Technische Universität Berlin

Institut für Kommunikation und Sprache
Fachgebiet Audiotechnologie

Fakultät I
Einsteinufer 17c
10587 Berlin
www.ak.tu-berlin.de

Master Thesis

Music Instrument Identification using Convolutional Neural Networks

Fabian Seipel

Matriculation Number: 372237
11.06.2018

Supervised by:

Prof. Dr. Stefan Weinzierl
Dr. Alexander Lerch

Hereby I declare that I wrote this thesis myself with the help of no more than the mentioned literature and auxiliary means.

Berlin, 11.06.2018

.....

(Signature [Fabian Seipel])

Abstract

Identifying musical instruments in polyphonic recordings with computational methods is a challenging but essential task in the domain of Music Information Retrieval. Such algorithms can improve music recommendation and genre recognition systems or inform other tasks like source separation, automatic score notation as well as music tagging. The following master thesis presents an Instrument Identification, or synonymously, an Instrument Recognition system for polyphonic classical music material including a pre-processing method to reduce crosstalk within multi-track recordings.

The above mentioned reduction method is able to adaptively estimate the crosstalk amount in the spectral domain and subsequently applies spectral subtraction to remove it. Different datasets with randomly generated crosstalk from anechoic classical music recordings were employed for development and evaluation purpose. The algorithm shows promising results regarding the crosstalk reduction performance while only producing a small amount of musical artefacts. In order to adjust the algorithm for different application purposes, the trade-off regarding reduction strength and musical artefacts can be adapted by parameter scaling. Besides its original function, to pre-process audio material for labelling, the algorithm can also be utilized as post production effect for multi-track recordings or as de-noising approach within multi-channel microphone arrays.

The instrument identification system is based upon a convolutional neural network which is trained on a dataset of multi-track recordings from different classical music concerts at the Berlin University of Arts. Those multi-tracks both contain spot microphone recordings as well as the individual stereo mixtures. By utilizing the above mentioned crosstalk reduction method, the spot microphone recordings are first pre-processed to determine the respective instrument activities. Those activities are then transformed to labelling information by thresholding. Input features for the convolutional neural net are extracted from the stereo mixtures in form of mel-spectrograms. Both labels and features are generated on a variety of different time frame sizes. After the training procedure, the classification model is able to predict the combination of instruments in a given classical music piece for each frame. Furthermore, the influence of frame size on the classification performance is investigated. Additionally, this work also evaluates two different methods based on sliding classification and multiple instance learning to train a classifier on large time frames but eventually predict on smaller time frames. In this way, imprecise labels may be employed to classify music recordings more accurately regarding the time resolution.

Zusammenfassung

Automatische Instrumentenerkennung in polyphonem Audiomaterial mit Hilfe von computergestützten Methoden gilt als anspruchsvolle Aufgabe im Bereich Music Information Retrieval. Diese Form von Algorithmen kann dazu beitragen andere Arten von Audioanalysesystemen zur Musikempfehlung oder Genreerkennung zu verbessern und dient darüber hinaus als Grundlage für Methoden zur Quellentrennung, automatischen Musiknotation oder zum Labeln von Audiodaten. Die vorliegende Masterthesis präsentiert ein solches Instrumentenerkennungssystem für polyphone klassische Musik. Dazu gehört ebenfalls eine eigens entwickelte Vorverarbeitungsmethode um Übersprechen auf Mehrspurmusikaufnahmen zu reduzieren.

Mit der oben genannten Methode wird zunächst der Betrag des Übersprechens im Frequenzbereich geschätzt um daraufhin mit spektraler Subtraktion das Übersprechen zu entfernen. Zur Entwicklung und Evaluation des Algorithmus wurden Orchesteraufnahmen aus reflexionsarmen Räumen herangezogen. Diese wurden in der Folge genutzt um verschiedene Datensets mit unterschiedlichen Anteilen von Übersprechen zu generieren. Testergebnisse zeigen, dass die Methodik in der Lage ist Übersprechen effizient zu reduzieren und dass das Audiomaterial dabei nur geringfügig beeinträchtigt wird durch musikalische Artefakte. Die Relation zwischen Betrag der Reduktion und Anzahl der musikalischen Artefakte lässt sich je nach Anwendungsfall mit verschiedenen Parametern steuern. Der Algorithmus kann neben seiner ursprünglichen Aufgabe als Vorverarbeitungsmethode um Audiodaten zu labeln auch als De-noising Methode für Mehrkanalmikrofonanordnungen oder als Audioeffekt in der Post-Produktion eingesetzt werden. Das Instrumentenerkennungssystem basiert auf einem sogenannten Convolutional Neural Network, das mit Hilfe von Mehrspuraufnahmen trainiert wurde. Die erwähnten Mehrspuraufnahmen sind von der Universität der Künste Berlin zur Verfügung gestellt worden und enthalten verschiedene Arten klassischer Musikkonzerte. Dabei handelt es sich sowohl um Spuren einzelner Instrumente wie auch deren Stereomischungen. Zunächst wurden die Instrumentenspuren mit der oben genannten Methode von Übersprechen befreit, um aus diesen im zweiten Schritt Labels zu generieren. Die entsprechenden Features werden als Mel-Spektrogramme aus der zugehörigen Stereomischung extrahiert. Der Vorgang wird in beiden Fällen für verschieden große Zeitabschnitte, sogenannte Frames, durchgeführt. Nach der Trainingsphase des Netzwerks ist das berechnete Modell in der Lage, die Instrumentenzusammensetzung für ein beliebiges Stück klassischer Musik pro Frame zu prädizieren. Darüber hinaus wurde auch der Einfluss dieser Framegröße auf die Klassifikationsergebnisse untersucht. In einem weiteren Experiment wurden zwei ähnliche Methoden analysiert, die es ermöglichen Modelle auf größeren solcher Frames zu erlernen, diese aber trotzdem fähig sind kleinere Frames zu klassifizieren. Dazu gehören „Sliding Classifications“ und „Multiple Instance Learning“. Somit können zeitlich grob aufgelöste Labels eingesetzt werden um letztendlich kleinere Zeitabschnitte zu klassifizieren.

Contents

List of Figures

List of Tables

1 Introduction

We can describe music by low level attributes such as key, tuning, tempo and instrument ensemble or try to categorize music on higher levels, for example in terms of genres or tags. While humans are relatively skilled to determine such attributes, deriving functions or models to discriminate and classify music by algorithms is significantly more challenging. However, this is what Music Information Retrieval (MIR) is concerned with. One such MIR task that is investigated in the present thesis is called Instrument Identification or, synonymously, Instrument Recognition (IR). In most cases, the input to MIR systems only consists of the most basic description of sound, a digitalized waveform. Because of recent technological advances, a plethora of audio nowadays exists in such a digital form in order to be stored or distributed over network systems. At the same time, machine learning methods have proven their ability to learn discriminations from big data, in a similar way that humans learn through experience. Combining both these elements, we can exploit the continuously rising amount of digital audio by analysis approaches from machine learning to automatically describe a particular audio waveform by higher level attributes and annotations without any prior knowledge. Deep learning techniques, a special form of machine learning, are particularly suitable for this task since they are able to create hierarchical models to explain data patterns, similar to the inherent hierarchical structure of music.

Motivation and Outline

This thesis is concerned with designing a deep learning classifier to identify instruments in polyphonic classical music data. Such a method can provide information to support other Music Information Retrieval tasks such as source separation and recommendation or constitute a fundamental building block of higher level systems for score annotation, musical retrieval, genre recognition or automatic tagging. Possible use cases include mobile applications to learn instruments, upmix systems to prepare stereo content to multichannel speaker systems as well as audio effects or remix methods for music production environments.

Acoustic musical instruments differ in a variety of physical principles which influence their unique sound; the playing interface, the sound generation method and the filtering effects of material and corpus shape. Altogether, these properties produce specific and directional time-frequency patterns and hence allow to differentiate between individual instruments. A polyphonic IR system is able to determine which instruments occur in a given time frame with a certain possibility.

The goal of this thesis is to develop and investigate such a system based on a convolutional neural network machine learning approach for classical music data. Points of interest include to which extend this system can achieve correct classifications depend-

ing on the individual instruments and the given dataset, how the analysis window or frame length influences the system performance and if it is possible to obtain predictions on small time frames even though the neural network system was trained on larger time frames. This could improve either the classification scores or time resolution of predictions from trained models for audio applications.

The dataset which was employed to develop and evaluate the above mentioned instrument recognition system comprises multi-track recordings from 116 classical musical pieces, interpreted at several concerts at the Berlin University of Arts. Those multitracks both include spot microphone recordings from individual instruments in the ensemble as well as the stereo mixture of the regarding pieces.

The overall IR systems includes a variety of processing steps. First, the spot microphone recordings are treated with a custom built algorithm to reduce crosstalk in order to prepare for the labelling process. This algorithm was developed and evaluated separately on an individual dataset of anechoic classical music recordings (see Section 3). After applying this reduction method, the processed multi-track audio was transformed into labelling information by frame-wise thresholding of their root mean square values. Features were extracted from the stereo mix of each recording in form of mel-spectrogram representations. Subsequently a convolutional neural net is trained on these features and labels which were generated on different time frames. After training the model, the classifier is able to determine the instrument composition for different time frames in a given classical music piece. Finally, the influence of frame width on classification results is further evaluated and discussed. The system also includes two other prediction approaches, sliding classification windows and a form of multiple instance learning, to evaluate if models which were trained on larger time frames can be utilized to predict the instrument activities on smaller time frames.

Chapter 2 introduces the basic concepts and mathematical foundations of deep learning in a compact form, followed by the presentation of the research field and applications of Music Information Retrieval, especially in combination with Deep Neural Networks. Chapter 3 presents an newly developed algorithm which is able to adaptively reduce crosstalk from multi-track recordings by spectral subtraction. This method can also be applied to other kinds of multi-channel input like microphone array systems or as post production effect for music recordings with crosstalk. In the following, Chapter 4 introduces the instrument recognition system. Finally, Chapter 5 rounds up this thesis by summarizing and discussing the present research and further gives an outlook about future work.

2 Fundamentals and Related Work

Techniques summarized as supervised machine learning are able to learn parameterized models from data. In a training phase, these models are tuned to find an optimal set of parameters by minimizing an objective function (i.e.error measure). Subsequently, the goal is to classify and predict unseen test data with the previously trained model. Such methods are especially valuable for problems that cannot be solved analytically but can be approximated iteratively. Machine Learning tasks differ from other optimization problems in a specific and important point: the key is to achieve high classification accuracies on new observations instead of maximizing performance on already present data examples to make profound predictions. In that case, the model is able to *generalize* well. If the model only shows good performance for the given training data, but generalizes poorly, this results in so-called *overfitting*. To prevent such overfitting, several *regularization* methods were designed to control the learning procedure (see Section 2.1.4). Besides supervised machine learning, several other techniques like unsupervised learning, transduction, transfer learning or reinforcement learning have been applied to different specific task. An in-depth description of machine learning concepts is out of this thesis' scope and can be found in Bishop [2007] and Duda et al. [2012].

Deep learning, a specific form of ML, employs mathematical models of neurons to form artificial neural nets which are particularly suitable to explain non-linear structures in large datasets. The application of above mentioned learning techniques has recently been pushed by technological advances such as improved computational capacities and efficient storage opportunities for big data. Machine and deep learning models have shown superior performance compared to former data analysis technologies and are state of the art in a variety of contexts such as computer vision, machine listening, natural language processing and speech recognition, recommender systems, machine translation, market analysis, robotics, self-driving vehicles, bioinformatics and many more. A specific audio-related research field which increasingly relies on machine learning methods is termed Music Information Retrival (MIR). This research area analyzes music in its raw (wave-)form or respectively another representation and extracts valuable information to structure and/or describe musical content.

In order to connect Music Information Retrieval tasks to Machine and Deep Learning, the following chapter first describes each topic in a compact manner to then fuse both subjects and then presents important concepts and studies which are related to the present thesis' topic of Instrument Identification.

2.1 Deep learning

Deep learning algorithms consist of multiple non-linear processing layers which are able to autonomously learn different representations of data with multiple levels of abstraction. In comparison, conventional machine learning methods require pre-engineered features to first transform the raw data into an interpretable representation for the learner, a time-consuming progress which demands for domain expertise. Such complex deep learning architectures comprise a plethora of stacked, simple but non-linear modules that each transforms the data representation to a higher and slightly more abstract level. For classification problems, higher levels of representation emphasize aspects of the data which are important for discrimination while suppressing irrelevant information. The motivation behind this approach is that many natural visual or auditive signals indicate such compositional low- to high-level hierarchies [Dieleman, 2015]. Images contain edges or motifs which in turn assemble to objects. Music signals are structured both harmonically and temporally. Frequencies and pitches in a tonal system form chords that arrange to progressions while note or drum sequences are rhythmic patterns. Lecun et al. [2015] and Schmidhuber [2015] provide a comprehensive overview about the topic of deep learning, whereas Goodfellow et al. [2016] cover all aspects in detail.

2.1.1 Neural Networks

Artificial neural networks are mathematical models inspired by biological neural systems. Each biological neuron receives input signals from its dendrites and produces output signals along its axon. The axon eventually branches out and connects via synapses to dendrites of subsequent neurons. Motivated by the structure of its biological equivalent, the mathematical formulation of a single artificial neuron is described as:

$$y = f(\sum_i w_i x_i + b) = f(\mathbf{w}^T \mathbf{x} + b) \tag{2.1}$$

where $f(\cdot)$ is a non-linear transfer or activation function and $\boldsymbol{\theta} = (b, \mathbf{w})$ are adjustable model parameters: the weight vector $\mathbf{w}$ and a bias term b. This single unit maps an input vector $\mathbf{x}$ to a scalar output value y by computing a weighted sum as the dot product of $\mathbf{w}^T\mathbf{x}$. Figure 2.1 shows a comparison of biological and artificial neurons.

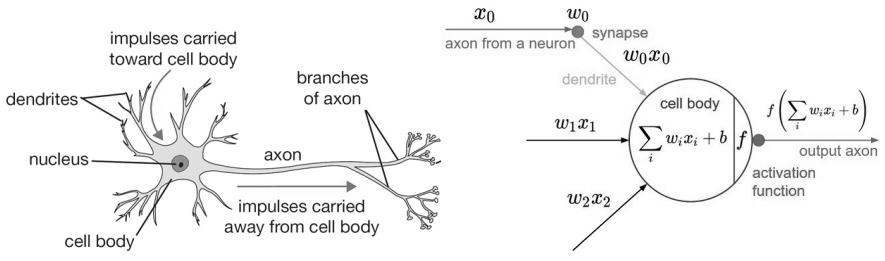


Figure 2.1: Comparison of biological and artificial neurons

Common choices for the activation function are *sigmoidal functions*, for example:

$$f(x) = (1 + e^{-x})^{-1} \tag{2.2}$$

$$f(x) = \tanh(x) \tag{2.3}$$

or *linear rectifications* as well as *leaky linear rectifications*:

$$f(x) = \max(x, 0) \tag{2.4}$$

$$f(x) = \max(x, 0.01x). \tag{2.5}$$

To obtain more complex input-output mappings, multiple neurons can be processed simultaneously and layered sequentially to form a neural net. Each unit contains its own parameters $\boldsymbol{\theta} = (\mathbf{w}, b)$. A parallel computation of neurons can be described as a matrix product of weights and inputs plus bias terms:

$$\mathbf{y} = f(\mathbf{W}^T\mathbf{x} + \mathbf{b}) \tag{2.6}$$

Stacking multiple layers or units requires to simply insert the previous layer's output into the next layer's input:

$$\mathbf{y} = f_2(\mathbf{W}^T f_1(\mathbf{W}^T\mathbf{x} + \mathbf{b_1}) + \mathbf{b_2}) \tag{2.7}$$

The data vector $\mathbf{x}$ is conventionally referred to as input layer while the outermost function is denoted as output layer. Any layers in between are called hidden layers. For a given task, the size of the output layer as well as the type of its transfer function $f(\cdot)$ are normally defined by the number of target classes and the problem statement.

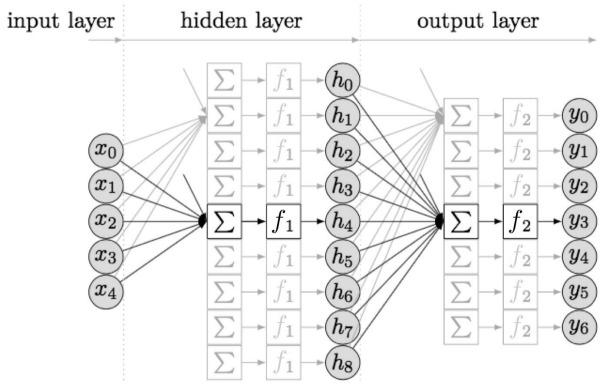


Figure 2.2: Neural net with one hidden unit (from Schlüter [2017])

The number of hidden layers, also referred to as the depth of the network, their respective size and the type of transfer function for each unit can be designed according to the problem definition. However, one specific activation function is usually chosen for all hidden layers for convenience. Figure 2.2 displays a simple neural net with one hidden layer to visualize the computation process.

2.1.2 Convolutional Neural Networks

Convolutional Neural Networks (CNN) or ConvNets are a variant of Neural Networks with two special types of layers: *convolutional layers* and *pooling layers*. These layers introduce constraint connectivity patterns within the CNN which are especially useful if local groups of the data are highly correlated like edges and motifs in images or harmonics of a tone in a spectrum representation of audio data. Additionally, CNNs are capable of detecting local statistics of data which are invariant to the overall location, for example similar or equivalent objects that appear in different parts of a picture or an audio signal.

Convolutional layers

ConvNets are designed to process multidimensional array data such as 2D pictures which contain multiple colour channels. A convolutional layer takes as input a stack of *feature maps*, like the pixels of those colour channels, and convolves each feature map with a set of learnable filters to obtain a new stack of output feature maps.

Starting from the mathematical formulation of an artificial neuron from Section 2.1.1, this can be implemented by replacing the matrix-vector product $\mathbf{W}^T\mathbf{x}$ with a sum of convolutions. The result of this weighted sum is then passed through a non-linear function $f(\cdot)$ to calculate the output feature map $\mathbf{Y}^{(l)}$, $l = 1...L$:

$$\mathbf{Y}^{(l)} = f\Big(\sum_{k=1}^{K} \mathbf{W}^{(k,l)} * \mathbf{X}^{(k)} + b^{(l)}\Big). \tag{2.8}$$

Herein, $\mathbf{X}^{(k)}$ denotes the set of K matrices with $k = 1...K$ which comprise the input feature maps, while the $*$ operator describes a two-dimensional convolution. Learnable parameters are represented by the weight or kernel matrices $\mathbf{W}^{(k,l)}$ including the filter coefficients and the bias terms $b^{(l)}$ for the respective output feature map l.

Figure 2.3 on the left shows the computation of a single output feature patch from a $5 \times 5 \times 3$ input feature map ($\mathbf{X}_0$,$\mathbf{X}_1$ and $\mathbf{X}_2$) by using a 3×3 filter. This filter is then shifted step by step over each possible position to iteratively obtain the output representation $\mathbf{Y}$ (see Figure 2.3 on the right). In comparison to a standard neural network, the convolutional unit restricts the connections in between layers to one filter per output map plus biases and thereby immensely reduces the number of parameters which acts as a strong regularizer that in turn prevents overfitting. Each filter operates as an individual feature detector for local patterns depending on the filter shape.

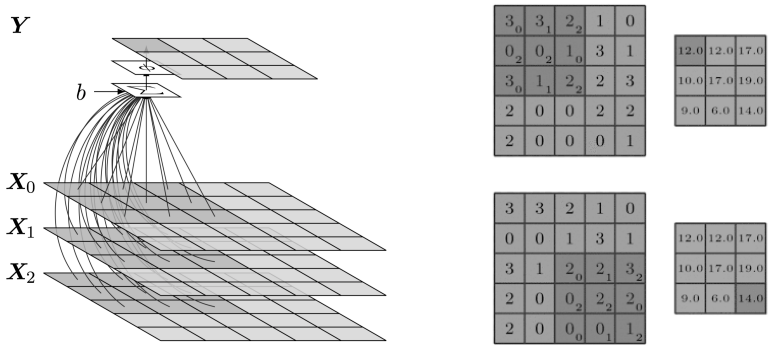


Figure 2.3: Visualization of the convolutional process (from Schlüter [2017])

There are a number of hyperparameters which determine the output feature map shape and hence need to be considered when designing convolutional layers. Increasing the filter size widens the receptive field which results in an extension of the spatial context per output patch. The number of filters specifies the depth of the output feature map. To decrease the spatial size of the resulting feature map, each filter can be shifted over the input with a larger step size, determined by the *stride* parameter. Using a stride of 2 with the example from Figure 2.3 would result in an output map size of 2×2. To prevent the network from decreasing in spatial size of the feature representation, the input can be *zero-padded* on each edge. Figure 2.4 shows the same 5×5 feature map and 3×3 filter from Figure 2.3, but this time using a stride of 2 while zero padding the input.

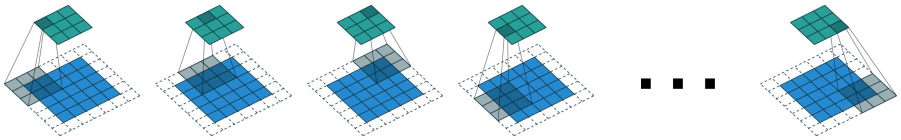


Figure 2.4: Filtering process with a stride of two and zero padding

Pooling layers

While the advantage of convolutional layers is to detect local data patterns, pooling layers are able to aggregate semantically similar features together. This allows to model correlations across a larger part of the input with the downside of decreasing the resolution. Reducing the dimensions in this way creates an invariance to translations or distortions of the input, a beneficial model attribute when classifying image or audio data. Pooling units typically compute the mean or maximum of a local patch. Figure 2.5 (left) shows a 2×2 *max-pooling* computation of a 4×4 input feature map, resulting in a 2×2 output feature map. As opposed to convolutional units, pooling layers do not

have any trainable parameters.

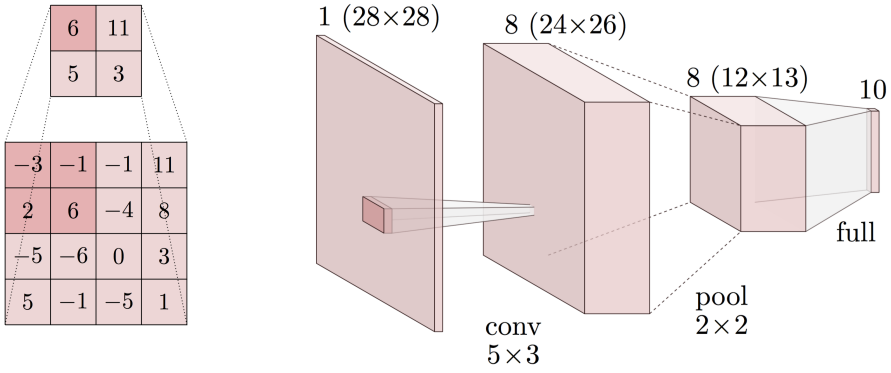


Figure 2.5: Max-pooling (left) and overall architecture of a simple CNN (right), (from Schlüter [2017])

Figure 2.5 (right) displays a simple CNN architecture. Usually, several convolutional units are stacked in the first part of a ConvNet before pooling layers then reduce the spatial network complexity. A fully connected layer at the end of the CNN maps all features to the desired output dimension. Feature maps are commonly visualized as 3D volumes. The design choices for ConvNets are highly dependent on the individual task. For larger classification tasks, CNNs sometimes contain more than 100 layers and exceed 100 million parameters to train. For most computer vision tasks and many audio recognition problems ConvNets represent state of the art algorithms.

2.1.3 Optimization

Like other machine learning techniques, the optimization of (Convolutional) Neural Networks is based on minimizing an objective function $J(\boldsymbol{\theta})$ with respect to the model parameters $\boldsymbol{\theta} = (\mathbf{W}, \mathbf{b})$ which in the case of CNNs consist of weights and bias terms. In order to adjust the parameter vector for an optimal solution, the learning algorithm calculates a gradient vector $\Delta\boldsymbol{\theta}$ that indicates in which direction and to what extent the loss changes for small derivatives of $\boldsymbol{\theta}$:

$$\Delta\boldsymbol{\theta} = \sum_{(\mathbf{x},\mathbf{t})\in\mathcal{D}} \frac{\partial}{\partial\boldsymbol{\theta}} J\Big(f(\mathbf{x}|\boldsymbol{\theta}), \mathbf{t}\Big) = \sum_{(\mathbf{x},\mathbf{t})\in\mathcal{D}} \frac{\partial}{\partial\mathbf{y}} J(\mathbf{y},\mathbf{t}) \frac{\partial}{\partial\boldsymbol{\theta}} f(\mathbf{x}|\boldsymbol{\theta}) \tag{2.9}$$

Herein, $\partial J/\partial\mathbf{y}$ describes the gradient with respect to the networks' outputs and $\partial f/\partial\boldsymbol{\theta}$ denotes all partial derivatives with respect to the model parameters. In order to associate the loss function (at the network's output) to the input data with respect to the individual neuron's weights, the chain rule can be applied by working backwards through the network. This method, called *backpropagation*, can be applied if both objective and activation functions are differentiable. It constitutes the mathematical foundation of the

learning procedure in neural networks. By presenting unseen input data to the learning algorithm, the weight vector $\boldsymbol{\theta}$ is then adjusted in the opposite direction of that gradient to iteratively minimize the loss function:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \Delta \boldsymbol{\theta} \tag{2.10}$$

This parameter update rule is known as the simplest form of *gradient descent* (GD). The negative gradient indicates the direction of steepest descent in the multi-dimensional parameter space. Updated values of $\boldsymbol{\theta}$ then need to be evaluated again regarding the loss function in a loop wise procedure until $J(\boldsymbol{\theta})$ eventually converges. In the above equation η denotes the *learning rate* which in turn controls how much the parameters change for each iteration step. The learning rate is a crucial but sensitive parameter. For large values, the convergence may be reached faster, however, if η is too high, $J(\boldsymbol{\theta})$ can start to oscillate or even diverge completely. A small value of η avoids such behaviour, but possibly leads too a very slow convergence process or even stop the optimization before a minimum is found. In most learning tasks, η is a hyperparameter that needs to be assessed carefully.

For practical reasons, due to computational complexity, the learning algorithm is only presented a often randomly chosen subset of the input data in each iteration step, a so-called *mini-batch*. If the loss and weight updates are calculated based on such mini batches, the optimization method is referred to as *stochastic gradient descent* since subsets of the input data can only produce a noisy estimate of the gradient in comparison to using all available data. Simple Gradient Descent, stochastic or not, is a relatively slow and unreliable approach to find global minima for $J(\boldsymbol{\theta})$. The method often fails to find global minima on the parameter space surface due to vanishing gradients. Several successors of GD have evolved, employing different techniques to efficiently find minima for the objective function:

- **Momentum** [Sutskever et al., 2013] proposes to add a short term memory to gradient descent, considering the previous steps to accelerate progress in stable directions:

$$\upsilon_{\boldsymbol{\theta}} \leftarrow \alpha\, \upsilon_{\boldsymbol{\theta}} - \eta \Delta \boldsymbol{\theta} \tag{2.11}$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \upsilon_{\boldsymbol{\theta}} \tag{2.12}$$

  $\upsilon_{\boldsymbol{\theta}}$ basically denotes a moving average over the gradient $\Delta\boldsymbol{\theta}$. In a more physical interpration of the momentum update, $\upsilon_{\boldsymbol{\theta}}$ can be referred to as a velocity when moving through the parameter space with $\Delta\boldsymbol{\theta}$ behaving as excitation force and α being a friction hyperparameter which dampens this movement.

- **Nesterov Accelerated Gradient** (NAG) [Nesterov, 2013] is a slight adaptation of the momentum update rule. Instead of evaluating the gradient at its current position $\boldsymbol{\theta}$, NAG calculates a modification of this gradient at the parameter space position $\boldsymbol{\theta}'$ that would be reached with the current velocity $v_{\boldsymbol{\theta}}$:

$$v_{\boldsymbol{\theta}} \leftarrow \alpha\, v_{\boldsymbol{\theta}} - \eta\Delta(\boldsymbol{\theta} + \alpha\, v_{\boldsymbol{\theta}}) \tag{2.13}$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + v_{\boldsymbol{\theta}} \tag{2.14}$$

  In comparison to the original momentum, the "look ahead" gradient evaluation within the NAG method dampens oscillations.

- **ADAM**, developed by [Kingma and Ba, 2014], computes a long-term exponential moving average of the gradient and scales the updates according to its reciprocal value. In this way, updates are scaled down in regions and dimensions where gradients are large and vice versa:

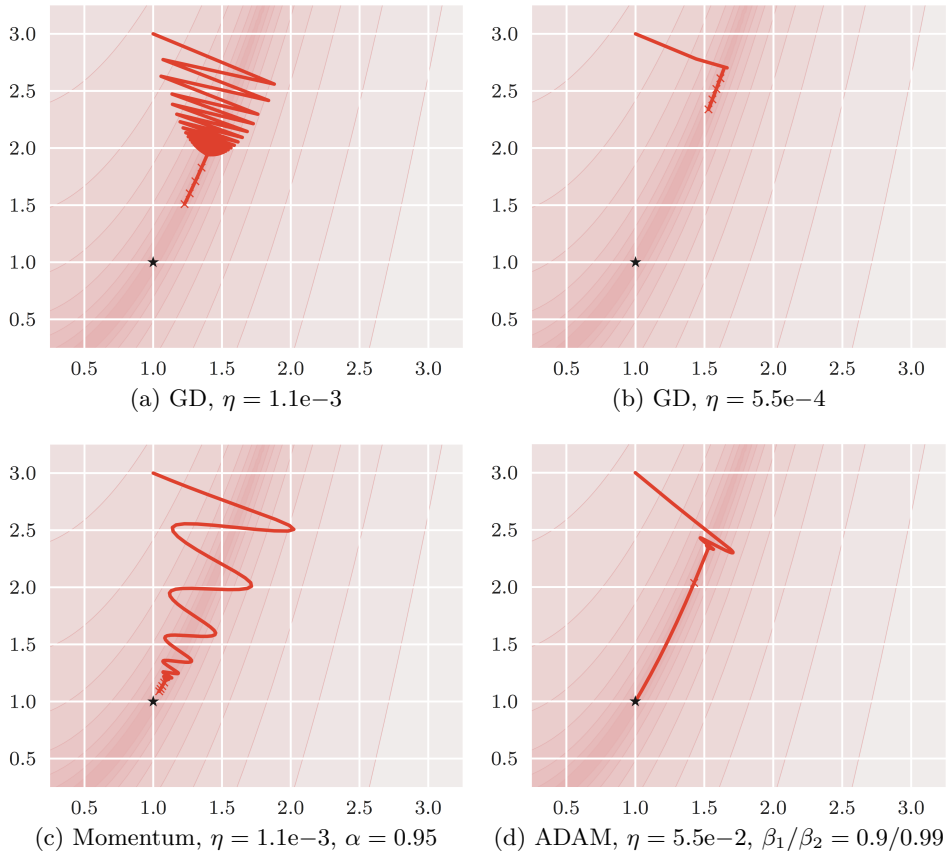
$$v_{\boldsymbol{\theta}} \leftarrow \beta_1\, v_{\boldsymbol{\theta}} - (1-\beta_1)\Delta\boldsymbol{\theta} \tag{2.15}$$

$$m_{\boldsymbol{\theta}} \leftarrow \beta_2\, m_{\boldsymbol{\theta}} - (1-\beta_2)(\Delta\boldsymbol{\theta})^2 \tag{2.16}$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\frac{v_{\boldsymbol{\theta}}}{\sqrt{m_{\boldsymbol{\theta}}} + \epsilon}\sqrt{1-\beta_2^t}/(1-\beta_1^t) \tag{2.17}$$

  Similar to the NAG/Momentum, $v_{\boldsymbol{\theta}}$ represents a first order moment whereas $m_{\boldsymbol{\theta}}$ describes the second order moment of gradients, both scaled by individual friction parameters β_1 and β_2. The addition of ϵ in the denominator, typically set to small values in the order of 1e$-$8, is supposed to avoid extreme scaling values. By formulating the update rule as a ratio of gradients, ADAM is invariant to rescaling of objective function or gradients and further ensures steady movement throughout the parameter space independent of its curvature and shape.

Figure 2.6 displays the typical behaviour of the presented optimization methods for the same function and feature space. While simple gradient descent oscillates heavily for higher learning rates (a), the step wise minimization converges too soon when choosing a low value of η (b). Momentum is able to dampen the oscillations and drives the solution towards the minimum by accumulating velocity (c). ADAM, by far the most efficient optimization technique in this case, both prevents the oscillation of the system and moves straightforward to the optimum (d). Besides that, ADAM also needs less than half of the iteration steps compared to each other method. Due to these advantages ADAM is frequently used in many neural networks as state of the art optimizer at the moment.



(a) GD, $\eta = 1.1\text{e}{-3}$ (b) GD, $\eta = 5.5\text{e}{-4}$

(c) Momentum, $\eta = 1.1\text{e}{-3}$, $\alpha = 0.95$ (d) ADAM, $\eta = 5.5\text{e}{-2}$, $\beta_1/\beta_2 = 0.9/0.99$

Figure 2.6: Comparison of different optimization methods (from Schlüter [2017])

2.1.4 Generalization

Similar to other machine learning techniques, the aim of utilizing neural networks for a given task is estimating predictions on unseen data with a computed probability or in a confidence interval. Deep neural networks offer several techniques to improve generalization. Two especially valuable of these techniques are presented in the following.

Dropout

For neural networks that are optimized via Gradient Descent, a method called *Dropout* has been proposed for generalization purpose [Srivastava et al., 2014]. In every iteration step of the training procedure, a predefined amount of the neurons' inputs in each layer are randomly set to zero. To compensate for those omitted units, the remaining neurons' inputs are scaled up so that the expected average input for each unit stays the same.

Figure 2.7 shows a three layer network that dropout was applied to. There are several reasons why generalization improves with this technique. Dropout can be interpreted as a specific variation of so-called *ensemble learning*, where a variety of different models

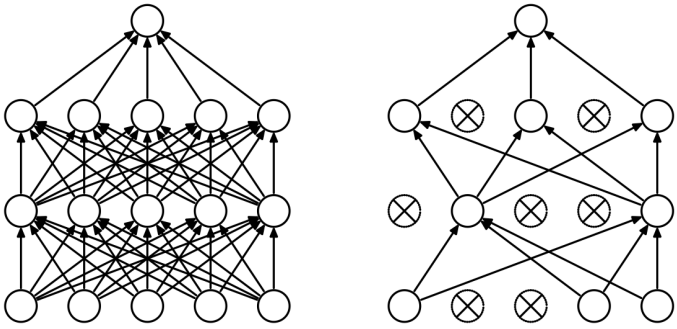


Figure 2.7: Small network with (right) and without (left) dropout (from Srivastava et al. [2014])

are trained in order to average predictions at the end. With the intended deactivation of neurons, the method prevents the network to focus on a small subset of features which possibly explain a large amount of variance in the training data. Using dropout, the network needs to learn more robust features that consider more units at the same time. Instead of concentrating on only a few neurons, the net has to be prepared to compensate for cancelled units. In other words, the network has to adopt multiple solutions to a problem which strongly supports the idea of generalization. Figure 2.8 displays the effect of dropout on training and validation loss. Both loss functions behaviours are noisy due to the stochastic approach within the dropout method. While the network starts to overfit after about 50 epochs for standard optimization, the validation loss flattens out with the dropout technique.

Data augmentation

A generalization technique that is particularly interesting for Convolutional Neural Networks consists of purposefully augmenting training data [Krizhevsky et al., 2012]. For

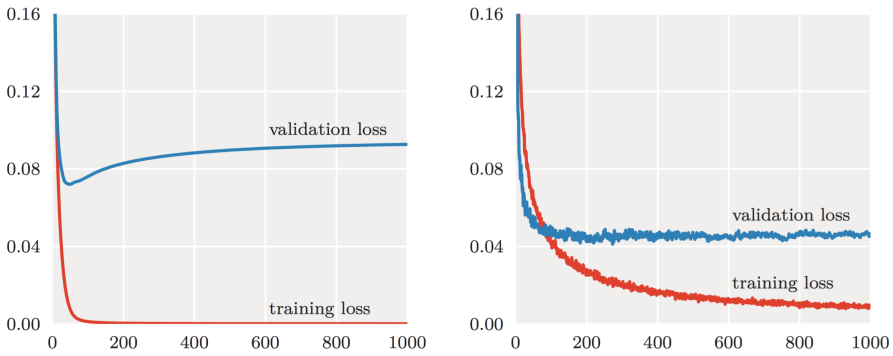


Figure 2.8: Training and validation loss without (left) and with (right) dropout as a function of epoch counts (from Schlüter [2017])

many learning problems, the amount and variety of available data is limited. One way to synthesize new training material is transforming the inputs in various manners. For image classification tasks, pictures can be rotated, translated, scaled, cropped, flipped or sheared while leaving the associated target label unchanged. Besides increasing the number of data samples without further acquisition of new material, augmentation also helps the model to become invariant to such transformations which again improves generalization. This method is particularly effective with CNNs since ConvNets are able to adapt to augmentations by exploiting convolutional pooling layers (see Section 2.1.2). The type of transformations, however, needs to be chosen carefully. Augmentations are ought to represent possible real world input observations. Extreme transformation can distort the desired input-output mappings.

This idea has been adapted for a range of audio classification tasks like blind source separation [Uhlich et al., 2017], acoustic event detection or environmental sound classification [Salamon and Bello, 2017] [Takahashi et al., 2016] [Piczak, 2015] [Su et al., 2017] [Virtanen et al., 2018], speech analysis [Cui et al., 2015] [Kanda et al., 2013] [Ragni et al., 2014], singing voice detection [Schlüter and Grill, 2015], instrument identification [Park and Lee, 2015], chord recognition [Humphrey and Bello, 2012] and several others. Common choices for transformations in the audio domain are *time stretching*, *pitch shifting*, *reverberation* or *room simulation*, *dynamic range compression*, *equalization*, *inserting background noise* and, in case of polyphonic source material, *mixture creation*. For efficient implementation of audio data augmentation the MUDA software framework [McFee et al., 2015] and the SCAPER python library [Salamon et al., 2017] were developed. Throughout all learning problems, data augmented training procedures result in higher system performance scores.

2.1.5 Validation

Since the actual achievement of machine learning models is to find a mapping from observations to target labels which can be applied to new data, it is of crucial importance to divide datasets in a training and test set to evaluate performances. While the former is essential to learn optimal model parameters $\boldsymbol{\theta}$, the latter is intentionally held out of the training procedure to obtain an unbiased estimate of the model's performance on unseen data. The training set itself is again split two fold, using one part, as above mentioned, to build the model and the second one, also known as validation set, to optimize so-called *hyper parameters*. In contrast to the model parameters $\boldsymbol{\theta}$, the setting of hyper parameters (e.g. the learning rate, the form of the loss function or the number of learning iterations, also called epochs) directly influences the training procedure and hence cannot be optimized during the learning phase. Given the fact that machine learning algorithms often have to be developed on relatively small datasets, this data three-fold sometimes is infeasible and impractical. Instead, a common practice in such situations is utilizing *K-fold crossvalidation*, a method to iteratively split training and validation set K times and perform the evaluation on each split. Figure 2.9 shows such a setting for

five folds. For this type of evaluation, stratification is advisable. This process denotes the distribution of data samples or observations over all folds in a supervised manner. In that way, each fold is representative of the dataset, for example by including the same amount of samples for each class. Within the crossvalidation technique, calculated performance metrics are then averaged over the the various folds.

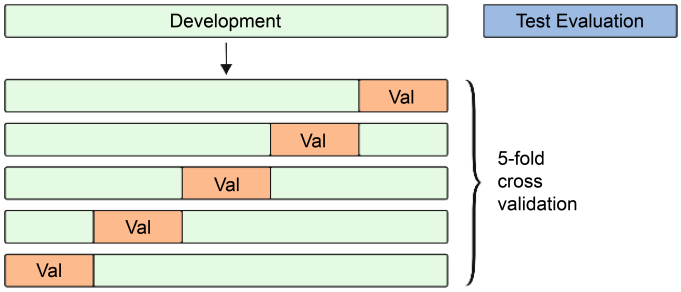


Figure 2.9: 5-fold Train-Validation split (from Virtanen et al. [2018])

Performance metrics
Evaluation is carried out by comparing the model's predictions to the actual ground truth or target labels of the test data. All necessary metrics for this evaluation can be calculated from the following intermediate statistics:

- *True positives* (TP): A correct prediction. Both model output and target label indicate that a certain class is present
- *True negatives* (TN): A correct prediction. Both model output and target label indicate that a certain class is not present
- *False positives* (FP): A false prediction. While the model output indicates that a certain class is present, the target label shows that the class is not present
- *False negatives* (FN): A false prediction. While the model output indicates that a certain class is not present, the target label shows that the class is present

Those statistics are exclusive, in other words, only one of each case can happen simultaneously in a binary classifier. Their total count sums up to the number of test data samples. Evaluation metrics are then derived from accumulated values of the intermediate statistics.

Accuracy measures how often the classifier makes a correct decision as the ratio of correct model predictions to the total number of test samples:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.18}$$

Precision represents the ratio of true positives to the total count of the model's positive predictions, while **Recall** stands for the ratio of true positives to the sum of true positives and false negatives. The **F-Score** is defined as the harmonic mean of precision and recall and serves as a valuable overall measure for many classification tasks:

$$P = \frac{TP}{TP + FP} \qquad R = \frac{TP}{TP + FN} \qquad F = \frac{2PR}{P + R} \tag{2.19}$$

F-Score is a commonly used and popular measure to compare classificators since it considers both precision and recall score at the same time. However, the F-Score does not take true negatives (TN) into account at all. For a variety of classification tasks, it is highly advisable to consider each metric individually to evaluate a model's performance.

ROC Curves and AUC

The receiver operating characteristic (ROC) and its corresponding area under the curve (AUC) offer the possibility to examine the performance of a binary classifier for a range of discrimination thresholds. A ROC curve displays the *true positive rate* as a function of the *false positive rate*, while the AUC in turn summarizes the ROC in a single value and thereby permits to compare classifiers across all operating points. Figure 2.10 shows the ROC for two different models and their corresponding AUC value. Higher AUC values indicate better performance. A ROC equivalent to the dotted grey line describes a random guess classifier.

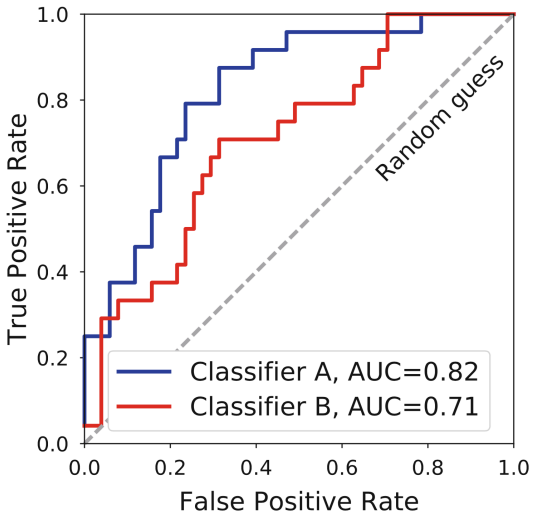


Figure 2.10: ROC and its respective AUC value for two classifiers (from Virtanen et al. [2018])

2.2 Music Information Retrieval

While human beings are highly skilled to recognize details from musical pieces like tempo, instrumentation or genre by mere listening and due to experience, this task is fairly difficult to formalize and then solve with the help of machines. Music can be represented in manifold ways. Low level representations include raw waveform or spectrograms while symbolic representations (MIDI files, score notation etc.) or meta data (e.g. composer, genre, tempo, key) are referred to as high-level representations, condensing the information in subordinate terms. The difference in between such abstraction levels is known as *semantic gap*. Music Information Retrieval systems focus on automatically bridging this gap by using computational methods. Extracting information from lower-level audio representations can have various reasons. Such applications include automatic transcription, genre classification, instrument identification, annotation, fingerprinting, recommendation, audio tagging and many others. A broader overview as well as more detailed information about the specific topics can be found in Müller [2007], Lerch [2012] and Schedl et al. [2014]

2.2.1 Audio signal representations

Most digital audio signals are representing continuous air pressure signals, captured by transducers such as microphones. The analog to digital conversion both includes a quantization in time and amplitude. This transformation, called *pulse code modulation*, stores a signal with a defined sampling frequency and bit depth in uncompressed formats like WAVE. These one-dimensional representations contain only few information compared to their storage size. Most MIR tasks, especially for deep learning, involve the transformation of raw wave forms to two-dimensional time-frequency representations.

Short time Fourier transform
The most common spectrogram representation is the short time Fourier transform (STFT) which indicates the time-varying energy across different frequency bands. STFTs are typically calculated for short periods of time, so called frames or windows of length N, to obtain the magnitude and phase information per frequency bin k for a certain time frame [Muller et al., 2011]:

$$X(k) = \sum_{n=0}^{N-1} w(n)\, x(n) e^{-j2\pi kn/N} \tag{2.20}$$

Herein x denotes a discrete time signal at sample point n and sampling rate f_s, $w(n)$ describes an N-point temporal window function. The corresponding frequency value for band k is determined by both f_s and window length N:

$$f(k) = \frac{k}{N} \cdot f_s \tag{2.21}$$

According to Equation 2.20, the maximum number of bands, also referred to as frequency bins, is always tied to the window length. For some applications, the window function is shifted step wise over the time signal with a so called hop-size which is commonly smaller than the window size. Consecutive frames are then concatenated to a spectrogram matrix. The STFT describes a linear, complete and invertible transform. An inverse STFT can obtain the time-domain signal from a spectrum representation of magnitude and phase.

Mel scale

In order to adapt this representation to the human perception of pitch, the linear STFT bands can be post-precessed to obtain other scales such as the mel scale. Using a bandpass filterbank of M bands, the frequency f can be mapped to the corresponding mel band m [O'shaughnessy, 1987]:

$$m(f) = 2595 \log_{10}(1 + \frac{f}{700}) \tag{2.22}$$

The mel scale represents perceptually equal pitch distances according to human listeners. Figure 2.11 displays the mapping from Hertz to mel scale as well as a triangular filter bank for a spectrum conversion to 8 mel bands.

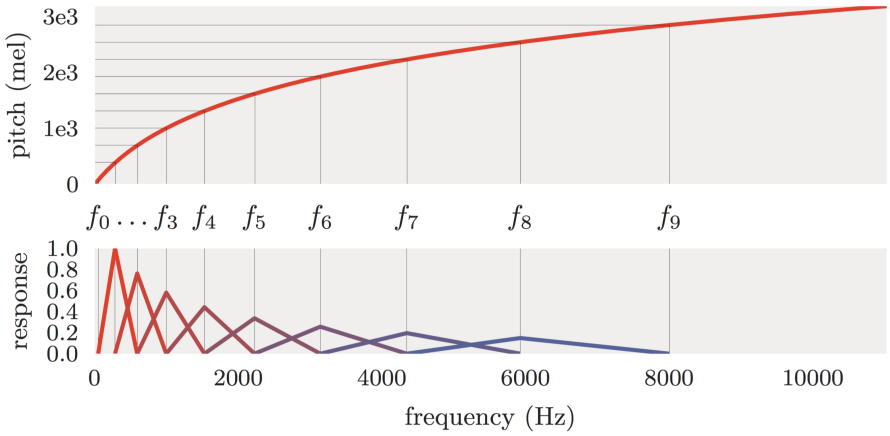


Figure 2.11: Hz to mel frequency mapping (top) and mel filter bank (bottom) (from Schlüter [2017])

Constant-Q transform

Another technique to compute a logarithmically scaled frequency representation that corresponds to the pitch scaling in western tonal systems is the constant-Q tranform (CQT) [Brown, 1991]:

$$X^{CQ}(k) = \frac{1}{N(k)} \sum_{n=0}^{N(k)-1} w(k,n)\, x(n) e^{-j2\pi Qn/N(k)} \tag{2.23}$$

By adapting the analysis window length $N(k)$ according to the bin frequency $f(k)$ (the relation from window length to frequency Q is constant), the CQT is able to obtain a higher resolution for low frequencies:

$$Q = \frac{f}{\Delta f} = \frac{1}{2^{1/c} - 1} \tag{2.24}$$

$$N(k) = \frac{f_s \cdot Q}{f(k)} \tag{2.25}$$

Herein, Q determines the frequency resolution and can be adjusted to match tonal scales, for example as a parameter that represents the number of bins per octave c. The CQT decribes a variation of the STFT which computes a spectrum only for specific, logarithmically spaced frequency bins. CQTs are practical tools to analyze pitch and chord structures. The variation of window lengths, however, also leads to blurring effects in the spectrum. While longer analysis windows in lower frequency ranges cause inaccuracies regarding the time domain (horizontal blurring), short analysis windows for higher frequencies entail inaccuracies regarding the frequency domain (vertical blurring). As opposed to the STFT, the constant-Q transform is not invertible. [Lerch, 2012]

Figure 2.12 shows three different spectral representations for two successive 15 second square wave signals. Both signal's fundamental frequency rises from $20Hz$ to $2000Hz$; the first one linearly, the second one logarithmically. Employing a $M = 128$ band filter bank according to Equation 2.23 yields the mel scale representation (middle). The bottom plot displays the result of a constant-Q-transform with $c = 12$ to obtain a representation which corresponds to equal tempered half tone spacing. Horizontal smearing effects for lower frequencies are due to longer analysis windows, whereas vertical smearing in higher frequency regions is caused by narrow analysis windows.

2.2.2 Engineered features

Previous to the possibility of feature learning by networks, high level representations of signals needed careful engineering and domain expertise. Most audio descriptors are derivations from time-frequency representations. Considering the magnitude spectrum at a single time frame as a probability distribution over the individual bins allows to describe the spectral shape with statistical properties. Such descriptors include the *spectral centroid, spread, skewness* and *kurtoises* which are defined as the first to fourth order central moment of this distribution. *Spectral rolloff* denotes the 85- or 95-percentile of the accumulated STFT magnitudes, while *spectral slope* is the first coefficient a for a linear regression fit to the magnitude spectrum: $S(k) = ak + b$. [Lerch, 2012]

Mel-Frequency Cepstral Coefficients (MFCCs) have been extensively used in the domain of speech recognition during the last 30 years. By applying a *Discrete Cosine Transform* to the logarithmically scaled mel spectrum, they provide a compact representation which contains most principal timbre information. The number of utilized filter components varies depending on the application purpose. Common choices range in between 20 to 40 coefficients. [Davis and Mermelstein, 1990] [Logan and others, 2000]

Chroma vectors are 12-dimensional magnitude vectors that represent the accumulated magnitude in each pitch class of the western equal-tempered scale. They display the distribution of those individual classes in a histogram form. Originally introduced by [Wakefield, 1999] and [Fujishima, 1999], the chroma vector computation has been adapted and optimized in several ways. [Lerch, 2012] gives a detailed overview of the manifold variants.

Delta and acceleration features can be computed from any above shown audio representation. They denote the difference of a certain feature in consecutive frames (delta) or, in turn, the difference of the respective deltas in consecutive frames (acceleration). Both measures assess temporal short-term fluctuations which are often evaluated in parallel to the basic features in focus.

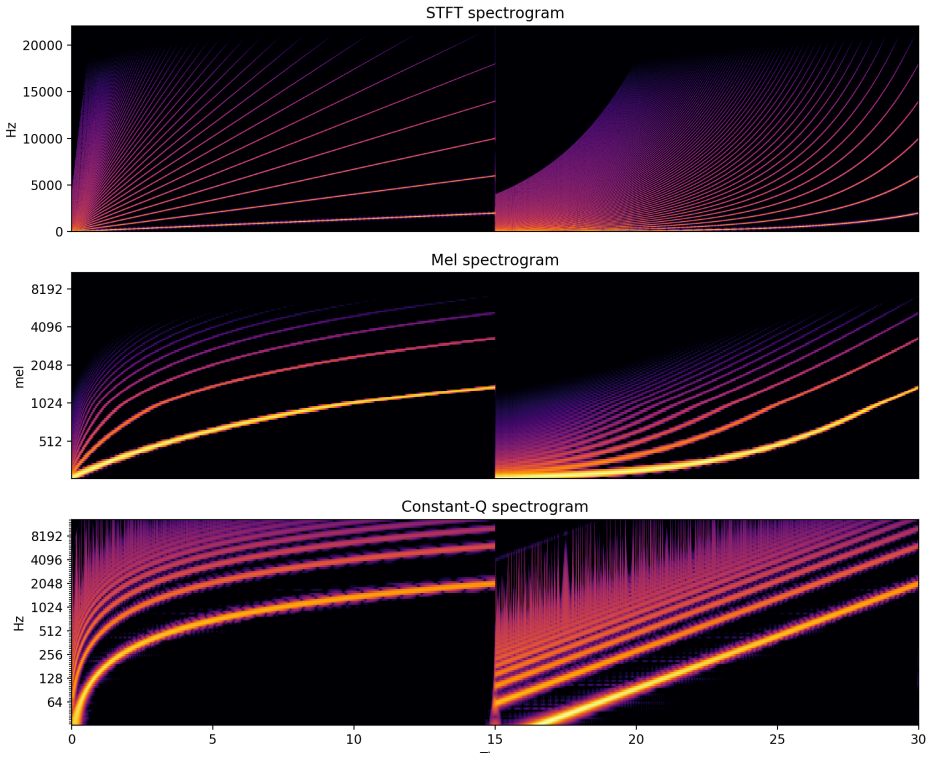


Figure 2.12: Comparison of different spectrogram computations

2.2.3 Deep learning for MIR

Music audio as well as deep learning share a common property: They both incorporate hierarchical structures [Dieleman, 2015]. Learning approaches can exploit such hierarchical models by stacking non-linear network layers to learn abstractions of feature representations. The learning procedure is then able to identify those contextual dependencies and map the respective model to the data structure at hand for a given task. In the MIR domain, CNNs have proven successful to model such connections due to robustness regarding variations in the data.
Before the advent of deep learning techniques, audio data needed to be first analysed to manually engineer representations as described in Section 2.1.2. Feature engineering demands for high expertise in the respective research field. While hand-designed features have been employed successfully for a range of problems, they have also reached a performance limit [Humphrey et al., 2012] [Humphrey et al., 2013]. However, the ability to learn feature representations comes with a certain trade-off. Learning approaches, especially convolutional neural networks, demand for large datasets, high computational resources and expertise regarding machine learning to effectively train models. Nevertheless, recent developments both for image as well as audio related tasks have shown superior results for deep learning techniques. Choi et al. [2017b] provides a comprehensive tutorial on the topic, including a current literature overview.[1] The research work of Dieleman [2015], Schlüter [2017] and Humphrey [2015] all contain valuable insights into deep learning for MIR with different individual focus areas.
The optimal employment of such techniques includes various interdependent aspects. Many advances in the field of computer vision have been adapted for music information retrieval tasks while other design principles still need to be assessed individually for the classification of audio data. Most deep learning approaches rely on either raw audio waveforms (end-to-end) or spectrograms features (mid-level representation) as input.

2.2.3.1 End-to-end learning

When working towards image classification, processing raw data in form of pixels has been established as state of the art. Full end-to-end learning includes all methods that operate directly on unprocessed data inputs. In the audio domain, such approaches then apply models on individual samples of the discretized waveform. By removing the spectrum calculation step, the feature learning process can also gain insights about what kind of information are salient for a given task and to further understand inherent data structures. Dieleman and Schrauwen [2014] have first explored this approach to music audio by comparing end-to-end to spectrum representation learning with ConvNets. Although the latter showed inferior results for this particular task and dataset, the network was able to discover frequency decompositions on its own. Figure 2.13 displays a subset of trained frequency-selective filters on raw audio for the lowest network layer.
Since learning on raw audio retains all entailed information in the data, including the

[1] An extensive and updated list of the latest Deep Learning research in the field of MIR is provided at https://github.com/ybayle/awesome-deep-learning-music.

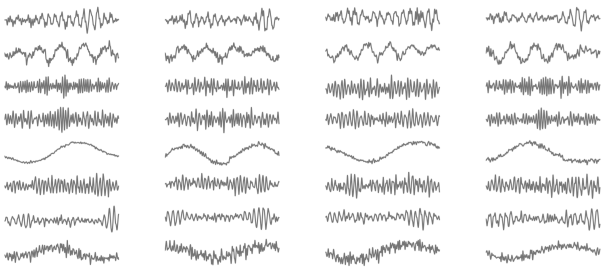


Figure 2.13: Subset of learned representations from raw audio (low-pass filtered)

phase, this approach is also highly appealing with regard to generative models in order to synthesize audio signals. Subsequent studies on learning on sample level have shown promising results for synthesis systems concerning text-to-speech (TTS) systems [van den Oord et al., 2016] [van den Oord et al., 2017] [Mehri et al., 2016] or musical instrument notes [Engel et al., 2017] [Mor et al., 2018]. Despite comparably good audio quality and flexibility, training such models demands for very high computational capacities that only few researchers have access to. Other approaches which exploit raw audio for classification instead of (re)synthesis have been catching up with spectrogram based learning for a variety of MIR tasks, yet still remain inferior [Tüske et al., 2014] [Golik et al., 2015] [Ghahremani et al., 2016] [Sainath et al., 2015] [Dai et al., 2017] [Lee et al., 2017]. Only for larger datasets, end-to-end approaches can achieve similar performance results [Pons et al., 2017].

2.2.3.2 Mid-level representation learning

For most MIR tasks, feature learning and classification still relies on mid-level representations in form of spectrograms. From a biological point of view, the transformation to spectrograms is more than reasonable: In advance to reaching the human brain, the cochlea processes audio signals by applying a filter bank in a similar manner. Converting audio to image-like features in form of spectrograms furthermore simplifies the adaptation of computer vision learning methods. The most commonly used input features are STFTs and mel spectrograms (see Section 2.2.1). While STFT representations also contain phase information which can be employed for a sonification of layers via retransformation to the audio domain [Choi et al., 2016], mel spectrograms only preserve magnitude values. However, most studies report best performance results for the latter or other logarithmically scaled representations [Choi et al., 2017b]. Such compressed and compact frequency scales allow for capturing overtone structures in convolutional layers with smaller receptive fields since filter weights can be shared within a larger range of frequencies and pitches. CQT spectra are especially helpful for tasks like chord recognition. Herein, harmonic relations in equally tempered tuning show similar spectral profiles independent from the frequency region. STFTs often contain a lot of redundant information for classifying music audio. Learning problems which involve noise-like

material, for example environmental audio, can benefit from linear spectrum representations since such sound profiles are less related to harmonic structures and often involve a broader frequency range [Virtanen et al., 2018].

Preprocessing the magnitude values of spectrograms can further improve the classification results. Such steps include spectral whitening [Lee et al., 2009] or logarithmic mapping [Choi et al., 2017a] and the subsequent standardization to zero mean and unit variance [LeCun et al., 1998]. Besides the fact that logarithmic magnitude values correspond to the human loudness perception, Schlüter [2017] notes that a transformation to the logarithmic scale also results in a rather gaussian-like distribution of magnitude values which could possibly improve the learning procedure. For a more extensive explanation of feature representations and their pre-processing steps, also containing a list of related MIR studies, see Schlüter [2017, pp. 69] and Choi et al. [2017b].

The size of time-frequency input representations depends on the prediction time scale. For systems which have to classify data in real time, the input audio length cannot exceed the maximum allowed system latency, most often a single frame. In this case, the model architecture is restricted to 1-D convolutions along the frequency axis, ignoring temporal relationships if not incorporating any long or short term memory to the model in form of recurrent units.[2] Without depending on real-time predictions, several frames can be concatenated to time-frequency matrices, henceforth called blocks. The choice of frame and block length depends on the task at hand. While smaller analysis frames can capture finer temporal structures at the cost of loosing frequency resolution, a broader analysis scope can identify correlations on a larger timescale and include more context for the system. Multiscale representations present a possible solution to integrate several spectograms with different resolutions at the same time. Hamel et al. [2012] has introduced this approach which was further adapted by Dieleman and Schrauwen [2013], incorporating Gaussian and Laplacian pyramids. The representations at different time scales can also serve as an input for different CNN architectures, for example to aggregate features of different scales and layers to build more powerful classifiers [Lee and Nam, 2017]. Independent from the computation parameters, matrix representations allow for 2-D convolutions by sliding various filter kernels over the time and frequency domain simultaneously.

2.2.3.3 Network design

Although spectral images are often compared to their visual counterparts, they comprise different structures. Real world pictures show high local correlations regarding colour and intensity in both dimensions, representing objects or parts. In comparison, spectrograms are mostly correlated along a single axis, either due to harmonic content or transient behaviour. While the advantage of scaling and rotation invariance due to convolutional network layers does not apply to spectral audio representations in the same way, translation invariance in both directions is of crucial importance to detect

[2] Recurrent Neural Networks are out of this thesis' scope. Further reading is provided by Goodfellow et al. [2016]

time shifted or pitched structures of its own kind [Choi et al., 2017b]. Even though relatively small rectangular filter shapes have proven to work reasonably well in practice, Pons et al. [2016] recommend employing musically motivated filters with adapted and different receptive fields in the same convolutive layer to improve performance results for automatic music tagging and timbre analysis [Pons et al., 2016] or genre recognition [Pons and Serra, 2017]. By using shallow architectures and a relatively small amount of trainable parameters, those approaches are less prone to overfitting, reduce computational complexity and especially scale well to small datasets.

As long as learning approaches are based on mid-level representations, design principles for convolutional layers still highly depend on the task at hand. Some networks are adapted or even fully designed towards a particular task which again requires domain expertise within the related field. Other hyperparameters to tune the network are the number of filters and the employment of strided convolutions [Choi et al., 2017b].
Besides choosing appropriate filter kernels, deep learning models can further vary in terms of layer types or network depth. The possibilities to design ConvNets for MIR are manifold. Iteratively determining hyperparameters would require an enormous amount of computational power, depending on the dataset size. Recent machine learning research also focuses on building network structures that optimize their design hyperparameters automatically for further improvement [Gu et al., 2017].

2.2.3.4 Transfer learning

The main prerequisite to solve machine learning problems is the availability of large datasets. Unfortunately, data is more than sparse for a lot of MIR tasks. In computer vision, this challenge has been tackled by *transfer learning*. Models like AlexNet [Krizhevsky et al., 2012] which were trained on huge datasets can be re-utilized for similar image recognition tasks where less training material is available in order to bypass the shortage of data. In general, the concept of transfer learning describes the employment of learned features from a source task and applies those features for a different but related target task [Pan and Yang, 2010]. In particular, an already trained network is utilized to first extract features from a new but smaller dataset and afterwards employs those features for the target classification. The adaptation of the original network is often referred to as *fine tuning*. Transfer learning for MIR has not yet gained a lot of attention since large datasets are barely available. The Million Song Database [Bertin-Mahieux et al., 2011] which has been employed by Van Den Oord et al. [2014] and Choi et al. [2017c] for transfer learning towards a variety of audio-related target tasks constitutes the only big enough and publicly available dataset to apply this concept in the MIR domain. Hamel et al. [2013] has explored an alternative technique by embedding features and labels from several datasets into a shared latent space with linear transformations. However, in comparison to computer vision where the employment of large and powerful models for different tasks is already state of the art, transfer learning for MIR still fails to be properly adapted due to the shortage of large datasets and the diversity of audio classification tasks.

3 Multi-track crosstalk reduction using spectral subtraction

While many music-related blind source separation methods focus on mono or stereo material, the detection and reduction of crosstalk in multi-track recordings is less researched. Crosstalk or 'bleed' of one recorded channel in another is a very common phenomenon in specific genres such as jazz and classical, where all instrumentalists are recorded simultaneously. We present an efficient algorithm that estimates the crosstalk amount in the spectral domain and applies spectral subtraction to remove it. Randomly generated artificial mixtures from various anechoic orchestral source material were employed to develop and evaluate the algorithm, which scores an average SIR-Gain result of 15.14 dB on various datasets with different amounts of simulated crosstalk.

3.1 Introduction

In many real-world music performance recordings, bands and ensembles are recorded without perfect acoustic separation. As a result, microphones which were intended to capture a particular instrument also record nearby signals; these additional signals are referred to as crosstalk, spill, or bleed. Although mixing practices allow to integrate crosstalk into the final mix, there are many instances where a better separation of these tracks is desirable for mixing. Other applications such as correctly annotating audio data with activation values, crosstalk suppression in speech audio, or collecting audio data for classification, could benefit from this approach as well.

Established blind source separation methods for music are typically focused on mono or stereo content while employing techniques based on factorization algorithms like NMF [Lerch, 2012] [Müller, 2007], ICA or PCA [Comon and Jutten, 2010], Hidden-Markov-Models [Mysore et al., 2010], or spatial correlation [Ozerov et al., 2012]. More recent approaches also apply deep neural networks to train separation models [Chandna et al., 2017] [Uhlich et al., 2015]. However, if there is single source material from recording scenes in form of multitrack data available, none of the methods mentioned above take advantage of this additional information since they are tailored for mono/stereo content.

The method presented in this paper focuses on cases where multi-track recordings of, e.g., classical ensembles are available. Clifford and Reiss [2011] have investigated a method for crosstalk cancellation for multiple sources by using delay estimation and centered adaptive filters. Both Kokkinis et al. [2012] and Prätzlich et al. [2015] estimate the spectral power density of each voice and then apply a Wiener filter for crosstalk reduction.

In the proposed system, the crosstalk on a particular track is modeled as a weighted sum of the remaining tracks of this recording. The amount of crosstalk between each pair of tracks is estimated by minimizing a cost function based on spectral energy content through gradient descent with momentum [Sutskever et al., 2013]. As an alternative to Wiener filtering, the crosstalk is then removed through spectral subtraction [Boll, 1979].

For evaluation purposes, various datasets of anechoic orchestral multi-track recordings [Pätynen et al., 2008] are used to create artificial mixtures with different amounts of crosstalk (-18 dB, -12 dB and -6 dB). These artificial mixtures will be referred to as mixture tracks. Results are evaluated in two ways: by comparing the mixing matrix to the estimated spectral subtraction weights via correlation and by computing the standard blind source separation performance metrics SDR, SIR, and SAR [Vincent et al., 2006].

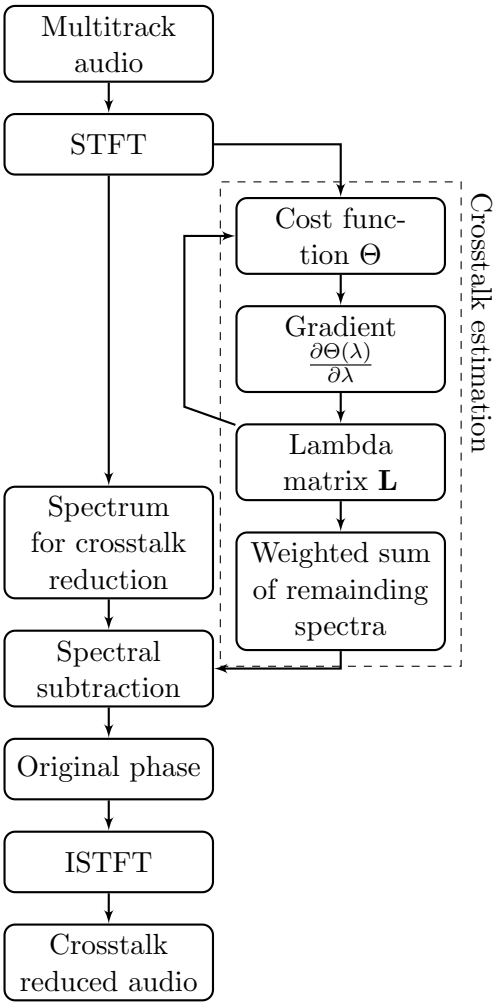


Figure 3.1: Processing steps of the crosstalk reduction algorithm

3.2 Method

The main processing steps of the presented method are displayed in the flowchart in Fig. 4.1. First, a frequency domain representation of the multi-track data is computed by STFT. The crosstalk estimation algorithm models the crosstalk on a particular mixture track as a weighted sum of the other tracks. This estimation process employs an optimization technique based on gradient descent to minimize a spectral energy cost function. After the spectral subtraction, the crosstalk-reduced magnitude spectrogram is recombined with its original phase information to obtain the crosstalk-reduced audio data by inverse STFT. Let $X(j, n, k)$ denote the matrix containing the magnitude spectrogram of the j-th mixture track, calculated with a Hamming window (framesize 4096 samples, hopsize 2048 samples) with k representing the frequency bin index and n the time frame. The analysis window length is approx. 85 ms (at 48 kHz). The rationale behind using comparably long analysis frames is that short time delays have less effect on the crosstalk reduction method and can therefore be neglected. $X_{\mathrm{mix},l}(n, k) = X(j{=}l, n, k)$ represents the spectrum of the l-th track with crosstalk.

3.2.1 Crosstalk estimation

This section outlines the process of estimating the amount of crosstalk from the multi-track data. For each target instrument l, the amount of crosstalk from all other tracks is estimated to derive the weighting factor $\lambda_{l,j}$ via gradient descent on a cost function $\Theta(\lambda)$ that aims to minimize the spectral crosstalk energy in the target mixture spectrum $X_{\mathrm{red},l}(n, k)$, summed over all time frames N as well as all frequency bins K:

$$\Theta(\lambda) = \frac{1}{N} \cdot \sum_{n=1}^{N} \sum_{k=1}^{K} \Big[X_{\mathrm{mix},l}(n, k) - \sum_{j=1,\, j\neq l}^{J} \lambda_{l,j} \cdot X(j, n, k) \Big]^2 \tag{3.1}$$

A correction factor $1/N$ accounts for different track lengths. The gradient for $\lambda_{l,j=i}(m)$ in iteration step m is then given by:

$$\begin{aligned} \frac{\partial \Theta(\lambda_{l,i}(m))}{\partial \lambda_{l,i}(m)} = -\frac{2}{N} \cdot \sum_{n=1}^{N} \sum_{k=1}^{K} \Big[X_{\mathrm{mix},l}(n, k) - \\ \sum_{j=1,\, j\neq l}^{J} \lambda_{l,j}(m) \cdot X(j, n, k) \Big] \cdot X(i, n, k). \end{aligned} \tag{3.2}$$

The update rule with momentum [Sutskever et al., 2013] is defined by:

$$\lambda_{l,i}(m+1) = \lambda_{l,i}(m) - \gamma(m) \cdot v(m+1). \tag{3.3}$$

The adaptation v is computed as:

$$v(m+1) = \beta \cdot v(m) + \frac{\partial \Theta(\lambda_{l,i}(m))}{\partial \lambda_{l,i}(m)} \tag{3.4}$$

with β as the momentum parameter, sometimes called friction, set to 0.8. The learning rate γ slightly decreases in each iteration step m:

$$\gamma(m+1) = 0.99 \cdot \gamma(m) \tag{3.5}$$

with an initial value of $\gamma(0) = 0.001$. Convergence is reached if the stepwise optimization of cost function $\Theta(\lambda(m))$ falls below a certain threshold δ:

$$\Theta(\lambda(m+1)) - \Theta(\lambda(m)) < \delta. \tag{3.6}$$

The gradient descent algorithm aims to find the $\lambda_{l,j}$ that guarantee the lowest overall power within the spectrum $X_{\text{red},l}$ according to the cost function $\Theta(\lambda)$. By utilizing this approach, $\lambda_{l,j}$ adapts to the relative crosstalk amount of the different mixture tracks during the crosstalk estimation. All weighting factors $\lambda_{l,j}$ smaller than zero are automatically set to zero during the gradient descent process.

Table 3.1 shows an entire set of $\lambda_{l,j}$ values for a dataset with nine tracks. The first row displays all estimated weighting factors $\lambda_{\text{bassoon},j}$ that have to be subtracted from the bassoon track to minimize crosstalk.

$\lambda_{l,j}$	bassoon	clarinet	bass	flute	f_horn	sopran	viola	violin	cello
bassoon	0	0.208	0	0.141	0.314	0.046	0.086	0.147	0.058
clarinet	0.182	0	0.07	0.119	0.18	0.065	0	0.105	0.072
bass	0	0.073	0	0	0.298	0.114	0.03	0.18	0.287
flute	0.065	0.062	0	0	0.204	0.017	0	0.112	0
f_horn	0.095	0.066	0.086	0.14	0	0.031	0.141	0	0.059
sopran	0.053	0.085	0.118	0.054	0.118	0	0.102	0.084	0.017
viola	0.059	0	0.022	0	0.4	0.078	0	0.2	0.157
violin	0.102	0.089	0.122	0.179	0	0.054	0.166	0	0.065
cello	0.047	0.062	0.185	0	0.135	0.011	0.128	0.063	0

Table 3.1: Example matrix for estimated $\lambda_{l,j}$ values, computed by the gradient descent algorithm

3.2.2 Crosstalk reduction

After estimating the weight factors for all the bleeding instruments for each track as outlined in Sect. 3.2.1, a sum of their weighted magnitude spectra can be subtracted from $X_{\mathrm{mix},l}(n,k)$ by simple spectral subtraction Boll [1979]. The result with reduced crosstalk $X_{\mathrm{red},l}(n,k)$ is therefore calculated as

$$X_{\mathrm{red,l}}(n,k) = X_{\mathrm{mix,l}}(n,k) - \sum_{j=1,\, j\neq l}^{J} \lambda_{l,j} \cdot X(j,n,k), \tag{3.7}$$

where J represents the total number of tracks. The weighting factor $\lambda_{l,j}$ is estimated from the spectrograms as explained below. If the subtraction results in negative spectrum values in $X_{\mathrm{red},l}(n,k)$, they will be set to zero. Finally, the reduced magnitude spectrum is combined with the original phase information from the STFT analysis to obtain the crosstalk-reduced audio file by inverse Fourier transform.

Figure 3.2 displays example results of the process described for one excerpt from the dataset. The left graphic shows the magnitude spectrogram of a single clean bassoon signal $X_{\mathrm{dry,bassoon}}$, the plot in the middle represents the magnitude spectrogram of the same signal with crosstalk $X_{\mathrm{mix,bassoon}}$, and the result with reduced crosstalk $X_{\mathrm{red,bassoon}}$ is shown on the right. All spectrograms are in logarithmic dB scale.

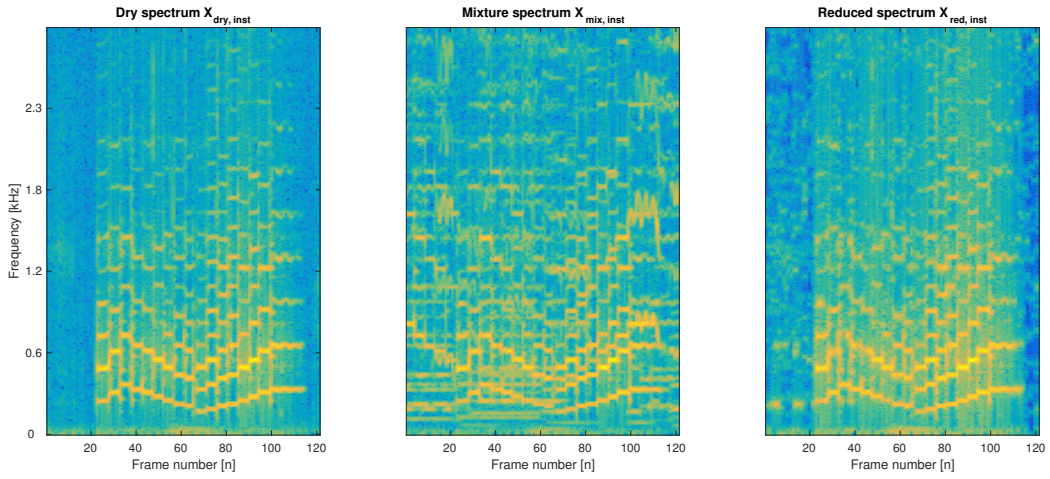


Figure 3.2: Three different magnitude spectra excerpts from the unmixed original bassoon track (left), the artificially generated bassoon mixture with 6 dB crosstalk (middle) and the crosstalk reduced track (right)

3.3 Evaluation

There exist no standardized datasets or evaluation methods for the tasks of crosstalk estimation and reduction for multi-track data. A dataset for these tasks should allow for full control over parameters in the mixing process such as the amount and combination of crosstalk. The multi-track recordings used to create the dataset used in this study are from anechoic symphonic recordings.

Two different metrics are used to evaluate the two main processing blocks of the presented algorithm, respectively. First, the correlation coefficient is computed between the estimated lambda values and a transformed mixing matrix to evaluate the crosstalk estimation. Second, established blind source separation measures (SDR, SIR, and SAR, see below) are used on the audio results to evaluate the overall system.

3.3.1 Dataset

The dataset is created from excerpts from four orchestral anechoic multi-track recordings [Pätynen et al., 2008]:

- Beethoven: Symphony no. 7, I mov. (3:11 min):
  11 Parts: Flutes, Oboes, Clarinets, Bassoon, French horns, Trumpets, Timpani, Violin, Viola, Cello, Contrabass
- Bruckner: Symphony no. 8, II mov. (1:27 min):
  13 Parts: Flutes, Oboes, Clarinets, Bassoon, French horns, Trumpets, Trombones, Tuba, Timpani, Violin, Viola, Cello, Contrabass
- Mahler: Symphony no. 1, IV mov. (2:12 min):
  14 Parts: Flutes, Oboes, Clarinets, Bassoon, French horns, Trumpets, Trombones, Tuba, Timpani, Percussions, Violin, Viola, Cello, Contrabass
- Mozart: An aria of Donna Elvira from the opera Don Giovanni (3:47 min):
  9 Parts: Flute, Clarinet, Bassoon, French horns, Violin, Viola, Cello, Contrabass, Soprano

For each of these four pieces, three different mixture sets are constructed with a randomly generated mixing matrix. This mixing matrix has ones on the diagonal and positive values elsewhere that are limited to a defined maximum crosstalk value for the remaining elements. The three mixture sets have a different maximum crosstalk amount: -6 dB, -12 dB, and -18 dB, which relates to maximum mix factors of 0.5, 0.25, and 0.126, respectively. Table 3.2 shows an example mixing matrix (-18 dB dataset of the Mozart piece). Every row shows the contributions of each input track to one mixture track. The first row, for example, contains all fractions of the solo anechoic instrument tracks that are combined to the artificial bassoon mixture. A symmetric mixing matrix ensures that, for example, the scaling factor of the bassoon instrument on the clarinet mixture track equals the factor of the clarinet instrument on the bassoon mixture track. It is important to note that these mixing values are scaling factors that are independent of

the individual track's acticity and loudness; thus, the actual amount of crosstalk is not necessarily reflected through the mixing matrix value.

Time delays are then calculated according to the reciprocal quadratic relation of distance and amplitude in the free field and accounted for in the mixing process. The reference amplitudes $A = 1$ (diagonal elements in the mixing matrix) correspond to a a distance of 1 m. Finally, all mixture tracks are normalized to the maximum amplitude of the loudest mixture to preserve the mixing matrix relations.

	bassoon	clarinet	bass	flute	f_horn	sopran	viola	violin	cello
bassoon_mix	1	0.097	0.051	0.085	0.088	0.066	0.074	0.097	0.026
clarinet_mix	0.097	1	0.094	0.062	0.073	0.06	0.031	0.073	0.07
bass_mix	0.051	0.094	1	0.024	0.103	0.101	0.084	0.117	0.079
flute_mix	0.085	0.062	0.024	1	0.111	0.029	0.011	0.073	0.004
f_horn_mix	0.088	0.073	0.103	0.111	1	0.063	0.079	0.002	0.077
sopran_mix	0.066	0.06	0.101	0.029	0.063	1	0.083	0.015	0.046
viola_mix	0.074	0.031	0.084	0.011	0.079	0.083	1	0.109	0.006
violin_mix	0.097	0.073	0.117	0.073	0.002	0.015	0.109	1	0.062
cello_mix	0.026	0.07	0.079	0.004	0.077	0.046	0.006	0.062	1

Table 3.2: Randomly generated mixing matrix for the -18 dB Mozart dataset, maximal values of 0.126

3.3.2 Correlation results

The mixing procedure may be represented as a system of linear equations $\mathbf{Ax} = \mathbf{b}$ in which $\mathbf{A}$ is the mixing matrix, $\mathbf{x}$ represents the vector of unmixed instrument tracks and $\mathbf{b}$ describes the mixture vector. Solving this equation for $\mathbf{x}$ leads to the de-mixing matrix $\mathbf{A^{-1}}$. The spectral subtraction Eq. (3.7) can be seen as related to this system of linear equations, where the $\lambda_{l,j}$ matrix (see Table 3.1) represents an estimation of this inverted matrix with flipped signs and empty diagonal. For the sake of concise notation, we will refer to $\lambda_{l,j}$ as $\mathbf{L}$ from now on. Similar to the mixing operation, we thus get a similar system of linear equations $\mathbf{x} \approx (-\mathbf{L} + \mathbf{I})\mathbf{b}$ and it follows that $\mathbf{A^{-1}} \approx -\mathbf{L} + \mathbf{I}$.

The matrices cannot be expected to be identical even in the best case as time delays were introduced when creating the dataset, however, the correlation between $\mathbf{A^{-1}}$ and $(-\mathbf{L} + \mathbf{I})$ should be high if the estimation works. Table 3.3 shows the correlation results for each of the three mixture sets, averaged over all four orchestral pieces. While the -12 dB and -18 dB sets both show very high correlation values of at least 0.9 over all four pieces with barely any variation, the -6 dB sets perform comparably bad for all pieces except the Mozart one. While the Mahler -6 dB set still shows a correlation of about 0.75, the two other sets show only correlation values between 0.6 and 0.437.

	-6dB	-12dB	-18dB
Beethoven	0.437	0.964	0.933
Bruckner	0.595	0.961	0.941
Mahler	0.752	0.931	0.900
Mozart	0.932	0.973	0.976
Average	0.678	0.957	0.937

Table 3.3: Correlation of $\mathbf{A^{-1}}$ and $(-\mathbf{L} + \mathbf{I})$

3.3.3 BSS Eval results

In order to evaluate the crosstalk suppression, the blind source separation (BSS) evaluation measures signal-to-distortion-ratio (SDR), signal-to-interference-ratio (SIR), and signal-to-artifacts-ratio (SAR) [Vincent et al., 2006] were computed and investigated. These measures have become standard metrics for the evaluation of blind source separation systems, for example, in the SiSEC campaign[1]. Since the present approach processes multi-track data as opposed to most BSS methods which work with mono or stereo content, the following results cannot be compared directly to other studies. BSS evaluation metrics are highly dependent on the datasets used for separation (or crosstalk reduction). For this reason, a comparison of BSS Eval measures of the mixtures and the actual crosstalk reduced audio files seems more suitable to get a better insight of the algorithm performance. Table 3.4 shows the BSS Eval measures of mixtures and crosstalk reduced tracks as well as their difference.

	Mixture			Crosstalk-reduced			Difference		
	-6 dB	-12 dB	-18 dB	-6 dB	-12 dB	-18 dB	-6 dB	-12 dB	-18 dB
SDR	0.04	5.79	11.83	7.88	12.95	15.15	7.84	7.16	3.32
SIR	0.04	5.8	11.93	13.3	22.16	27.75	13.26	16.36	15.81
SAR	48.05	39.56	31.53	10.48	13.75	15.49	-37.57	-25.81	-16.05

Table 3.4: BSS evaluation measures for mixtures, crosstalk reduced tracks and their difference, sorted by dataset and averaged over all four orchestral pieces

SDR and SIR measures for the mixture tracks show very similar values, ranging from 0 dB to about 12 dB according to the intervals given by the datasets. The SAR values for those vary from 48 dB down to about 31.5 dB in equal distance. For both the -6 dB and -12 dB mixture sets the crosstalk reduced audio tracks show SDR improvements of about 7–8 dB while the -18 dB sets only gain about 3 dB. The difference in terms of SIR measures is very similar, all sets show improvements in the range of 13–16 dB. Changes

[1] http://sisec.inria.fr, last accessed March 6, 2018

regarding the SAR values depend more on the mixture set. While the SAR measure drops heavily for the -6 dB set, the -12 dB set shows a negative gain of about 26 dB and the -18 dB SAR value only decreases by 16 dB.

To investigate the variation of the BSS Eval measures for different music pieces, Table 3.5 displays the SDR/SIR difference of mixtures and crosstalk-reduced tracks as well as the absolute SAR scores for the crosstalk reduced results. SDR values increase nearly uniformly over all four orchestral pieces. While the increase for the -6 dB and -12 dB mixture sets ranges from 6 dB to 9 dB, the difference regarding the -18 dB mixture set only results in about 2–4 dB. SIR values are even more evenly distributed: most results lie in the interval of about 14–16 dB except the Bruckner/Beethoven values for the -6 dB dataset (10–11 dB) and all Mozart scores which are about 4 dB higher.

SAR values, in turn, quantify the musical noise in the crosstalk-reduced audio tracks. Similar to the SDR/SIR gain results, the metrics especially vary for the -6 dB mixture set, where the values range from 9 dB to about 12.5 dB. Results constantly increase for the mixture sets with lower crosstalk, although the improvement from the -6 dB to -12 dB sets is more distinct than from -12 dB to -18 dB. Again, the Mozart pieces have the best results being about 3 dB higher than the other pieces.

	SDR gain			SIR gain			SAR absolute (results)		
	-6 dB	-12 dB	-18 dB	-6 dB	-12 dB	-18 dB	-6 dB	-12 dB	-18 dB
Beeth.	7.53	7.33	3.5	11.21	15.83	15.52	10.78	14.09	15.54
Bruck.	6.67	6.66	2.41	10.12	14.62	14.29	8.99	12.69	14.12
Mahler	8.04	7.33	3.46	14.66	15.39	14.99	9.71	12.24	14.54
Mozart	9.14	7.31	3.92	17.03	19.59	18.46	12.43	15.97	17.74

Table 3.5: SDR/SIR gain for each mixture set as well as absolute SAR values of the crosstalk reduced results

3.4 Discussion

The results of evaluation metrics show generally consistent trends. Very high correlation values of te de-mixing matrix $\mathbf{A^{-1}}$ and $(-\mathbf{L} + \mathbf{I})$ for both the -12 dB and -18 dB dataset for all pieces validate the success of the presented approach and prove that the gradient descent algorithm finds suitable $\lambda_{l,j}$ values minimizing the cost function. Multiple runs with random initialization further indicate that the detected cost minima are actually global minima.

There exist multiple reasons explaining the variation of the results between the different pieces. First, the total amount of crosstalk is not equal between the pieces. This is shown by the SIR measure of the mixtures given in Table 3.6 as these values represent the actual amount of crosstalk. While the increase over mixtures is consistently

6 dB as expected, the variation between pieces amounts to up to 4 dB. There are two reasons for that. First, the way the mixing matrix is generated means that the resulting amount of crosstalk depends on the number of instrument tracks. Second, the mixing matrix only signifies the factor but the resulting amount of crosstalk also depends on the track content and distribution; for example, percussion or timpani mixture tracks often show very low SIR and SAR since events only occur rarely during the track. Therefore, the optimal solution for the minimization of the cost function using the spectral energy criterion produces relatively high $\lambda_{l,j}$ values which in turn result in a harsh spectral subtraction and more musical artifacts. Those artifacts are quantified in the SAR score (see Table 3.4).

Crosstalk reduction for instruments that have overlapping frequency ranges with a similar tonal character is harder than for instruments with a unique spectral signature. Sections where the whole ensemble plays simultaneously are more difficult to manage for the algorithm than solo parts of individual instruments. In all cases, the Mozart piece achieves the best performance scores. The SAR values of the three remaining pieces range about 3 dB in comparison the the Mozart piece, each one following the above mentioned relation so that the absolute SIR values increase in 6 dB steps towards the -18 dB datasets (see Table 3.6).

	-6dB	-12dB	-18dB
Beethoven	0.03	5.92	11.73
Bruckner	-1.35	5.1	11.46
Mahler	-0.91	3.81	10.81
Mozart	2.39	8.39	13.73

Table 3.6: Absolute SIR values of the mixtures

In general, the crosstalk reduction method generates promising results. Whether the algorithm is suitable for a specific use case highly depends on the application. For tasks such as annotating audio data with instrument activations, the amount of separation (compare SIR) is crucial while the actual audio quality is irrelevant. The amount of subtraction can be controlled by scaling the $\lambda_{l,j}$ values, which can be an advantage in this scenario. In other cases, such as mixing software or more consumer-oriented products, the amount of musical artifacts can be decreased by utilizing an improved separation approach. Possible such improvements could include filtering approaches [Ephraim and Malah, 1984, Lukin and Todd, 2007] or other musical noise suppression techniques [Goh et al., 1998, Esch and Vary, 2009] to reduce artifacts. Thresholding in the spectral or temporal domain could constitute another post-processing feasibility.

The generated dataset contains numerous different instruments which makes the task more challenging. To explore the applicability for a broader field of possible applications, the algorithm needs to be tested on different musical genres, for example with the MedleyDB [Bittner et al., 2014] or Mixing Secrets [Gururani and Lerch, 2017] dataset.

3.5 Summary

The present study has introduced a new method for crosstalk reduction applied to multitrack data such as multi-microphone ensemble recordings. In a first step, this approach estimates the amount of crosstalk from a particular mixture track with a weighted sum of the remaining tracks by iteratively minimizing a spectral cost function with gradient descent. Second, this weighted sum of remaining instruments is subtracted from the mixture track to perform the reduction. Combining the resulting magnitude spectrum with its original phase information allows to obtain the crosstalk-reduced audio data via inverse STFT. In order to evaluate the algorithm, various mixtures with different amounts of crosstalk were artificially generated from multiple anechoic orchestral recordings. Results were evaluated in two ways: first, by correlating the mixing matrices and the resulting lambda matrices containing the estimated crosstalk factors to investigate the crosstalk estimation itself, and second by employing the standard blind source separation evaluation metrics SDR, SIR, and SAR to evaluate the suppression. Both evaluation metrics showed promising results. Post-processing techniques such as filtering or noise suppression could further improve the algorithm by reducing artifacts. For possible applications in a wider musical scope, tests with more genres are recommended.

4 Investigating the influence of frame sizes in instrument identification systems based on convolutional neural networks

Identifying instruments in a given musical piece can help to improve recommender systems, implement new functions for music libraries or inform related tasks like source separation or score notation. The herein presented instrument identification systems are based upon a convolutional neural network system and trained on classical music multi-track data from the Berlin University of Arts. Labels and input features in form of mel-spectrograms are extracted in different time frames sizes to investigate the influence of frame size to classification performance. Besides a standard learning method, two different approaches to classify small time frames while learning on larger frames are investigated; sliding predictions and multiple instance learning. Results show a certain performance peak for frame sizes of about 2.7 seconds regarding this dataset and the applied learning settings. Using larger frame sizes to iteratively classify smaller frames only improves upon standard techniques in a limited frame size region. The employed multiple instance learning techniques cannot enhance classification performance.

4.1 Introduction

One of the characteristics of a musical piece is it's instrumentation. Very often, the choice of instruments itself provides insights into the music era, genre or even particular artist of a given song. Therefore, information about a song's instrumentation could improve Music Information Retrieval (MIR) tasks such as automatic tagging, user search requests in online music libraries or music recommendation in streaming services. Furthermore, knowledge about instruments and their properties can potentially improve other tasks such as source separation or automatic transcription by allowing these systems to adapt their parameters. Especially instrument recognition (IR) algorithms for polyphonic mixtures could be beneficial for such tasks since most popular or classical music recordings consist of those mixtures.

While humans, particularly musically trained experts, are able to analytically deconstruct music pieces into harmonic or rhythmic structures as well as their instrumentation, data-driven systems still struggle to solve this task sufficiently well. There exist multiple reasons for that. First, the instruments have to be identified in a complex polyphonic mixture of non-trivial sounds. Second, the number of different instruments throughout genres is large. Third, each musical instrument might offer a wide variability in terms of timbre, pitch, tuning, and playing techniques.

The present study is concerned with identifying instruments in polyphonic mixtures of classical music, both chamber and orchestra pieces within a corpus of 13 pre-defined classical instruments. One of the main challenges in such MIR and machine learning contexts is collecting a sufficiently large and diverse dataset to work with. Unfortunately, for classical polyphonic music, there do not exist any labeled datasets. Therefore, our dataset comprises a collection of multi-track recordings from concerts at Berlin University of Arts. Features and labels are extracted from these multi-tracks in order to train a classifier based on a convolutional neural network architecture. The extraction and learning process is carried out with various lengths of audio input snippets to investigate in which way this input length influences the classifier performance. Furthermore, two methods are presented and evaluated which allow for model predictions on short time scales, even though the labelling information and hence the learning process was carried out on a larger time scale. In this way, relatively inaccurate labels can still be utilized to give precise predictions.

4.2 Related work

Existing approaches to instrument identification differ in terms of methodology and source material. Generally, the identification task becomes harder the more instruments play simultaneously and the more those instruments are similar in their time-frequency patterns. Most studies either use single instrument notes or phrases that were recorded individually or polyphonic mixtures of all instruments playing together in an ensemble as in this work.

4.2.1 Monophonic instrument recognition

The analysis of individually recorded notes has been tackled with a variety of approaches on different datasets and instruments. As one of the first studies in this field, Eronen and Klapuri [2000] used Gaussian- and kNN-classifiers on a set of 43 engineered temporal and spectral features from 30 different orchestral instruments. This work is often referred to as baseline system and marks a first milestone in the research field of instrument recognition. Diment et al. [2013] investigated modified group delay features including phase information plus mel-frequency cepstral coefficients (MFCCs) on different sets up to 22 instruments from the RWC database [Goto et al., 2002]. Yu et al. [2014] employed sparse coding on cepstrum with temporal sum-pooling for classifying 10 instruments from the ParisTech dataset [Joder et al., 2009]. Park and Lee [2015] extract spectrogram images in combination with multi-resolution recurrence plots including phase information from different notes of 20 different instrument, taken from the UIOWA MIS database. Those are fed into a convolutional neural network to build a classifier. Han et al. [2016] propose a sparse coding approach to learn features from mel-spectrograms, extracted from single notes of 24 instruments from the RWC dataset [Goto et al., 2002], to train a support vector machine (SVM).

While formerly mentioned approaches focus on isolated notes, various studies have

also applied instrument recognition on monophonic solo phrases. Krishna and Sreenivas [2004] utilized line spectral features combined with gaussian mixture models (GMM) to classify 14 individual instruments. Essid et al. [2004] also employed GMM on the task of classifying solo performances of five different instruments, but paired the approach with MFCCs and principal component analysis (PCA).

4.2.2 Polyphonic instrument recognition

High performance results for instrument recognition regarding monophonic source material have shifted the research focus more towards polyphonic mixtures. Heittola et al. [2009] focussed on synthesized mixtures containing randomly generated notes from 14 instruments, classifying those with a non-negative matrix factorization based source filter model with MFCCs and GMM. Kitahara et al. [2007] employed several spectral, temporal and modulation features combined with PCA and linear discriminant analysis (LDA) as classification methods. Duan et al. [2014] investigated mel-scale uniform discrete cepstrums as novel spectral representation with a radial basis function SVM. The approach was evaluated on randomly mixed chords from note samples of 13 different instruments, taken from the RWC database [Goto et al., 2002].

For many applications, a sufficient outcome from IR algorithms is determining a predominant instrument. Fuhrmann et al. [2009] and Bosch et al. [2012] use a large set of feature, paired with source separation methods as pre-processing step to identify such predominant instruments with individual SVM models for each instrument class.

In the domain of IR, machine learning efforts, especially convolutional neural networks (CNN) which include feature learning, have become more and more popular due to their success in the whole MIR domain [Choi et al., 2017b]. Regarding the task at hand several studies have already exploited this learning techniques for instrument recognition. Park and Lee [2015] introduce a feature approach with spectrogram plus so-called multi-resolution recurrence plots which incorporate the audio signal's phase trajectory distances in matrix form to feed a multi-column CNN. Han et al. [2017] apply multi-convolutional-layer CNNs with single-label data to determine predominant instruments, hereby including a detailed evaluation of network architecture, analysis time windows and prediction configuration.

4.3 Methods

Similar to the above mentioned techniques, the herein proposed approach is also based on polyphonic source material. However the classification is not determining a predominant instrument but all instruments simultaneously in a multi-class and multi-label learning setting, which means multiple instruments can be classified at the same time. To summarize the methodology of this work, figure 4.1 shows a flowchart of the process. Each individual step is described in the following.

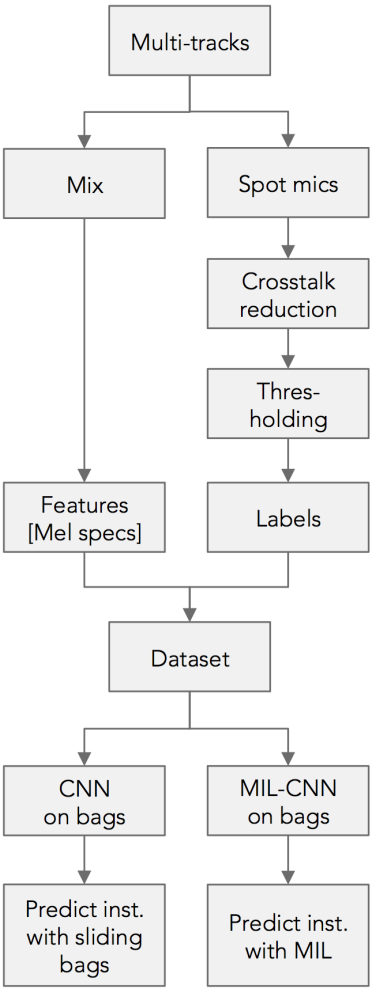


Figure 4.1: Flowchart of processing steps

4.3.1 Pre-processing

All multi-track audio data was sampled with 44.1 kHz, converted to mono and normalized by its maximum value per track. Each track is divided into *frames* of $n_S = 1024$ samples (22.23 ms). A number of n_F frames are combined to blocks, hereafter called *instances*. In turn, n_I Instances are concatenated to a *bag*. This follows the Multiple Instance Learning terminology used by Herrera et al. [2016].

4.3.2 Labels

The multi-track files are split into two groups. While the stereo mixtures are used for extracting features, the remaining tracks, which are recordings from the individual

instruments such as spot microphone recordings, are processed to obtain the labelling information.

In the case of our dataset, those spot microphones contain a certain amount of crosstalk depending on the spatial layout of the ensemble as well as the concert hall. In most cases, the amount of crosstalk does not allow to determine the instrument activity from the original recordings. Therefore, a crosstalk reduction method for multi-track data was applied to the data [Seipel and Lerch, 2018]. After reducing the amount of crosstalk, a frame-wise calculation of the root mean square value yields an instrument activity (IA) value per frame. The instance IA in turn is the maximum of all frame-wise IA in that particular instance, a form of max-aggregation. The instance IA is then thresholded to produce a binary label per instance. The label was carried out on different instance sizes of a multiple of $n_F = 16$ ranging from 16 to 320 frames.

4.3.3 Input representation

Every multi-track stereo mixture is transformed to a spectral representation using the STFT and further processed by a mel filterbank [O'shaughnessy, 1987] to obtain a $n_M = 128$ band mel-spectrogram in the frequency range of 0–10000 Hz. The mel-scale representation has been proven superior to other spectral representations in many Music Information Retrieval tasks [Choi et al., 2017b]. The mel-spectrograms for each stereo mixture track are divided into instance-length pieces and serve as input audio features to the convolutional neural net. Similar to the label creation, the feature extraction was also carried out on different instance sizes of a multiple of $n_F = 16$ ranging from 16 to 320 frames.

4.3.4 Network architecture

The advantage of Convolutional Neural Nets (CNN) for audio classification is based on their ability to map feature inputs to a more compact representation by using layer-wise non-linear transformations. Convolutional layers introduce constraint connectivity patterns which are especially useful to detect locally correlated features, for example edges and lines in a spectrogram which represent single tones. Pooling layers ensure that a certain patterns like motifs or chords can be detected independent from its location in the spectrogram.

The employed architecture is adopted from [Han et al., 2017] which in turn was inspired by AlexNet [Krizhevsky et al., 2012] and VGGNet [Simonyan and Zisserman, 2014]. Table 4.1 displays an overview of the network architecture for a mel-spectrogram input with 16 frames. Compared to the approach of Han et al. [2017], every input to the individual convolutional layers is zero padded with a size of 2×2 instead of 1×1. Otherwise the network structure is identical. It consists of four convolution blocks with an increasing number of filters (32, 64, 128, 256). Each block contains two convolutional layers with ReLu activation function plus above mentioned zero-padding beforehand, 3x3 max-pooling and 25% Dropout. At the end of the last convolution block, global max-pooling is applied before using a fully connected ReLu layer combined with 50% Dropout.

Finally, another fully-connected layer with sigmoid activation maps the network output to the labeling information.

Layer	Output size	Parameters
Mel-spectrogram	128 x 16 x 1	0
2x2 Zero padding	132 x 20 x 1	0
3x3 Conv, 32 filters	130 x 18 x 32	320
2x2 Zero padding	134 x 22 x 32	0
3x3 Conv, 32 filters	132 x 20 x 32	9248
3x3 Max pooling	44 x 6 x 32	0
Dropout(0.25)	44 x 6 x 32	0
2x2 Zero padding	48 x 10 x 32	0
3x3 Conv, 64 filters	46 x 8 x 64	18496
2x2 Zero padding	50 x 12 x 64	0
3x3 Conv, 64 filters	48 x 10 x 64	36928
3x3 Max pooling	16 x 3 x 64	0
Dropout(0.25)	16 x 6 x 64	0
2x2 Zero padding	20 x 7 x 64	0
3x3 Conv, 128 filters	18 x 5 x 128	73856
2x2 Zero padding	22 x 9 x 128	0
3x3 Conv, 128 filters	20 x 7 x 128	147584
3x3 Max pooling	6 x 2 x 128	0
Dropout(0.25)	6 x 2 x 128	0
2x2 Zero padding	10 x 6 x 128	0
3x3 Conv, 256 filters	8 x 4 x 256	295168
2x2 Zero padding	12 x 8 x 256	0
3x3 Conv, 256 filters	10 x 6 x 256	590080
Global max pooling	256	0
Fully connected	1024	263168
Dropout(0.5)	1024	0
Sigmoid	13	13325
Total		1.448.173

Table 4.1: Summary of network architecture for 128x16 input mel-spectrogram (1 instance)

4.3.5 Experimental design

This work is split into three experiments which include different training and test settings. In all cases the learning procedure was carried out on a NVIDIA GTX1060 (6GB memory) by optimizing the binary cross-entropy loss with the ADAM [Kingma and Ba, 2014] technique (learning rate = 0.001, no learning rate decay) and using a minibatch size of 64 samples over 10 epochs. As mentioned in Sect. 4.3.4, dropout [Srivastava et al., 2014] was applied to each convolutional block as well as previous to the final sigmoid layer to regularize the learning procedure and prevent overfitting.

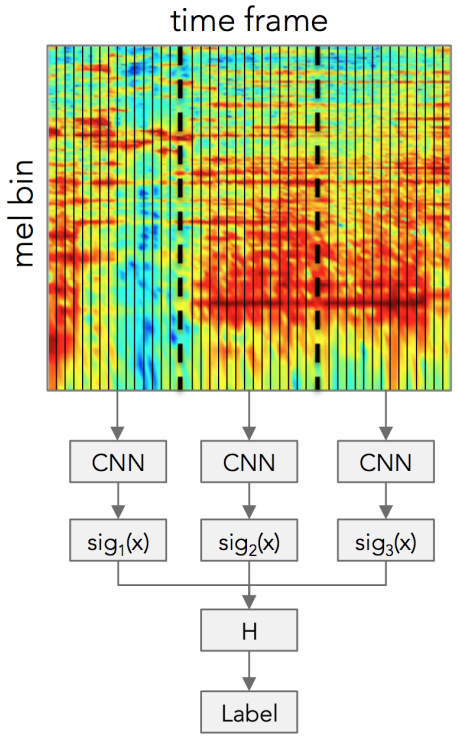


Figure 4.2: Overview of MIL learning procedure of experiment 2

4.3.5.1 Experiment 1 - Standard learning

In the first experiment, training was performed in a common training setting with $n_I = 1$, so each bag only contains only one instance with a number of $k \cdot 16 \cdot n_F$ spectrogram frames ranging from 16 to 320. The CNN architecture shown in Sect. 4.3.4 was employed. For testing, the model which was trained on a specific number of n_F frames was then also utilized to predict sections of the same number of n_F frames.

4.3.5.2 Experiment 2 - Sliding predictions

For the second experiment, the training phase equals the proceeding of experiment 1. However, at test time, models which were trained on larger number of spectrogram frames $n_{F,T}$ are utilized to predict spectrogram excerpts with a smaller number of frames $n_{F,P}$ by sliding the trained model over the test audio iteratively. For example, a model that was trained on $n_{F,T} = 48$ training frames is given the spectrogram patch of the first $n_{F,T} = 48$ frames of an audio signal, the prediction however is only valid for the $n_{F,P} = 16$ prediction frames in the center of the patch. Thereafter, the model is shifted $n_{F,P} = 16$ frames further to predict the following $n_{F,P} = 16$ prediction frames. With this method, models which were trained on larger spectrum inputs can still be used to predict on a finer time scale. For correct prediction, the audio signal need to be zero padded in the beginning and at the end with $(n_{F,T} - n_{F,P})/2$ frames.

4.3.5.3 Experiment 3 - Multiple Instance Learning

Similar to the previous approach, this experiment also targets the case that training labels are only available for larger time periods, but predictions need to be given on a finer time scale. For this reason, training was carried out using a multiple instance learning (MIL) setup with a wrapper approach [Herrera et al., 2016, pp. 68]. Figure 4.2 shows such a setup for an instance size of $n_F = 16$ frames and a bag size of $n_I = 3$ instances. Each of the three instances which consist of a $n_F = 16$ frames spectrogram is fed into an individual CNN presented in Sect. 4.3.4. The sigmoid outputs for each CNN are then combined with an averaging aggregation function H to 13 class activation that are connected to the bag label. The backpropagation runs through the aggregation function to the individual CNNs in order to optimize the weights regarding the bag labels when, in this case, three instance spectrograms are presented to the networks. With such a MIL setup, it is possible to learn on a bag level with its corresponding bag labels but predict at the instance-level. This procedure can be carried out with a larger number of frames per instance n_F as well as a larger number of instances per bag n_I. If $n_I = 1$, this setup corresponds to a standard learning procedure.

At test time, the last model layer, containing the aggregation function, is removed, so the model shows the individual CNN sigmoid activations for each instance. In the case above, when presenting the model with an input spectrogram of $n_F \cdot n_I = 48$ frames, it is able to obtain predictions for all $n_I = 3$ instances with $n_F = 16$ frames individually.

4.4 Evaluation

4.4.1 Dataset

The multi-track dataset comprises a variety of concerts recordings from the Berlin University of Arts. In total, the dataset contains 116 classical music pieces, either symphony or chamber music and includes various concert venues, composers, artists, and different instrumentations. As Table 4.2 displays, the total length of all pieces is 19.16 hours with

a corpus of 13 instruments. However, the dataset is unbalanced in terms of instrument occurrence throughout the pieces. While string instruments are nearly always present, both in chamber music and symphony pieces, other instruments such as the brass section can only be found in approximately a fifth of all pieces. For each piece, there exist a spot microphone track for every instrument as well as a stereo downmix. As usual in acoustically recorded music, the individual spot microphone tracks contain crosstalk. In order to avoid too optimistic results, the test/train split was carried out piece-wise to ensure that no parts of the same pieces are in both training and test set. The resulting training set includes 92 pieces, while the testing set comprises 23 tracks. Figure 4.3 shows the distribution of track lengths for all pieces.

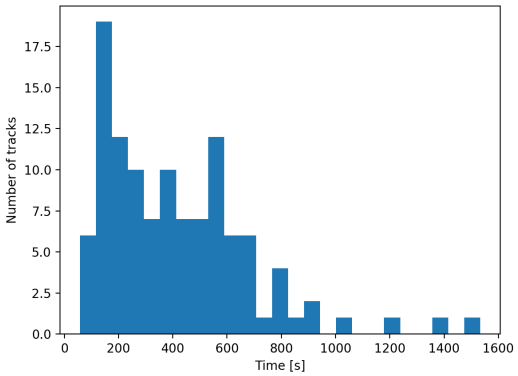


Figure 4.3: Overview of track lengths in seconds in both test and training set together

4.4.2 Performance metrics

In the context of polyphonic instrument identification, different classes may occur in the same sample, or this case in the same instance, defining this task as a multi-class as well as multi-label problem. All metrics are calculated per track and averaged afterwards (macro). This macro approach normalizes the effect on different piece lengths, but at the same time, weights instances in shorter tracks.

4.4.2.1 Threshold dependent metrics

Since the dataset is imbalanced in terms of class/instrument distribution, the evaluation demands for metrics which are not influenced by such different class sizes. For this reason, the established evaluation metrics *precision*, *recall* and *f1-score* are calculated for each instrument class separately and then averaged over the individual classes. The sigmoid activation of each CNN produces a probability per class. A fixed threshold $\sigma = 0.5$, which has been found iteratively on maximizing the scores on the training

Instrument	Size[MB]	Hours	Tracks
Bassoon	2033.061	6.40	51
Clarinet	1997.021	6.28	48
Double Bass	2135.858	6.73	54
Flute	1342.118	4.23	39
French Horn	2033.061	6.40	51
Oboe	1523.016	4.80	39
Piano	3851.172	12.13	72
Trombone	834.472	2.63	28
Trumpet	834.472	2.63	28
Tuba	783.636	2.47	24
Viola	3611.268	11.37	75
Violin	4937.731	15.55	87
Violoncello	5399.067	17.00	104
Total/Mix	6082.926	19.16	116

Table 4.2: Overview of the dataset, separated by instrument

set, then defines how those activations are binarized and hence influences the prediction scores on the above mentioned metrics.

4.4.2.2 Threshold independent metrics

Another metric, which is independent from the given binarizing threshold, is the so called AUC. The *AUC* represents the *Area Under The Receiver Operating Characteristic Curve* which in turn evaluates the true positive against the false positive rate for various thresholds [Virtanen et al., 2018].

4.5 Results and Discussion

4.5.1 Experiment 1 - Standard learning

In the first experimental case, which is hereby referred to as standard learning, the classification results for varying frame sizes in between $n_F = 16$ (0.37 s) and $n_F = 160$ (3.70 s) were evaluated. The results, displayed in Figure 4.4, show to what extent the classification scores are dependent on the frame size of labels and features which were used for training the convolutional neural network. In general, the overall classifier performance throughout all measures improve until $n_F = 112$ (2.60 s) before they slightly decrease to higher frame size values. For this training setup, the network seems to need an appropriate amount of frames to learn characteristics of the different instruments which include both the attack, sustain and decay phase of tones or chords. In case of smaller receptive fields of only $n_F = 16$ or $n_F = 32$ frames the time window can be too small to capture the above mentioned temporal structure of tones produced by classical musical instruments. Furthermore, the difference in between recall and precision measure for smaller frames indicates a non-optimal binarizing threshold since the AUC score stays consistently at the same level. While this threshold was found by optimizing the scores for all frame sizes, an adaptive threshold for different frame sizes can improve the results here.

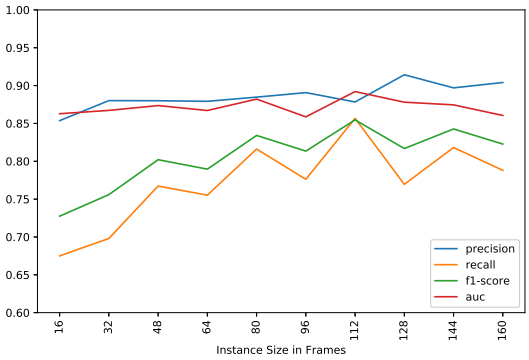


Figure 4.4: Evaluation results for the standard learning approach

4.5.2 Experiment 2 - Sliding predictions

The second approach describes a prediction method based on iteratively sliding the classifier over the audio material's mel-spectrorams. Figure 4.5 shows the prediction frame size $n_{F,P}$ on the x-axis while the different subplots denote the ratio of training to prediction frame size $r = n_{F,T}/n_{F,P}$. The results for $r = 1$ equal the performance scores of the standard learning approach shown above. By increasing the ratio to $r = 3$, the larger receptive field of the classifier can improve the performance, especially for

smaller prediction frames $n_{F,P}$. However, with higher ratios of $r = 5$ and $r = 7$, the scores decrease consistently with rising prediction frame size which also relates to large training frame sizes then. This indicates that above a certain threshold, the training frame size seems to be too large to train the network efficiently. Additionally, the usage of such larger training frame sizes also decreases the overall samples for the training procedure which again can impair the learning procedure.

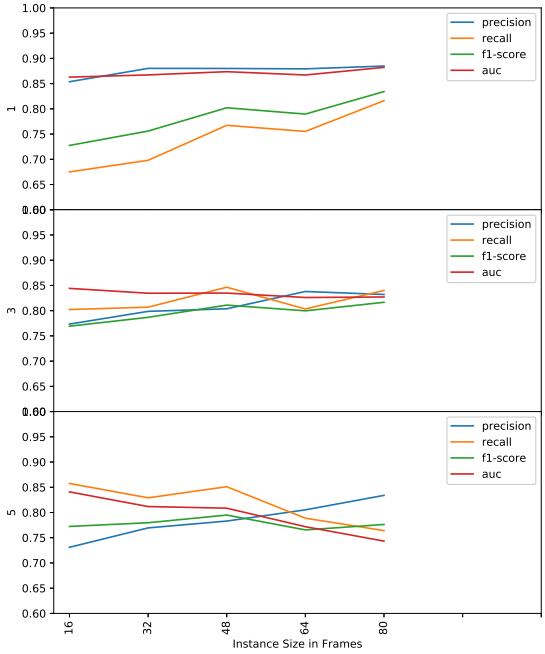


Figure 4.5: Evaluation results for sliding predictions

4.5.3 Experiment 3 - Multiple Instance Learning

Similar to the previous method, multiple instance learning with so called wrapper methods also targets to predict on small time frames while learning on larger input features and labels. The results for this experiment are shown in Figure 4.6, following the same structure than the previous approach. While the increase of the receptive field in the last experiment showed improvement, the MIL approach fails for both cases, using $n_I = 3$ as well as $n_I = 5$ instances. Possible reasons can include a non-optimal aggregation function or the inefficient training of parallel network architectures.

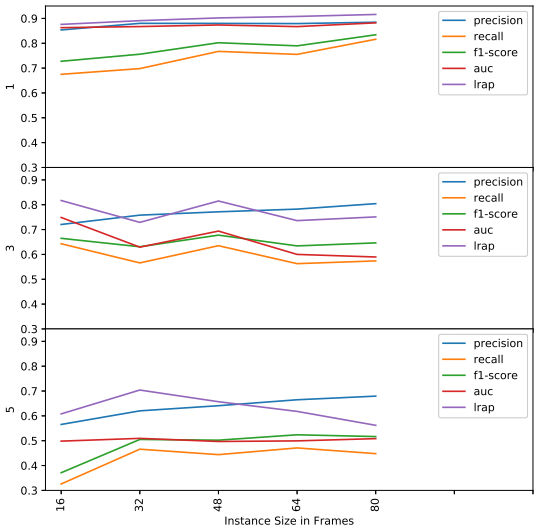


Figure 4.6: Evaluation results for the multiple instance learning approach

4.6 Summary

This work has investigated an instrument identification approach based on convolutional neural networks for classical music audio data. The network was trained on different audio frame sizes in order to investigate the influence of frame size to performance results. The overall processing steps include a crosstalk reduction pre-processing step on spot microphone recordings to correctly determine the individual instrument activities and subsequently extract labels. Features were generated in form of mel-spectrograms from the respective stereo mixture. The trained models for a standard learning approach show increasing performance until frame sizes of $n_F = 112$ are exceeded. In case of the sliding predictions approach, the usage of classifiers which were trained on larger frame size $n_{F,T}$ can enhance performance for small prediction frame sizes $n_{F,P}$ but cannot improve upon the standard approach for higher values of $n_{F,T}$ and $n_{F,T}$. The MIL method for this setup has proven unsuccessful for all learning parameters. Future work must therefore include an evaluation with various aggregation functions, a different dataset containing normalized track lengths and different genres for learning and test phase as well as an evaluation on both micro and macro level, also including more suitable measures for multi-label classification.

5 Conclusion

This master thesis has presented and investigated several convolutional neural network approaches to instrument recognition on polyphonic audio material and further introduced a crosstalk reduction method for multi-track audio.

The crosstalk reduction method was developed to clear multi-tracks from spill, a common phenomenon when recording acoustic ensembles in concert halls or studios. Besides its original purpose, the preparation of audio files to correctly extract labelling information, the algorithm can also serve as a de-noising method for multi-channel microphone input or as post production effect to clean multi-track audio recordings. The algorithm shows solid evaluation scores, in particular the improvement of the signal-to-interference ratio within the given dataset of artificially created mixtures from anechoic recordings. However, the algorithm also introduces musical artefacts as a result of the spectral subtraction. By scaling the reduction parameters, the algorithm can be adapted to its application purpose and in this way either improve the reduction at the cost of more musical artefacts or vice versa. A further investigation of the approach could involve different datasets with more musical genres and convolutive mixtures. Advanced post-processing like filtering in the spectral domain or thresholding in the time domain to reduce musical artefacts could further improve the algorithm.

In order to investigate instrument identification with convolutional neural networks three different experiments were designed. For all approaches, a multi-track classical music dataset incorporating 13 different instruments was employed. After applying the herein developed crosstalk reduction method on the spot microphone tracks, the individual instrument activities were first determined to then binarize those to labelling information. Features were extracted in form of mel-spectrograms from the according stereo mixtures. To examine the effects of frame size to classification performance, labels as well as features were generated with different frame sizes. After the training phase, the learned model is able to predict the instrument composition for the given frame size. For the present dataset and the employed learning settings, the best classifications scores were obtained for a frame size of about 2,7 seconds. Two different models based on sliding predictions and multiple instance learning were further investigated. While the former can improve classification for small frame sizes by utilizing classifiers which were trained on larger frames but only classify center frames, the latter does not show any enhancements. Future work includes the application of the above presented methods on different datasets and an investigation of the MIL approach, for example by using other aggregation functions. To understand the behaviour of the convolutional neural net for Instrument Identification a visualization of activations and weights might help to draw further conclusions.

List of Acronyms

AUC	Area Under Curve
BSS	Blind Source Separation
CNN	Convolutional Neural Network
CQT	Constant-Q Transform
FN	False Negatives
FP	False Positives
GD	Gradient Descent
GMM	Gaussian Mixture Model
ICA	Independent Component Analysis
IA	Instrument Activity
IR	Instrument Recognition
kNN	k-Nearest Neighbour
LDA	Linear Discriminant Analysis
MFCC	Mel Frequency Cepstral Coefficient
MIC	Multi Instance Classification
MIDI	Musical Instrument Digital Interface
MIL	Multiple Instance Learning
MIR	Music Information Retrieval
ML	Machine Learning
MSE	Mean Squared Error
NAG	Nesterov Accelerated Gradient
NMF	Non-negative Matrix Facorization
PCA	Principal Component Analysis
ROC	Receiver Operating Characteristics
SAR	Signal-to-Artefacts-Ratio
SDR	Signal-to-Distortion-Ratio
SIR	Signal-to-Interference-Ratio
STFT	Short Time Fourier Transform
SVM	Support Vector Machine
TN	True Negatives
TP	True Positives
TTS	Text to Speech

Bibliography

Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. The Million Song Dataset. In *Ismir*, volume 2, page 10, 2011.

C Bishop. Pattern Recognition and Machine Learning (Information Science and Statistics), 1st edn. 2006. corr. 2nd printing edn. *Springer, New York*, 2007.

Rachel M Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Pablo Bello. MedleyDB: A Multitrack Dataset for Annotation-Intensive MIR Research. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, volume 14, pages 155–160, 2014.

S. Boll. Suppression of acoustic noise in speech using spectral subtraction. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 27(2):113–120, April 1979.

Juan J Bosch, Jordi Janer, Ferdinand Fuhrmann, and Perfecto Herrera. A comparison of sound segregation techniques for predominant instrument recognition in musical audio signals. In *ISMIR*, pages 559–564, 2012.

Judith C Brown. Calculation of a constant Q spectral transform. *The Journal of the Acoustical Society of America*, 89(1):425–434, 1991.

Pritish Chandna, Marius Miron, Jordi Janer, and Emilia Gómez. Monoaural audio source separation using deep convolutional neural networks. In *International Conference on Latent Variable Analysis and Signal Separation*, pages 258–266. Springer, 2017.

Keunwoo Choi, George Fazekas, and Mark Sandler. Explaining deep convolutional neural networks on music classification. *arXiv preprint arXiv:1607.02444*, 2016.

Keunwoo Choi, George Fazekas, Kyunghyun Cho, and Mark Sandler. A comparison on audio signal preprocessing methods for deep neural networks on music tagging. *arXiv preprint arXiv:1709.01922*, 2017a.

Keunwoo Choi, György Fazekas, Kyunghyun Cho, and Mark Sandler. A tutorial on deep learning for music information retrieval. *arXiv preprint arXiv:1709.04396*, 2017b.

Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. Transfer learning for music classification and regression tasks. *arXiv preprint arXiv:1703.09179*, 2017c.

Alice Clifford and Joshua D. Reiss. Microphone interference reduction in live sound. In *Proceedings of the 14th International Conference on Digital Audio Effects (DAFx-11)*, 2011.

Pierre Comon and Christian Jutten. *Handbook of Blind Source Separation: Independent component analysis and applications*. Academic press, 2010.

Xiaodong Cui, Vaibhava Goel, and Brian Kingsbury. Data augmentation for deep neural network acoustic modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(9):1469–1477, 2015.

Wei Dai, Chia Dai, Shuhui Qu, Juncheng Li, and Samarjit Das. Very deep convolutional neural networks for raw waveforms. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 421–425. IEEE, 2017.

Steven B Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. In *Readings in speech recognition*, pages 65–74. Elsevier, 1990.

Sander Dieleman. *Learning feature hierarchies for musical audio signals*. PhD Thesis, Ghent University, 2015.

Sander Dieleman and Benjamin Schrauwen. Multiscale approaches to music audio feature learning. In *14th International Society for Music Information Retrieval Conference (ISMIR-2013)*, pages 116–121. Pontifícia Universidade Católica do Paraná, 2013.

Sander Dieleman and Benjamin Schrauwen. End-to-end learning for music audio. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 6964–6968. IEEE, 2014.

Aleksandr Diment, Padmanabhan Rajan, Toni Heittola, and Tuomas Virtanen. Modified group delay feature for musical instrument recognition. In *Proc. Int. Symp. Computer Music Multidisciplinary Research*, pages 431–438, 2013.

Zhiyao Duan, Bryan Pardo, and Laurent Daudet. A novel cepstral representation for timbre modeling of sound sources in polyphonic mixtures. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 7495–7499. IEEE, 2014.

Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Douglas Eck, Karen Simonyan, and Mohammad Norouzi. Neural audio synthesis of musical notes with wavenet autoencoders. *arXiv preprint arXiv:1704.01279*, 2017.

Yariv Ephraim and David Malah. Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(6):1109–1121, 1984.

Antti Eronen and Anssi Klapuri. Musical instrument recognition using cepstral coefficients and temporal features. In *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, volume 2, pages II753–II756. IEEE, 2000.

Thomas Esch and Peter Vary. Efficient musical noise suppression for speech enhancement system. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4409–4412. IEEE, 2009.

Slim Essid, Gaël Richard, and Bertrand David. Musical instrument recognition on solo performances. In *Signal Processing Conference, 2004 12th European*, pages 1289–1292. IEEE, 2004.

Ferdinand Fuhrmann, Martín Haro, and Perfecto Herrera. Scalability, generality and temporal aspects in automatic recognition of predominant musical instruments in polyphonic music. In *ISMIR*, pages 321–326, 2009.

Takuya Fujishima. Realtime Chord Recognition of Musical Sound: a System Using Common Lisp Music. In *ICMC*, pages 464–467, 1999.

Pegah Ghahremani, Vimal Manohar, Daniel Povey, and Sanjeev Khudanpur. Acoustic Modelling from the Signal Domain Using CNNs. In *INTERSPEECH*, pages 3434–3438, 2016.

Zenton Goh, Kah-Chye Tan, and TG Tan. Postprocessing method for suppressing musical noise generated by spectral subtraction. *IEEE Transactions on Speech and Audio Processing*, 6(3):287–292, 1998.

Pavel Golik, Zoltán Tüske, Ralf Schlüter, and Hermann Ney. Convolutional neural networks for acoustic modeling of raw time signal in LVCSR. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. Rwc music database: Popular, classical and jazz music databases. In *ISMIR*, volume 2, pages 287–288, 2002.

Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, and Jianfei Cai. Recent advances in convolutional neural networks. *Pattern Recognition*, 2017.

Siddharth Gururani and Alexander Lerch. Mixing Secrets: A multitrack dataset for instrument detection in polyphonic music. In *Late Breaking Demo (Extended Abstract), Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, 2017. URL

`http://www.musicinformatics.gatech.edu/wp-content_nondefault/uploads/2017/10/Gururani_Lerch_2017_Mixing-Secrets.pdf`.

Philippe Hamel, Yoshua Bengio, and Douglas Eck. Building Musically-relevant Audio Features through Multiple Timescale Representations. In *ISMIR*, pages 553–558, 2012.

Philippe Hamel, Matthew Davies, Kazuyoshi Yoshii, and Masataka Goto. Transfer learning in MIR: Sharing learned latent representations for music audio classification and similarity. 2013.

Yoonchang Han, Subin Lee, Juhan Nam, and Kyogu Lee. Sparse feature learning for instrument identification: Effects of sampling and pooling methods. *The Journal of the Acoustical Society of America*, 139(5):2290–2298, 2016.

Yoonchang Han, Jaehun Kim, Kyogu Lee, Yoonchang Han, Jaehun Kim, and Kyogu Lee. Deep convolutional neural networks for predominant instrument recognition in polyphonic music. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 25(1):208–221, 2017.

Toni Heittola, Anssi Klapuri, and Tuomas Virtanen. Musical instrument recognition in polyphonic audio using source-filter model for sound separation. In *ISMIR*, pages 327–332, 2009.

Francisco Herrera, Sebastián Ventura, Rafael Bello, Chris Cornelis, Amelia Zafra, Dánel Sánchez-Tarragó, and Sarah Vluymans. Multiple instance learning. In *Multiple Instance Learning*, pages 17–33. Springer, 2016.

Eric J Humphrey. *An exploration of deep learning in content-based music informatics*. New York University, 2015.

Eric J Humphrey and Juan Pablo Bello. Rethinking automatic chord recognition with convolutional neural networks. In *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, volume 2, pages 357–362. IEEE, 2012.

Eric J Humphrey, Juan Pablo Bello, and Yann LeCun. Moving Beyond Feature Design: Deep Architectures and Automatic Feature Learning in Music Informatics. In *ISMIR*, pages 403–408. Citeseer, 2012.

Eric J Humphrey, Juan P Bello, and Yann LeCun. Feature learning and deep architectures: New directions for music informatics. *Journal of Intelligent Information Systems*, 41(3):461–481, 2013.

Cyril Joder, Slim Essid, and Gaël Richard. Temporal integration for audio classification with application to musical instrument classification. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(1):174–186, 2009.

Naoyuki Kanda, Ryu Takeda, and Yasunari Obuchi. Elastic spectral distortion for low resource speech recognition with deep neural networks. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 309–314. IEEE, 2013.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Tetsuro Kitahara, Masataka Goto, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G Okuno. Instrument identification in polyphonic music: Feature weighting to minimize influence of sound overlaps. *EURASIP Journal on Applied Signal Processing*, 2007(1): 155–155, 2007.

E. K. Kokkinis, J. D. Reiss, and J. Mourjopoulos. A Wiener Filter Approach to Microphone Leakage Reduction in Close-Microphone Applications. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(3):767–779, March 2012.

AG Krishna and Thippur V Sreenivas. Music instrument recognition: from isolated notes to solo phrases. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, volume 4, pages iv–iv. IEEE, 2004.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Yann LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–50. Springer, 1998.

Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.

Honglak Lee, Peter Pham, Yan Largman, and Andrew Y. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in neural information processing systems*, pages 1096–1104, 2009.

Jongpil Lee and Juhan Nam. Multi-level and multi-scale feature aggregation using pretrained convolutional neural networks for music auto-tagging. *IEEE signal processing letters*, 24(8):1208–1212, 2017.

Jongpil Lee, Jiyoung Park, Keunhyoung Luke Kim, and Juhan Nam. Sample-level deep convolutional neural networks for music auto-tagging using raw waveforms. *arXiv preprint arXiv:1703.01789*, 2017.

Alexander Lerch. *An introduction to audio content analysis: Applications in signal processing and music informatics*. John Wiley & Sons, 2012.

Beth Logan and others. Mel Frequency Cepstral Coefficients for Music Modeling. In *ISMIR*, volume 270, pages 1–11, 2000.

Alexey Lukin and Jeremy Todd. Suppression of musical noise artifacts in audio noise reduction by adaptive 2-D filtering. In *Audio Engineering Society Convention 123*. Audio Engineering Society, 2007.

Brian McFee, Eric J. Humphrey, and Juan Pablo Bello. A Software Framework for Musical Data Augmentation. In *ISMIR*, pages 248–254. Citeseer, 2015.

Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio. SampleRNN: An unconditional end-to-end neural audio generation model. *arXiv preprint arXiv:1612.07837*, 2016.

Noam Mor, Lior Wolf, Adam Polyak, and Yaniv Taigman. A universal music translation network. *arXiv preprint arXiv:1805.07848*, 2018.

Meinard Muller, Daniel PW Ellis, Anssi Klapuri, and Gaël Richard. Signal processing for music analysis. *IEEE Journal of Selected Topics in Signal Processing*, 5(6):1088–1110, 2011.

Gautham J Mysore, Paris Smaragdis, and Bhiksha Raj. Non-negative hidden markov modeling of audio with application to source separation. In *International Conference on Latent Variable Analysis and Signal Separation*, pages 140–148. Springer, 2010.

Meinard Müller. *Information retrieval for music and motion*, volume 2. Springer, 2007.

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

Douglas O'shaughnessy. *Speech communication: human and machine*. Universities press, 1987.

Alexey Ozerov, Emmanuel Vincent, and Frédéric Bimbot. A general flexible framework for the handling of prior information in audio source separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(4):1118–1133, 2012.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

Taejin Park and Taejin Lee. Musical instrument sound classification with deep convolutional neural network using feature fusion approach. *arXiv preprint arXiv:1512.07370*, 2015.

Karol J Piczak. Environmental sound classification with convolutional neural networks. In *Machine Learning for Signal Processing (MLSP), 2015 IEEE 25th International Workshop on*, pages 1–6. IEEE, 2015.

Jordi Pons and Xavier Serra. Designing efficient architectures for modeling temporal features with convolutional neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 2472–2476. IEEE, 2017.

Jordi Pons, Thomas Lidy, and Xavier Serra. Experimenting with musically motivated convolutional neural networks. In *Content-Based Multimedia Indexing (CBMI), 2016 14th International Workshop on*, pages 1–6. IEEE, 2016.

Jordi Pons, Oriol Nieto, Matthew Prockup, Erik M. Schmidt, Andreas F. Ehmann, and Xavier Serra. End-to-end learning for music audio tagging at scale. *arXiv preprint arXiv:1711.02520*, 2017.

T. Prätzlich, R. M. Bittner, A. Liutkus, and M. Müller. Kernel Additive Modeling for interference reduction in multi-channel music recordings. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 584–588, April 2015.

Jukka Pätynen, Ville Pulkki, and Tapio Lokki. Anechoic recording system for symphony orchestra. *Acta Acustica united with Acustica*, 94(6):856–865, 2008.

Anton Ragni, Kate M Knill, Shakti P Rath, and Mark JF Gales. Data augmentation for low resource languages. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

Tara N Sainath, Ron J Weiss, Andrew Senior, Kevin W Wilson, and Oriol Vinyals. Learning the speech front-end with raw waveform CLDNNs. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

Justin Salamon and Juan Pablo Bello. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, 24(3):279–283, 2017.

Justin Salamon, Duncan MacConnell, Mark Cartwright, Peter Li, and Juan Pablo Bello. Scaper: A library for soundscape synthesis and augmentation. In *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2017 IEEE Workshop on*, pages 344–348. IEEE, 2017.

Markus Schedl, Emilia Gómez, Julián Urbano, and others. Music information retrieval: Recent developments and applications. *Foundations and Trends® in Information Retrieval*, 8(2-3):127–261, 2014.

Jan Schlüter. *Deep Learning for Event Detection, Sequence Labelling and Similarity Estimation in Music Signals*. PhD Thesis, Johannes Kepler University Linz, Austria, July 2017.

Jan Schlüter and Thomas Grill. Exploring Data Augmentation for Improved Singing Voice Detection with Neural Networks. In *ISMIR*, pages 121–126, 2015.

Jürgen Schmidhuber. Deep Learning in neural networks: An overview. *Neural Networks*, 61:85–117, January 2015.

Fabian Seipel and Alexander Lerch. Multi-track crosstalk reduction using spectral subtraction. In *Audio Engineering Society Convention 144*. Audio Engineering Society, 2018.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Ting-Wei Su, Jen-Yu Liu, and Yi-Hsuan Yang. Weakly-supervised audio event detection using event-specific gaussian filters and fully convolutional networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 791–795. IEEE, 2017.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

Naoya Takahashi, Michael Gygli, Beat Pfister, and Luc Van Gool. Deep convolutional neural networks and data augmentation for acoustic event detection. *arXiv preprint arXiv:1604.07160*, 2016.

Zoltán Tüske, Pavel Golik, Ralf Schlüter, and Hermann Ney. Acoustic modeling with deep neural networks using raw time signal for LVCSR. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

Stefan Uhlich, Franck Giron, and Yuki Mitsufuji. Deep neural network based instrument extraction from music. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2135–2139. IEEE, 2015.

Stefan Uhlich, Marcello Porcu, Franck Giron, Michael Enenkl, Thomas Kemp, Naoya Takahashi, and Yuki Mitsufuji. Improving music source separation based on deep neural networks through data augmentation and network blending. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 261–265. IEEE, 2017.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Aaron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis C. Cobo, and Florian

Stimberg. Parallel WaveNet: Fast High-Fidelity Speech Synthesis. *arXiv preprint arXiv:1711.10433*, 2017.

Aäron Van Den Oord, Sander Dieleman, and Benjamin Schrauwen. Transfer learning by supervised pre-training for audio-based music classification. In *Conference of the International Society for Music Information Retrieval (ISMIR 2014)*, 2014.

Emmanuel Vincent, Rémi Gribonval, and Cédric Févotte. Performance measurement in blind audio source separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(4):1462–1469, 2006.

Tuomas Virtanen, Mark D Plumbley, and Dan Ellis. *Computational Analysis of Sound Scenes and Events*. Springer, 2018.

Gregory H Wakefield. Mathematical representation of joint time-chroma distributions. In *Advanced Signal Processing Algorithms, Architectures, and Implementations IX*, volume 3807, pages 637–646. International Society for Optics and Photonics, 1999.

Li-Fan Yu, Li Su, and Yi-Hsuan Yang. Sparse cepstral codes and power scale for instrument identification. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 7460–7464. IEEE, 2014.