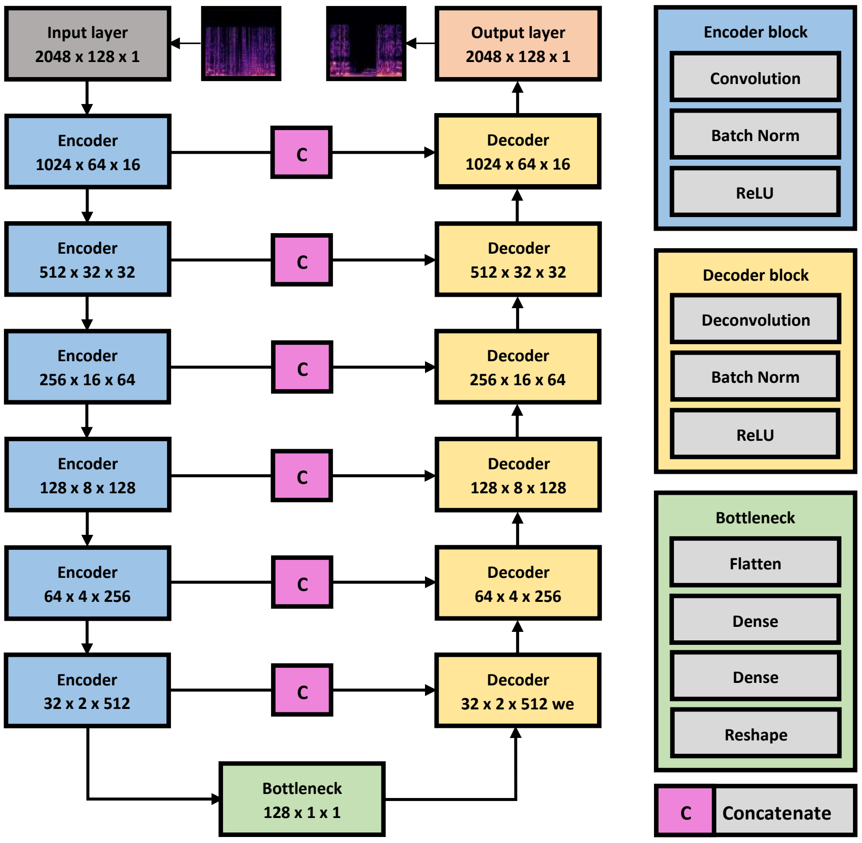


Figure 3.2: Spectrogram-based autoencoder

The spectrogram-based autoencoder contains six layers of convolutional blocks that compress the input spectrograms into a compact encoded representation. Each convolution block consists of a 2D convolution followed by batch normalisation and a ReLU activation function. We experimented with the dilated 2D convolutions [41] that expand the receptive field of the model. However, we omitted these as we only use a small number of convolutional layers and the dilation caused too much information loss which resulted in convergence problems. We think that if more computational memory is available, increasing the number of layers could potentially make the use of dilated convolutions a viable option. After the sixth layer, the encoder output reaches the bottleneck, which restricts the flow of information from the encoder to the decoder. The bottleneck contains the maximum amount of information of the input in a compressed lower-dimensional representation. The decoder includes six layers of convolutional decoder blocks that decompress the compact information in the bottleneck towards the dimensionality of the target spectrogram. The first five decoder blocks consist of 2D transposed convolutional layers followed by batch normalisation and a ReLU activation function. The last layer consists of a 2D transposed convolution, batch normalisation and TanH activation function. Unlike [20], we did not choose to make the output layer a mask of the input spectrogram. Instead, we directly produced the target output in the final layer utilising the activation function. The TanH function compresses the output values to the range -1 and 1, which matches the range of the amplitude values in our normalised logarithmic spectrograms. Therefore, our model directly outputs normalised amplitude spectrogram representations of the target source containing only the combination of vocals and other stems from the original input mixture.

### 3.2.1 Model parameters

We chose the following architectural model parameters to initialise the model as preliminary experiments showcased these parameters performed best. The Convolutional filters for each layer; (16, 32, 64, 128, 256, 512), which is similar to [20]. The filters store the spectrogram features.

The early layers detect mostly detailed local shapes (edges of certain frequencies) that are often shared between different spectrograms. The deeper layers contain more conceptual features. As these features are more diverse, more filters are needed to capture the differences. Convolutional kernels; (5, 5, 3, 3, 3, 3), larger kernel sizes in the early layers resulted in faster convergence. The strides were 2 for each layer, serving as a downsampling mechanism. The first dense layer in the bottleneck reduces the data and creates a latent space dimensionality of 128. The second dense layer upsamples this latent space towards the dimensionality of the flattened encoder output. Combined with the skip connections, we found that the dimensionality of the latent space is expressive enough for the decoder to upsample the compressed data representation towards the target spectrograms [45].

To prevent overfitting, small experiments (50 epochs) were conducted for three regularisation methods; data augmentation, dropout and weight regularisation. The following data augmentations, also used by Open Unmix and Demucs [11, 45], were applied; polarity inversion (multiplying the waveform by -1), time stretching (scaling the speed of the sound by a random factor between 0.5 and 1), pitch scaling (scaling the waveform by -4, -2, -1, 0,... +3, +4 semitones) and lastly, signal reversing. We experimented using 50% dropout in the first three encoder layers as described in [20] but omitted this from the model as it produced a lot of noise in the final output. We then experimented with L2 weight regularisation (parameter=1e-4), which penalises large weights. This slowed down training but reduced overfitting. We also experimented with truncated normal weight initialisation. The model's weights were initialised with a mean of 0 and a standard deviation of 1, drastically improving the model's convergence speed as it reduces the exploding gradient problem [32]. Lastly, we experimented with mean squared error (mse) and mean absolute error (msa) loss functions and found that msa produced the best results.

The models were trained on 1 GTX 960 GPU with 6 GB of VRAM for a total of 375 epochs, where one epoch is defined as a pass over all 3-second audio segments in the dataset. Each epoch took approximately 8 minutes, making the total train time around 50 hours. We selected the best model based on validation loss (mae) and calculated the evaluation metrics of this model on the test set. The Adam optimiser with a learning rate of 3e-4 and a batch size of 8 was used.

## 3.3 Postprocessing of source separation

In this section, the prostprocessing steps that are applied to the output of the model are discussed. The outputs of the spectrogram-based autoencoder are normalised logarithmic spectrograms. These are transformed into denormalised amplitude spectrograms to calculate the predicted stems' objective performance (SDR). However, objective performance metrics do not always correlate well with subjective human evaluation. To evaluate the reconstructed sound's subjective quality, the output spectrograms need to be converted to reconstructed waveforms. This process is explained in detail. Afterwards, the postprocessing steps that were taken to improve the subjective performance of the model are discussed.

### 3.3.1 Reconstruction of waveforms

The spectrogram-based autoencoder outputs normalised logarithmic spectrograms representing three-second audio segments of an input track. The following steps were used to convert these spectrograms into waveforms: denormalisation, time-frequency bin reconstruction, decibel to amplitude transformation, Griffinlim transformation and segment concatenation. First, to denormalise, the spectrogram-based autoencoder's output is multiplied by 150. Second, the original 2049 (frequency) by 126 (time) dimensionality is re-obtained by adding the removed frequency bin and removing the two added time bins to the spectrograms. These spectrograms are then converted to amplitude spectrograms by an inverse amplitude-to-decibel function. Finally, the Griffinlim algorithm is used to convert the amplitude spectrograms into waveforms. This algorithm iteratively runs inverse short-time Fourier transforms and estimates the omitted phase of the frequency spectrum that was omitted in the data preprocessing step [16]. We found that this algorithm provides better perceptual wave reconstructions than applying the inverse Fourier Transform combined with the phase of the input mixture used by [20]. Lastly, all three-second segments of the same song are concatenated to reconstruct the original music track.

### 3.3.2 Quality postprocessing

Earlier research showed that the quality of the source-separated output can drastically influence MIC performance [3, 15, 25]. When listening to the output of the model, very soft artefacts of the drum and bass stems that should have been removed were noticeable. Three independent people were asked to listen to the model's output and confirmed the observation. To remove some of these artefacts, a postprocessing pipeline was built that removed the softest sounds of the model output (less than 60dB). We believe these postprocessed stems have much better perceptual audio quality as there is a significant reduction in artefacts. Human evaluation is the most direct way to measure audio quality. However, it is time-consuming and expensive. Therefore, the objective performance of the raw and postprocessed model output will be measured and used to investigate the effect of separated stem quality on the MIC task.

## 3.4 Evaluation of source separation

In this section, the objective metric signal-to-distortion ratio (SDR), used to measure the performance of the raw and postprocessed source separation output, is discussed. SDR is frequently cited in audio source separation research and serves as the baseline measure for many source separation competitions (e.g., SISEC) [44]. SDR measures how much of the target signal is recovered, compared to the amount of noise produced by the generator model. Theoretically, values range from minus infinity to plus infinity but SDR scores in deep learning-based source separation usually lie between 5 and 15, with higher scores indicating better performance [46].

SDR reported in competitions is usually calculated directly on the waveform output [11, 36, 44, 46]. However, SDR measured on waveforms relies on having an exact alignment between the predicted and target waveform. We observed that the reconstructed waveforms did not satisfy this property. We believe the use of Griffinlim's algorithm to estimate the phase of the frequency spectrum could have caused this problem due to slight phase estimation errors. To still compare the performance between the raw and postprocessed source-separated stems, the SDR formula is applied to the amplitude spectrograms instead eq. (3.1)

$$SDR = 10\log_{10}\left(\frac{\|s_{target}\|^2}{\|s_{target} - s_{estimate}\|^2}\right) \tag{3.1}$$

where $s_{target}$ is the target amplitude spectrogram and $s_{estimate}$ is the raw or postprocessed amplitude spectrogram predicted by the model

## 3.5 Instrument classifier

In this section, we propose the convolutional neural network (CNN) that is used to classify the instruments playing in the audio segments of the IRMAS dataset. In addition, we showcase the parameters used to train the instrument classification model. To investigate the impact of deep learning-based source separation on MIC, three CNN models are trained and compared; the base, raw source-separated and postprocesed source-separated models. These instrument classifiers share many properties with the encoder of the spectrogram-based autoencoder as their architectures are rather similar. The base model is trained and tested on the original IRMAS dataset segments. The raw and postprocessed source-separated models are trained and tested on raw and postprocessed source-separated stems of the IRMAS dataset that were produced by the spectrogram-based autoencoder. We treat MIC as a multi-label classification problem where the CNN classifiers are trained to output all labels of the instruments playing in the audio segments. The CNNs take as input normalised logarithmic spectrogram representations and output scores ranging between 0 and 1 for each instrument. A decision threshold is then used to determine the existence of the instrument in the audio segment. An overview of our CNN model can be seen in fig. 3.3.

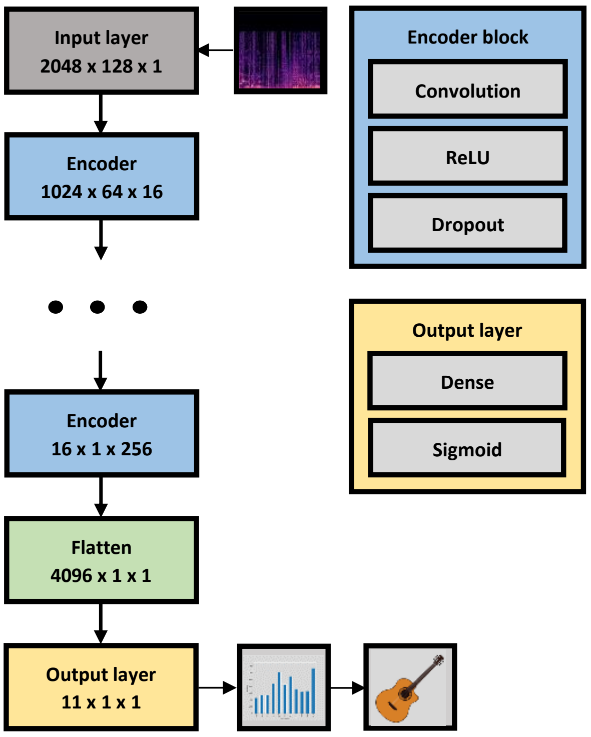


Figure 3.3: Convolutional instrument classifier

The CNN contains seven layers of convolutional blocks that compress the input spectrograms into a compact encoded representation. Each convolution block consists of a 2D convolution followed by batch normalisation and a ReLU activation function. After the seventh layer, we flatten the output. Next, a dense layer further reduces the dimensionality of the data. Finally, the sigmoid activation function generates independent class predictions for each instrument class.

### 3.5.1 Model parameters

Based on preliminary experiments, we found the following base parameters to perform best. The convolutional filters for each layer; (16, 32, 64, 128, 256, 512, 1024), kernel size; (7, 5, 3, 3, 3, 3, 3), and similarly, larger kernel sizes in the early layers resulted in faster convergence. The strides were 2 for each layer, serving as a downsampling mechanism. Dropout and weight regularisation were used to prevent overfitting. The convolutional and dense layers contain 20% and 50% dropout respectively as was also done in [26]. We then thoroughly experimented with the use of L2 weight regularisation. This hyperparameter significantly impacted the number of epochs required before convergence. L2 regularisation with a parameter of 1e-6 produced the best results for the base model, whereas the source-separated models did better with a setting of 1e-5. Similar to the autoencoder model, the instrument classification model's weights were initialised with a mean of 0 and a standard deviation of 1, which stabilised training and increased convergence speed. For the loss function, we used binary cross entropy, which generally works well for multi-label classification problems [26].

The models were trained on 1 GTX 960 GPU with 6 GB of VRAM. To find the best models, We implemented early stopping that monitored validation AUC with a patience level of 50. The evaluation metrics of the best models are calculated on the test set. One epoch is a pass over all 3-second audio segments in the dataset. The base, raw source-separated and postprocessed source-separated models were trained for 121, 173 and 98 epochs respectively. Each epoch took approximately 3 minutes, making the total train times per model range between 9 and 14 hours. The Adam optimiser was used with a learning rate of 3e-4 and a batch size of 16.

## 3.6 Evaluation of the instrument classifier

This section discusses the evaluation metric for the convolutional instrument classifier; area under the curve (AUC). We first describe the usefulness of this metric. Then an explanation of the technical details required to calculate the metric is given. Lastly, we motivate the reason to calculate instrument-wise AUC scores to measure the performance of the model.

AUC is a desirable metric for multi-label classification problems for two reasons: First, AUC is scale-invariant. It measures how well predictions are ranked rather than their absolute values. Second, AUC is decision-threshold-invariant. It measures the quality of the model's predictions irrespective of what classification threshold is chosen. The first property is useful as the exact prediction values of the instrument classifiers have no significant meaning. The second property is useful as it gives the end-user flexibility in taking the desired classification threshold that comes with a specific combination of true positive rate (TPR) and false positive rate (FPR). TPR and FPR can be best explained with the help of a confusion matrix. A confusion matrix summarizes the performance of a classifier in a table. For binary problems, a confusion matrix typically looks like table 3.1. As this research deals with a multi-label classification problem, the confusion matrices are computed for each instrument.

| | | Predicted class | | |
|---|---|---|---|---|
| | | Positive | Negative | Total |
| Actual class | Positive | $TP$ | $FN$ | |
| | Negative | $FP$ | $TN$ | |

Table 3.1: confusion matrix

**TP** True-Positive is the number of positive cases classified correctly

**TN** True-Negative is the number of negative cases classified correctly

**FP** False-Positive is the number of cases where the model predicted a positive class, but the actual class was negative.

**FN** False-Negative is the number of cases where the model predicted a negative class, but the actual class was positive.

The following metrics can be computed from the confusion matrix.

**True positive rate** eq. (3.2)
For a single instrument (e.g., guitar), out of all predicted positive cases, how many were correctly predicted.

$$TPR = \frac{TP}{TP + FN} \tag{3.2}$$

**False positive rate** eq. (3.2)
For a single instrument (e.g., guitar), out of all negative cases, how many were falsely predicted as positive.

$$FPR = \frac{FP}{FP + TN} \tag{3.3}$$

The output of the instrument classifier for a given instrument is a value between 0 and 1. A decision threshold that also ranges between 0 and 1 and splits the predictions into predicted positives and negatives. Given a decision threshold, the values in the confusion matrix can be calculated with the corresponding TPR and FPR scores. The ROC curve, which is plotted in a graph, showcases the relationship between the TPR and FPR of a classification model at all decision thresholds. AUC is the area under this curve and summarizes the performance of all possible TPR and FPR combinations into a single score. A classifier with high TPR and low FPR for multiple decision thresholds is desired and results in higher AUC scores. Depending on the classification task, AUC values range from 0 to 1. A perfect model has an AUC score of 1, meaning it has good separability for the different classes. A poor model has an AUC of 0.5, which means the model's class separation capacity is similar to that of chance. When a model has an AUC near 0, it means that the results are reciprocated. It predicts 0s as 1s and 1s as 0s.

Previous research that tested the effect of several music source separation algorithms on instrument classification, frequently mentioned that the impact of source separation highly depends on the characteristics of the instruments that had to be classified [1, 15, 25]. However, their evaluation scores only report macro-averaged AUC or f1 scores. Macro-averaged scores are calculated as mean scores of all the instruments in the dataset, without considering class balance. This is ideal for test sets that attempt to give scores that generalize into the real world. These averaged scores do not give insights into the effect of source separation per instrument. Therefore, we will showcase the instrument-wise AUC scores and analyze them to understand which instruments benefit most from source separation. The macro-averaged AUC scores will be used to calculate the model's overall AUC.

## 3.7 Validation analytical procedure

This section describes the analytical procedures used to test the research questions. First, a description of the statistical test used to compare the objective quality of the source separation algorithm is given. Next, the confidence interval-based mean difference score is explained that compares AUC scores between different models or instruments.

To test whether the objective quality differs between the raw and postprocessed source separation algorithms, an independent samples t-test is performed. This test compares the means of two independent groups in order to determine whether there is statistical evidence that the associated population means are significantly different. This research compares the raw and postprocessed source separation algorithm's SDR scores. To test whether the AUC of the source-separated instrument classifiers statistically differs from the base (non-preprocessed) classifier, a confidence interval-based mean difference analysis is conducted [2] that produces a distance score $D$. When $D$ is smaller than 0, it means that with a 95% confidence, it can be said that there is no positive difference in the mean AUC between the separated and base model. If $D$ is larger than 0, we can say with 95% confidence that there is a positive difference in the mean AUC between the separated and base model. The 95% confidence interval mean difference score $D$ is calculated as follows:

$$D = x_1 - (x_2 + se_{x1} + se_{x2}) \tag{3.4}$$

where $x_1$ is an individual instrument or mean AUC score of the raw or postprocessed instrument classification model, $x_2$ is an individual instrument or mean AUC score of the base model and $se_{x1}$, $se_{x2}$ are the standard errors of the 95% confidence intervals for the source-separated and base models[19].

To test whether music source separation impacts the performance of MIC, the difference scores for the macro-averaged AUC scores of the raw and postprocessed models are evaluated. For the instrument-wise analysis, the difference scores for each individual instrument are evaluated and compared between the source-separated models and the base model.

Chapter 4

# Results

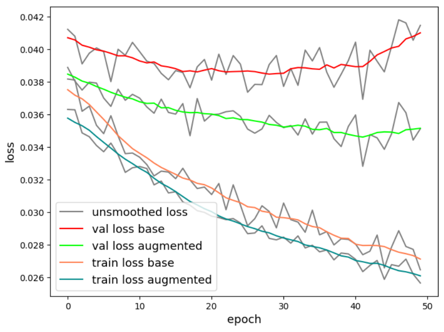
This chapter showcases the music source separation and MIC models' results, which will help answer the research questions. Starting with music source separation, the loss curves of the regularisation methods are visualised which were used to select the best hyperparameters for the final source separation model. Next, the comparative visualisations of the mixture, raw output, postprocessed output and target sources are shown for the MUSDB-18 test dataset. Finally, the objective quality of the two source-separated stems is compared using the independent samples t-test. The results of the instrument classification models start with loss curve visualisations of the small experiments with different regularisation methods. Afterwards, the ROC curves are shown that visualise the difference in AUC between the base, raw and postprocessed instrument classification models. Finally, the confidence interval-based difference scores are showcased that indicate whether deep learning-based source separation has an impact on MIC. Additionally, the instrument-wise analysis results are shown to get more insights into which instruments are affected most by source separation.

## 4.1 Source separation

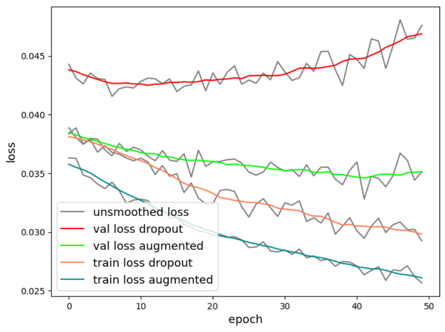
In this section, the results of the regularisation experiments are showcased that impacted the source separation models' validation loss and convergence speed. Next, the visualisations of the raw and postprocessed separation model output are displayed. Finally, the objective results of the raw and postprocessed source-separated stems are shown.

### 4.1.1 Model optimisation experiments source separation

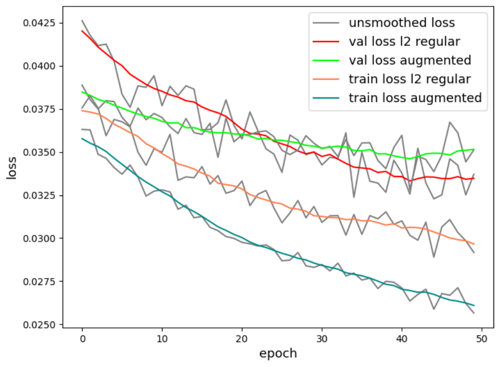
Severe overfitting was noticed when training the source separation models with the base parameters described in section 3.2.1. To counteract this, we experimented with three regularisation techniques; augmentation, dropout and l2 regularisation. In addition, the effect of truncated normal weight initialisation was also included in the experiments. The four above-stated methods were compared to decide which regularisation methods were implemented in the final source separation model. Each method was trained for 50 epochs while tracking validation loss. Figure 4.1a shows that adding augmentation decreased validation loss, supporting our decision to use it in our final model. Figure 4.1b shows that the addition of dropout to the augmented data caused the validation loss to increase significantly. Therefore, this regularisation method was omitted in the final model. Figure 4.1c shows that l2 regularisation (1e-4) on top of the augmented data starts with higher train and validation loss compared to only using augmented data. However, the validation loss becomes smaller after training for approximately 30 epochs, indicating that the model learns to generalise better. Therefore, l2 regularisation is also included in the final model. Lastly, fig. 4.1d shows a similar crossing for weight initialisation which supports our decision to implement the method into the final model. The loss curves of the final model can be seen in fig. 4.2. In fig. 4.2a, the validation loss decreases slowly compared to the train loss. However, the model keeps learning as the lowest validation loss is found at epoch 350 Figure 4.2b. Due to the long training time (8 minutes per epoch), training was stopped at epoch 375, and the weights of epoch 350 were used for the final model as they produced the lowest validation loss.



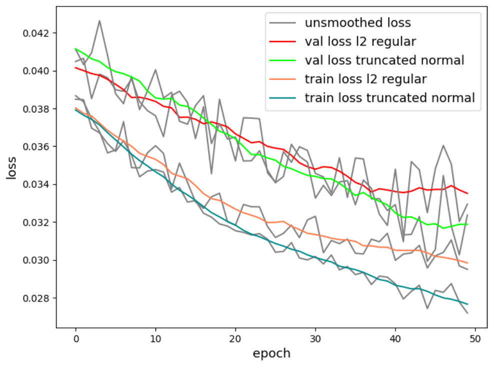
(a) Loss base and augmented



(b) Loss augmented and dropout

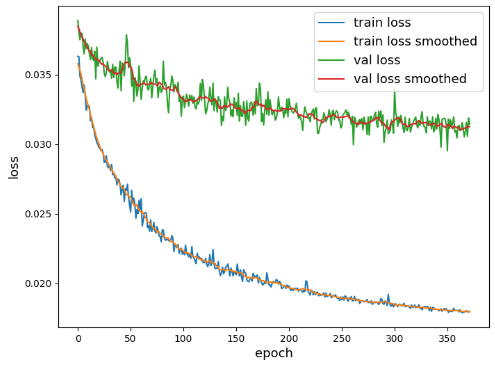


(c) Loss augmented and l2 regularisation

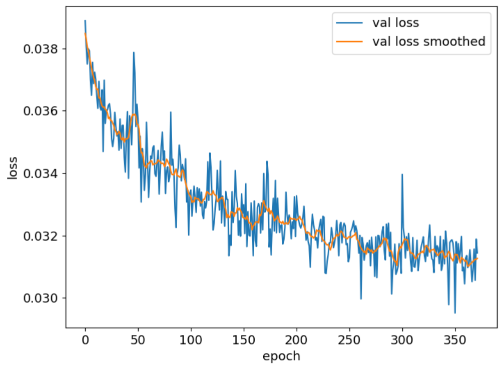


(d) Loss truncated normal and l2 regularisation

Figure 4.1: Loss curves of regularisation hyperparameters



(a) Loss final model train and validation

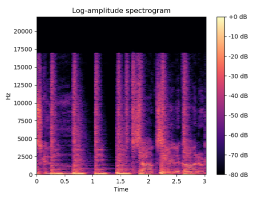


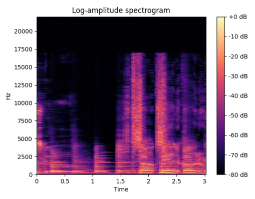
(b) Loss final model validation

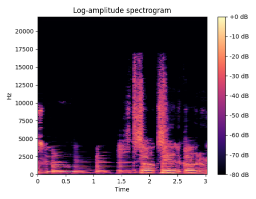
Figure 4.2: Loss curves of final model

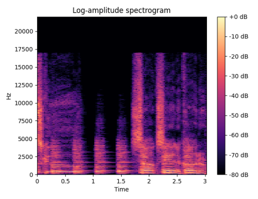
### 4.1.2 Source-separated spectrograms

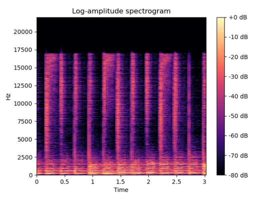
This section displays examples of the mixture, raw, postprocessed and target spectrograms in different tracks of the MUSDB-18 test database. The second row explicitly shows the intention behind the postprocessing algorithm. The errors in the amplitude spectrograms produced by the model mainly consist of soft sound artefacts from the removed drums and bass stems which can be seen in fig. 4.3f. Postprocessing removes all sounds softer than -60dB (purple color in spectrograms) fig. 4.3g. This removes most of the drum artefacts, but also some of the soft overtones produced by the instruments in the mixture.

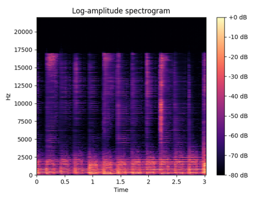

(a) Mixture track 1

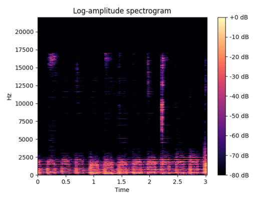

(b) Raw track 1

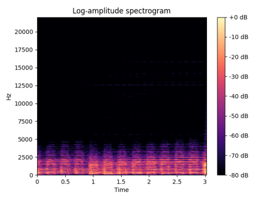

(c) Processed track 1

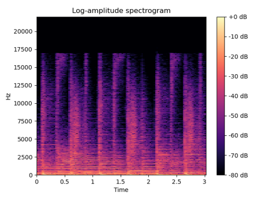

(d) Target track 1

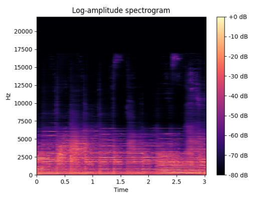

(e) Mixture track 13

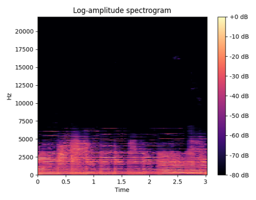

(f) Raw track 13

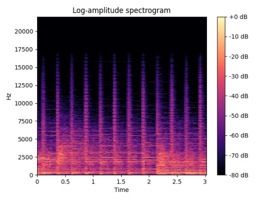

(g) Processed track 13

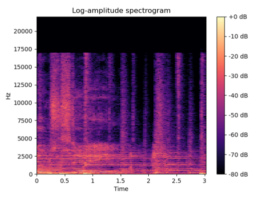

(h) Target track 13

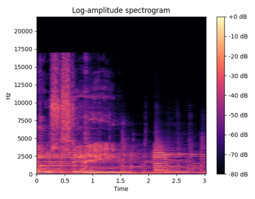

(i) Mixture track 21

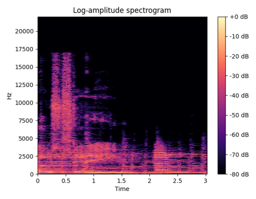

(j) Raw track 21

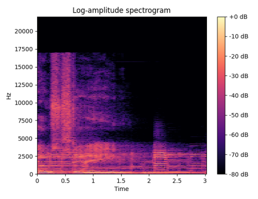

(k) Processed track 21

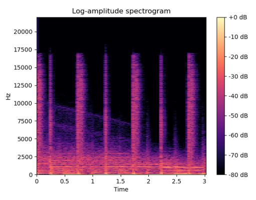

(l) Target track 21

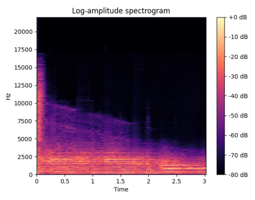

(m) Mixture track 32

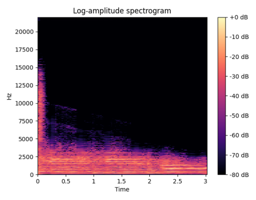

(n) Raw track 32


(o) Processed track 32


(p) Target track 32


(q) Mixture track 46


(r) Raw track 46


(s) Processed track 46

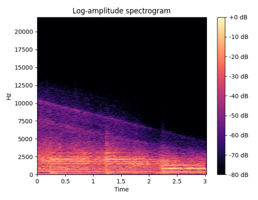

(t) Target track 46

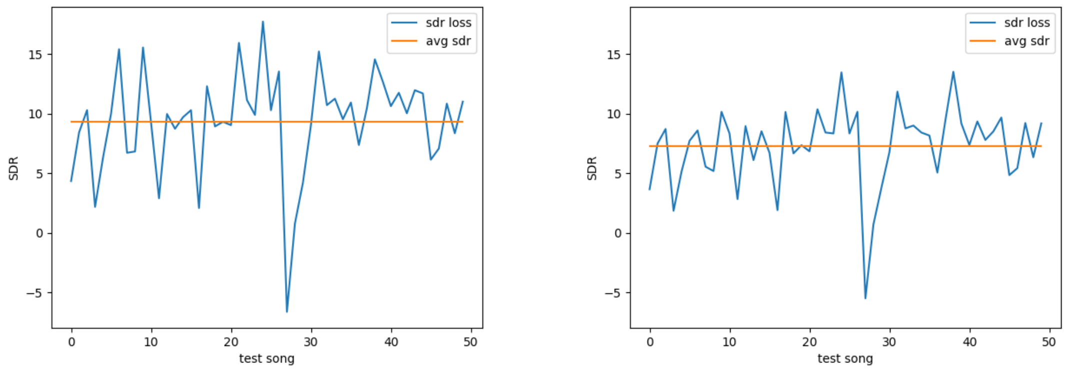
Figure 4.3: Examples of mixture raw processed and target spectrograms

### 4.1.3 Objective quality comparison of raw and postprocessed stems

To measure the objective quality difference of the raw and postprocessed source-separated stems, an independent samples t-test was conducted table 4.1. This test compares the raw and postprocessed stems' SDR scores on the MUSDB-18 test dataset. There was a significant difference in the scores between the raw separated stems (M=9.34, SD=4.29) and postprocessed separated stems (M=7.24, SD=4.28) conditions; t(98) = 2.69, p = 0.008. The raw separated stems produce higher SDR scores and thus have a higher objective quality than the postprocessed separated stems. Figure 4.4 showcases a graph of the SDR scores per song.

| Stem | Mean | SD | t-val | df | p |
|---|---|---|---|---|---|
| Raw | 9.33 | 4.29 | 2.69 | 98 | 0.008 |
| Post | 7.29 | 3.25 | | | |

Table 4.1: Independent samples t-test between raw and postprocessed source separated stems



(a) Raw separated SDR

(b) Postprocessed separated SDR

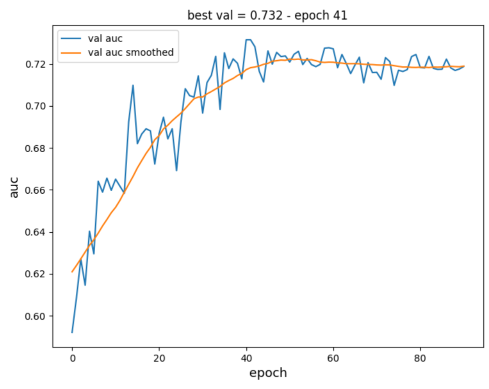
Figure 4.4: SDR per model for test tracks in MUSDB-18

## 4.2 Instrument classification

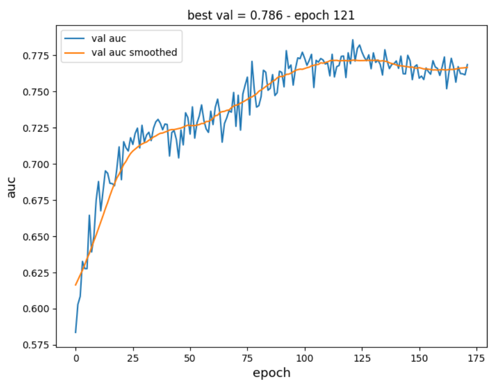
This section is structured as follows; first, the results of the weight regularisation experiments are showcased which were conducted to select instrument classification models with the highest AUC. Second, the highest AUC base, raw and postprocessed classification models' ROC curves are visualised. Finally, the results of the model and instrument-wise comparisons between the two source-separated models and the base model are presented.

### 4.2.1 Model optimisation experiments instrument classification

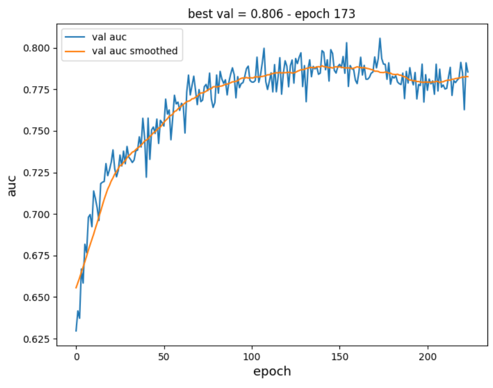
This section shows the L2 weight regularisation experiments that were conducted for three MIC model configurations: the base, raw and postprocessed model. For each model, the AUC curves for low (l2=1e-6) and high (l2=1e-5) regularisation were compared. The final classification models' l2 regularisation values were selected based on the highest AUC scores in these experiments. Higher regularisation means longer train time which can be seen in fig. 4.5. However, as train time was relatively low (3 minutes per epoch), we did not consider this as an important factor in selecting the best model. Figure 4.5b shows that the base model performed best with high l2 regularisation (AUC=0.732). Figures 4.5c and 4.5e show that both raw and postprocessed models obtained the highest AUC with low weight regularisation (raw AUC=0.807, postprocessed AUC=0.820).



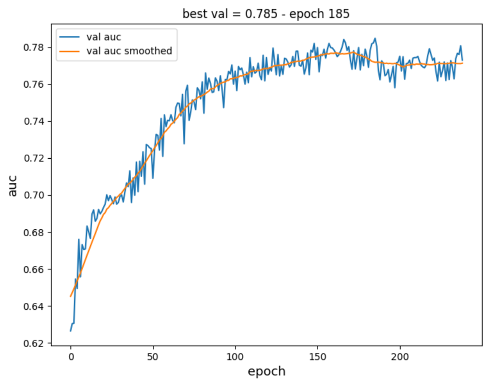
(a) Auc base model low l2 regularisation



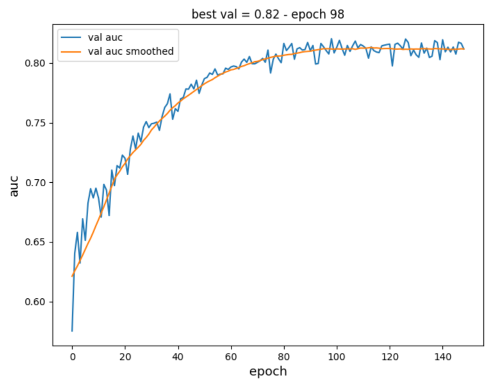
(b) Auc base model high l2 regularisation



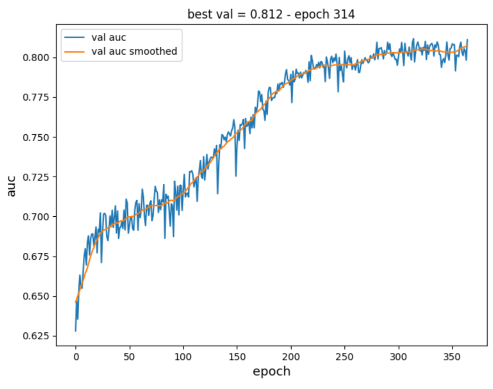
(c) Auc raw model low l2 regularisation



(d) Auc raw model high l2 regularisation



(e) Auc postprocessed model low l2 regularisation



(f) Auc postprocessed model high l2 regularisation

Figure 4.5: Test auc curves per model with low (1e-6) and high (1e-5) l2 weight regularisation

### 4.2.2 ROC curves for best models

The instrument-wise ROC curves can be seen in figs. A.1 and A.2 of Appendix A.

### 4.2.3 Objective performance instrument classifier

In this section, the results that answer the main research question are shown; does deep learning-based source separation have a positive impact on MIC. The left side of Table 4.2 showcases the AUC scores for each instrument classification model with the corresponding 95% confidence interval standard errors. The right side shows the difference scores $D$ between the base and the two source-separated instrument classification models, which were calculated with the formula described in section 3.7. The results show that the raw source-separated stems do not positively impact the performance of music instrument classification ($D_{raw} = -0.001$). The postprocessed source-separated stems do positively impact the performance of music instrument classification ($D_{post} = 0.015$). Lastly, we calculated instrument-wise difference scores to see which instruments benefited most from source separation. The right side of table section 3.7 shows that the cello ($D = 0.001$), piano ($D = 0.001$) and violin ($D = 0.001$) are positively impacted by the raw source-separated stems. The electric guitar, organ and trumpet were the only instruments that resulted in a raw loss of AUC for the raw classification model. The postprocessed source-separated stems positively affect the performance of the cello ($D = 0.021$), clarinet ($D = 0.048$) and piano ($D = 0.014$). For the postprocessed classification model, the flute, electric guitar and organ resulted in a reduction of raw AUC.

| Instrument | base | raw | post | D raw - base | D > 0 | D post - base | D > 0 |
|---|---|---|---|---|---|---|---|
| cel | 0.722 ± 0.052 | 0821 ± 0.046 | 0.839 ± 0.044 | 0.001 | T | 0.021 | T |
| cla | 0.784 ± 0.041 | 0.790 ± 0.041 | 0.904 ± 0.031 | -0.077 | F | 0.048 | T |
| flu | 0.763 ± 0.032 | 0.801 ± 0.031 | 0.737 ± 0.033 | -0.025 | F | -0.092 | F |
| gac | 0.724 ± 0.020 | 0.757 ± 0.019 | 0.758 ± 0.019 | -0.006 | F | -0.004 | F |
| gel | 0.780 ± 0.015 | 0.757 ± 0.015 | 0.762 ± 0.015 | -0.053 | F | -0.047 | F |
| org | 0.729 ± 0.023 | 0.706 ± 0.024 | 0.725 ± 0.024 | -0.070 | F | -0.051 | F |
| pia | 0.694 ± 0.016 | 0.726 ± 0.015 | 0.739 ± 0.015 | 0.001 | T | 0.014 | T |
| sax | 0.846 ± 0.024 | 0.856 ± 0.024 | 0.882 ± 0.022 | -0.038 | F | -0.001 | F |
| tru | 0.888 ± 0.030 | 0.879 ± 0.032 | 0.920 ± 0.026 | -0.071 | F | -0.025 | F |
| vio | 0.872 ± 0.020 | 0.911 ± 0.017 | 0.892 ± 0.019 | 0.001 | T | -0.018 | F |
| voi | 0.842 ± 0.013 | 0.861 ± 0.012 | 0.865 ± 0.012 | -0.005 | F | -0.002 | F |
| mean_auc | 0.786 ± 0.012 | 0.806 ± 0.009 | 0.820 ± 0.007 | -0.001 | F | 0.015 | T |

Table 4.2: Auc and difference scores for base, raw and postprocessed MIC models

# Chapter 5

# Discussion

## 5.1 Discussion

The goal of this research was to investigate whether deep learning-based source separation has an impact on music instrument classification. Additionally, we examined whether the quality of the source-separated stems influenced the performance of instrument classification and if certain instruments were more impacted than others.

First, a comparison was made between the objective and subjective quality of the raw and postprocessed stems produced by the spectrogram-based source separation algorithm. The results show that the raw stems have a better objective, but worse subjective quality than the postprocessed stems (fig. 4.4). As mentioned before (section 3.3.2), the raw source separation model significantly reduced the loudness of the drums and bass stems but could not remove them entirely. Soft artefacts of both drums and high percussion remained audible. The postprocessing algorithm removed all audible sound frequencies of the raw stems below 60dB. This meant that most of the soft artefacts from the drums (i.e., bass drum, timpani) and high percussion (i.e., cymbals, high hats) were removed, but also the soft overtones of other instruments were impacted. These overtones are more visible in the higher frequencies (section 2.1.1). However, due to the non-linear perception of frequencies in humans (section 2.1.1), the removal of the other instrument's higher frequency overtones makes subjective quality differences in this frequency range harder to detect. As the shape of the postprocessed stems' SDR scores is relatively similar to that of the raw stems (fig. 4.4), we think that a significant part of the reduction in the postprocessed SDR score can be attributed to the consistent removal of soft, high-frequency overtones from other instruments. These high-frequency errors in the postprocessed stem could have overshadowed the possible benefits that were made in the lower frequencies.

We then investigated the impact of the raw and postprocssed source separation stems on instrument classification by training three instrument classifiers. The base instrument classifier was trained and tested on the original IRMAS dataset segments. The raw and postprocessed source-separated classifiers were trained and tested on raw and postprocessed source-separated stems of the IRMAS dataset. The results show that only the postprocessed instrument classification model positively impacted the instrument classification performance table 4.2. These results are consistent with [3, 15], where it was mentioned that the separated stems' quality could drastically impact source separation performance on an instrument classification task. However, the quality level of a stem's lower frequency range might play a more important role in instrument classification than the overall quality of the whole frequency spectrum.

We carefully examined the instrument-wise performance on instrument classification between the two source separation models that differed in quality table 4.2. The results show that the cello and clarinet were the most positively impacted instruments by source separation, followed by the piano and violin. These findings do agree with prior research where it was found that the pitch-based solo/accompaniment produced better results for instruments with stable partials (woodwinds and strings) [15]. Three of our positively impacted instruments fall into these instrument categories. Perhaps, using timbre-based feature descriptors, such as the spectral envelope shape [37], could be used as conditional features to boost instrument classification for the other instruments.

However, our results also show that the most positively affected instruments (cello and clarinet), both have relatively low fundamental frequencies [30]. We think the effects of source-separated stem quality on instrument-wise classification potentially reveal important insights into the importance of sound quality in different frequency ranges. The longer sound waves of low-frequency sounds are more susceptible to wave interference from other sources [47]. We think the main reason that the postprocessed stems have drastically improved the performance on instrument classification for the cello and clarinet is that the post-processing additionally removed the drums and bass stems' artefacts, which could have interfered with the fundamental frequencies of these two instruments. The reduction of drums and bass artefact interference in this frequency range could make it systematically easier for the convolutional classifier to detect the instrument features. As errors in source separation remain high [11], it is essential to carefully consider which frequency range is considered to measure the quality of source separation algorithms. Future research is needed to investigate whether other low fundamental frequency instruments would benefit more from source separation in instrument classification tasks.

## 5.2 Limitations

The main limitations in this research consist of several computational issues, overfitting for the source separation algorithms and dataset problems for instrument classification. We also think improvements can be made in the spectrogram representation that was used for both models.

As discussed (section 1.2), the recent trend in machine learning is that bigger models produce better results [37]. This trend is also visible in music source separation competitions as the best model's [10] train and inference time is significantly slower than other models [20, 41, 45] Our proposed source separation model could have produced better stems if the model contained more convolutional layers and residual connections, but due to the already high training times, and computational memory limitations, we were not able to use these tools that have shown to increase separation performance [41]. Furthermore, our final spectrogram-based source separation model suffered from overfitting problems, which are visible in the train and validation loss curves of the final model (fig. 4.2). A possible cause for this observation can be the lack of enough regularisation. Due to computational training time constraints, our search for optimal regularisation parameters was shallow. We only tested the use of 50% dropout in the first three layers and did not experiment with smaller percentages. Additionally, our augmentation pipeline was static. We precomputed the augmentations, whereas augmenting on train time allows for a much richer augmented data representation [35].

A substantial problem for the instrument classifiers was the vast amount of monophonic train data in the IRMAS dataset. Prior research had already shown that training on polyphonic music improves the performance of polyphonic instrument classification [25, 27]. Future research could focus on augmentation methods that mix different monophonic segments to create polyphonic music. Some groundwork for these methods has already been proposed [26]. Additionally, the class imbalance of the test dataset resulted in some problems with the reliability of the low present instrument's AUC scores and their confidence intervals. We think a larger new dataset could drastically improve the performance of instrument classification algorithms.

Lastly, We used amplitude spectrogram representations with linearly-scaled frequencies as input representations for the convolutional classifiers [15]. However, the human perception of pitch is not linearly scaled; we are better at detecting pitch variations in the lower frequencies (section 2.1.1). Mel spectrogram representations, which have their frequency axis scaled based on human perception [26], would possibly be a better fit for both source separation and instrument classification. Prior research already used mel spectrograms for instrument classification [15]. However, as there is currently no efficient function to inverse the mel spectrogram to reobtain the waveform, no known source separation uses mel spectrogram representations as input. Exciting research that attempts to build efficient inverse mel spectrogram functions can be found here [49].

# Chapter 6

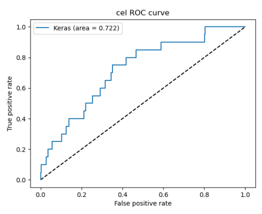
# Conclusion

In this work, we introduced a deep learning-based source separation algorithm and investigated the impact of the produced stems on music instrument classification. We performed several experiments to improve the performance of the source separation model on the MUSDB-18 dataset. To further increase the performance, we developed a postprocessing algorithm to compare the stems' performance with the raw output of the separation model. We found that the two stems differed in both subjective and objective quality. We then trained three instrument classifiers; the base, raw and postprocessed, which were used to investigate the impact of source separation on instrument classification. The models were trained on the IRMAS dataset. We found that only the postprocessed instrument classification model positively impacted the performance of instrument classification. Additionally, we performed instrument-wise analysis to examine which classified instruments were most impacted by music source separation. We found that the only instruments that benefited from source separation were the cello, clarinet, piano and violin. These findings suggest that the impact of source separation is both quality and instrument-dependent. End-users who want to enhance their instrument classification applications should be thoughtful about the quality level of their source separation algorithm and aware of the instruments to be classified.

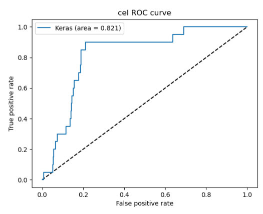
# Appendix A

# Visualisation of ROC Curves

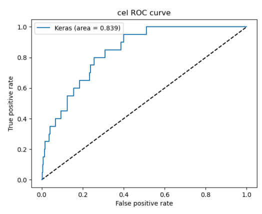
This appendix displays the instrument-wise ROC curves with corresponding AUC scores of the final base (left), raw (middle) and postprocessed (right) instrument classification models.



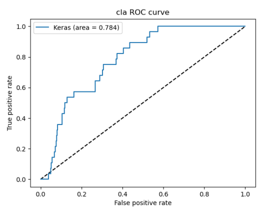
(a) Base cel 0.722 auc



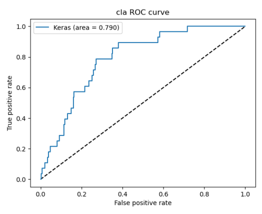
(b) Raw cel 0.821 auc



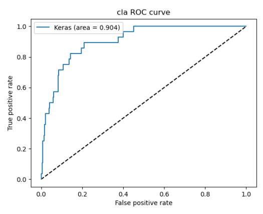
(c) Post cel 0.839 auc



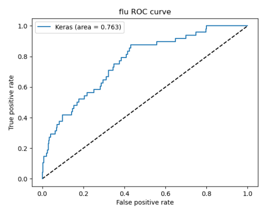
(d) Base cla 0.784 auc



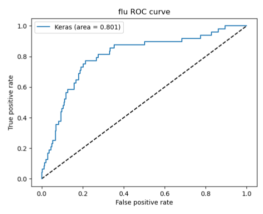
(e) Raw cla 0.790 auc



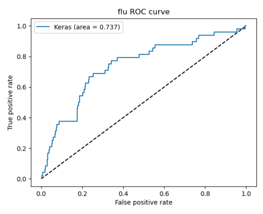
(f) Post cla 0.904 auc



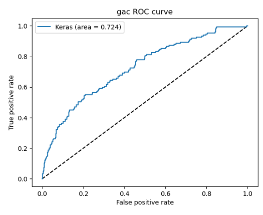
(g) Base flu 0.763 auc



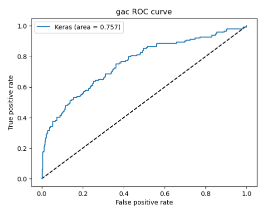
(h) Raw flu 0.801 auc



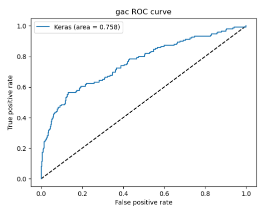
(i) Post flu 0.737 auc



(j) Base gac 0.724 auc

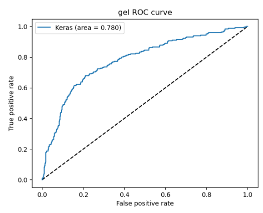


(k) Raw gac 0.757 auc

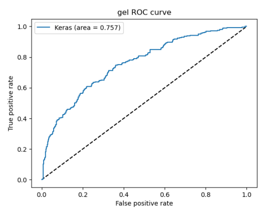


(l) Post gac 0.758 auc

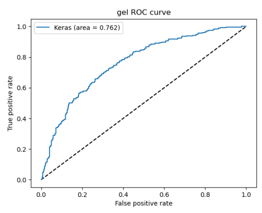
Figure A.1: ROC curves cel, cla, flu, gac



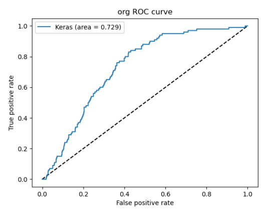
(a) Base gel 0.780 auc



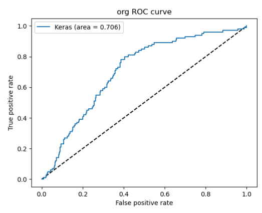
(b) Raw gel 0.757 auc



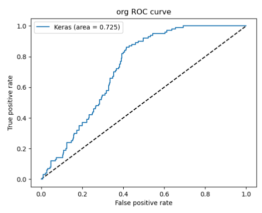
(c) Post gel 0.762 auc



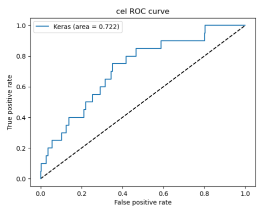
(d) Base org 0.729 auc



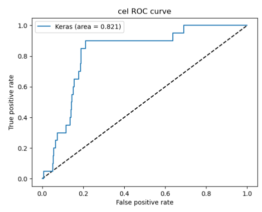
(e) Raw org 0.706 auc



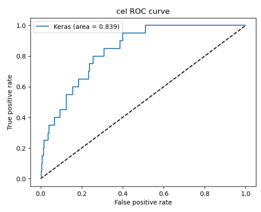
(f) Post org 0.725 auc



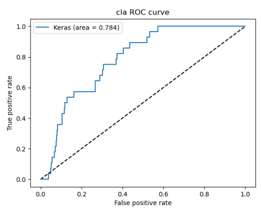
(g) Base pia 0.694 auc



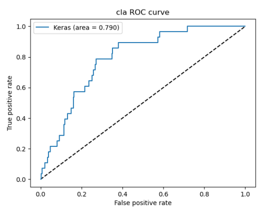
(h) Raw pia 0.726 auc



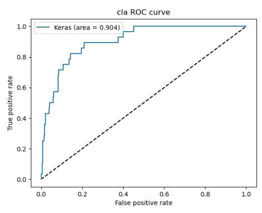
(i) Post pia 0.739 auc



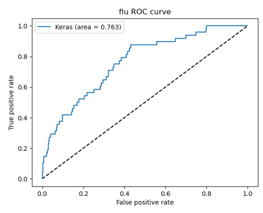
(j) Base sax 0.846 auc



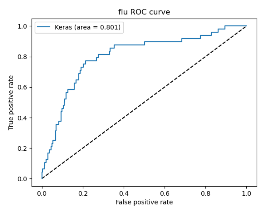
(k) Raw sax 0.856 auc



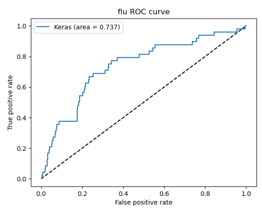
(l) Post sax 0.882 auc



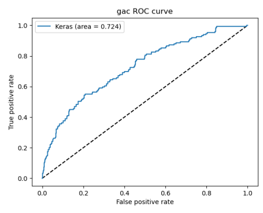
(m) Base tru 0.888 auc



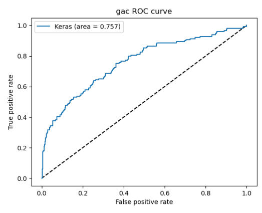
(n) Raw tru 0.879 auc



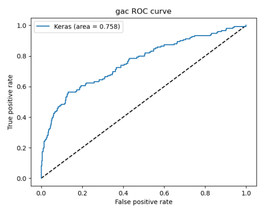
(o) Post tru 0.920 auc



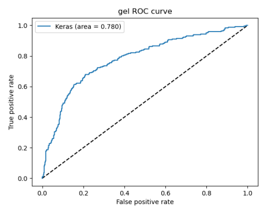
(p) Base vio 0.782 auc



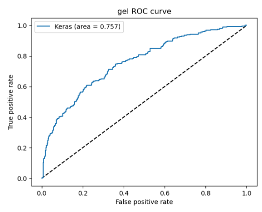
(q) Raw vio 0.911 auc



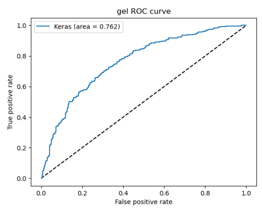
(r) Post vio 0.892 auc



(s) Base voi 0.842 auc



(t) Raw voi 0.861 auc



(u) Post voi 0.865 auc

Figure A.2: ROC curves gel org, pia, sax, tru, vio, voi

# Bibliography

[1] Roshni Ajayakumar and Rajeev Rajan. “Predominant instrument recognition in polyphonic music using gmm-dnn framework”. In: *2020 International Conference on Signal Processing and Communications (SPCOM)*. IEEE. 2020, pp. 1–5.

[2] James Algina, HJ Keselman, and Randall D Penfield. “An alternative to Cohen’s standardized mean difference effect size: a robust parameter and confidence interval in the two independent groups case.” In: *Psychological methods* 10.3 (2005), p. 317.

[3] Juan J Bosch et al. “A Comparison of Sound Segregation Techniques for Predominant Instrument Recognition in Musical Audio Signals.” In: *ISMIR*. Citeseer. 2012, pp. 559–564.

[4] Juan José Burred. “From sparse models to timbre learning: new methods for musical source separation”. In: (2009).

[5] Estefanıa Cano, Gerald Schuller, and Christian Dittmar. “Pitch-informed solo and accompaniment separation towards its use in music education applications”. In: *EURASIP Journal on Advances in Signal Processing* 2014.1 (2014), pp. 1–19.

[6] E Colin Cherry. “Some experiments on the recognition of speech, with one and with two ears”. In: *The Journal of the acoustical society of America* 25.5 (1953), pp. 975–979.

[7] Woosung Choi et al. “Investigating u-nets with various intermediate blocks for spectrogram-based singing voice separation”. In: *arXiv preprint arXiv:1912.02591* (2019).

[8] Woosung Choi et al. “Lasaft: Latent source attentive frequency transformation for conditioned source separation”. In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 171–175.

[9] Eileen Daniel. “Noise and hearing loss: a review”. In: *Journal of School Health* 77.5 (2007), pp. 225–231.

[10] Alexandre Défossez. “Hybrid spectrogram and waveform source separation”. In: *arXiv preprint arXiv:2111.03600* (2021).

[11] Alexandre Défossez et al. “Music source separation in the waveform domain”. In: *arXiv preprint arXiv:1911.13254* (2019).

[12] Ferdinand Fuhrmann and Perfecto Herrera. “Polyphonic instrument recognition for exploring semantic similarities in music”. In: *Proc. of 13th Int. Conference on Digital Audio Effects DAFx10*. 2010, pp. 1–8.

[13] William A Gardner. “Cyclic Wiener filtering: theory and method”. In: *IEEE Transactions on communications* 41.1 (1993), pp. 151–163.

[14] Christos Garoufis, Athanasia Zlatintsi, and Petros Maragos. “HTMD-Net: A Hybrid Masking-Denoising Approach to Time-Domain Monaural Singing Voice Separation”. In: *2021 29th European Signal Processing Conference (EUSIPCO)*. IEEE. 2021, pp. 341–345.

[15] Juan S Gómez, Jakob Abeßer, and Estefanıa Cano. “Jazz Solo Instrument Classification with Convolutional Neural Networks, Source Separation, and Transfer Learning.” In: *ISMIR*. 2018, pp. 577–584.

[16] Daniel Griffin and Jae Lim. “Signal estimation from modified short-time Fourier transform”. In: *IEEE Transactions on acoustics, speech, and signal processing* 32.2 (1984), pp. 236–243.

[17] Marvin HJ Gruber. *Statistical digital signal processing and modeling.* 1997.

[18] Yoonchang Han, Jaehun Kim, and Kyogu Lee. “Deep convolutional neural networks for predominant instrument recognition in polyphonic music”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 25.1 (2016), pp. 208–221.

[19] James A Hanley and Barbara J McNeil. “The meaning and use of the area under a receiver operating characteristic (ROC) curve.” In: *Radiology* 143.1 (1982), pp. 29–36.

[20] Romain Hennequin et al. “Spleeter: a fast and efficient music source separation tool with pre-trained models”. In: *Journal of Open Source Software* 5.50 (2020), p. 2154.

[21] Nori Jacoby et al. “Universal and non-universal features of musical pitch perception revealed by singing”. In: *Current Biology* 29.19 (2019), pp. 3229–3243.

[22] Umair Javed et al. “A review of content-based and context-based recommendation systems”. In: *International Journal of Emerging Technologies in Learning (iJET)* 16.3 (2021), pp. 274–306.

[23] Cyril Joder, Slim Essid, and Gaël Richard. “Temporal integration for audio classification with application to musical instrument classification”. In: *IEEE Transactions on Audio, Speech, and Language Processing* 17.1 (2009), pp. 174–186.

[24] Tero Karras et al. “Progressive growing of gans for improved quality, stability, and variation”. In: *arXiv preprint arXiv:1710.10196* (2017).

[25] Yoshiyuki Kobayashi. “Automatic Generation of Musical Instrument Detector by Using Evolutionary Learning Method.” In: *ISMIR*. 2009, pp. 93–98.

[26] Agelos Kratimenos et al. “Augmentation methods on monophonic audio for instrument classification in polyphonic music”. In: *2020 28th European Signal Processing Conference (EUSIPCO)*. IEEE. 2021, pp. 156–160.

[27] David Little and Bryan Pardo. “Learning Musical Instruments from Mixtures of Audio with Weak Labels.” In: *ISMIR*. Vol. 8. 2008, pp. 127–132.

[28] Yi Luo and Nima Mesgarani. “Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation”. In: *IEEE/ACM transactions on audio, speech, and language processing* 27.8 (2019), pp. 1256–1266.

[29] Mithun Madathil. “Music recommendation system spotify-collaborative filtering”. In: *Reports in Computer Music. Aachen University, Germany* (2017).

[30] Jürgen Meyer. “The sound of the orchestra”. In: *Journal of the Audio Engineering Society* 41.4 (1993), pp. 203–213.

[31] Aditya Arie Nugraha, Antoine Liutkus, and Emmanuel Vincent. “Multichannel music separation with deep neural networks”. In: *2016 24th European Signal Processing Conference (EUSIPCO)*. IEEE. 2016, pp. 1748–1752.

[32] George Philipp, Dawn Song, and Jaime G Carbonell. “The exploding gradient problem demystified-definition, prevalence, impact, origin, tradeoffs, and solutions”. In: *arXiv preprint arXiv:1712.05577* (2017).

[33] Renato Profeta and Gerald Schuller. “End-to-End Learning for Musical Instruments Classification”. In: *2021 55th Asilomar Conference on Signals, Systems, and Computers*. IEEE. 2021, pp. 1607–1611.

[34] Zafar Rafii et al. *The MUSDB18 corpus for music separation*. Dec. 2017. DOI: 10.5281/zenodo.1117372. URL: https://doi.org/10.5281/zenodo.1117372.

[35] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[36] Ryosuke Sawata et al. “All for one and one for all: Improving music separation by bridging networks”. In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 51–55.

[37] Markus Schedl, Emilia Gómez Gutiérrez, and Julián Urbano. “Music information retrieval: Recent developments and applications”. In: *Foundations and Trends in Information Retrieval. 2014 Sept 12; 8 (2-3): 127-261.* (2014).

[38] Bidisha Sharma, Rohan Kumar Das, and Haizhou Li. "On the Importance of Audio-Source Separation for Singer Identification in Polyphonic Music." In: *INTERSPEECH*. 2019, pp. 2020–2024.

[39] Christian Simmermacher, Da Deng, and Stephen Cranefield. "Feature analysis and classification of classical musical instruments: An empirical study". In: *Industrial Conference on Data Mining*. Springer. 2006, pp. 444–458.

[40] Arun Solanki and Sachin Pandey. "Music instrument recognition using deep convolutional neural networks". In: *International Journal of Information Technology* (2019), pp. 1–10.

[41] Xuchen Song et al. "CatNet: music source separation system with mix-audio augmentation". In: *arXiv preprint arXiv:2102.09966* (2021).

[42] Aleksandr Stikharnyi et al. "The Hybrid Intelligent Information System for Music Classification". In: *International Conference on Neuroinformatics*. Springer. 2019, pp. 71–77.

[43] Daniel Stoller, Sebastian Ewert, and Simon Dixon. "Wave-u-net: A multi-scale neural network for end-to-end audio source separation". In: *arXiv preprint arXiv:1806.03185* (2018).

[44] Fabian-Robert Stöter, Antoine Liutkus, and Nobutaka Ito. "The 2018 signal separation evaluation campaign". In: *International Conference on Latent Variable Analysis and Signal Separation*. Springer. 2018, pp. 293–305.

[45] Fabian-Robert Stöter et al. "Open-unmix-a reference implementation for music source separation". In: *Journal of Open Source Software* 4.41 (2019), p. 1667.

[46] Naoya Takahashi and Yuki Mitsufuji. "D3net: Densely connected multidilated densenet for music source separation". In: *arXiv preprint arXiv:2010.01733* (2020).

[47] Kenbu Teramoto, Tauhidul Islam Khan, and Sei-ichiro Torisu. "Acoustical blind source separation based on linear advection". In: *SICE Annual Conference 2007*. IEEE. 2007, pp. 157–162.

[48] Mikio Tohyama. *Waveform Analysis of Sound*. Springer, 2015.

[49] Kurt Anthony Weinheimer. "Efficient Audio Source Separation Using Mel-Spectrograms". PhD thesis. University of Maryland, Baltimore County, 2020.

[50] Norbert Wiener. "Norbert Wiener". In: *Information Theory* 4 (1919), p. 249.