Technische Universität Ilmenau
Fakultät für Elektrotechnik und Informationstechnik

Automatic Instrument Recognition using Deep Convolutional Neural Networks

Master Thesis at Fraunhofer Institute for Digital Media Technology

Handed in by:	Juan Sebastián Gómez Cañón
Handed in on:	March 31, 2018
Course of study:	Media Technology
Specialisation:	Audio Technology
Matriculation Number:	57010

Responsible Professor:	Prof. Dr.-Ing. Dr. rer. nat. h.c. mult. Karlheinz Brandenburg
Faculty Advisors:	Dr.-Ing. Jakob Abeßer
	Dr.-Ing. Estefanía Cano

Abstract

In the context of growing digital media and new classification/indexing demands, the task of Automatic Instrument Recognition in the field of Music Information Retrieval (MIR) has increasing importance. Through the use of deep learning techniques, namely convolutional neural networks, and different automatic source separation algorithms, developed at the Fraunhofer Institut für Digitale Medientechnologie (IDMT)[1, 2], this Master thesis investigates this recognition task and how different pre-processing stages can improve its classification performance. Several experiments have been conducted in order to reproduce and improve upon the results of the reference system reported by Han et al. [3]. Two systems are proposed in this research: an improved system using harmonic/percussive separation and post-processing using class-wise thresholding, and a combined system using solo/accompaniment separation and transfer learning methods for the special use case of jazz solo recognition. To validate the obtained results, diverse tests have been performed with multiple music data sets, with different complexities and instrument selections.

Kurzfassung

Im Kontext digitaler Medien und neuer Klassifikation- / Indizierungsanforderungen gewinnt die Aufgabe der automatischen Instrumentenerkennung im Bereich des Music Information Retrieval (MIR) zunehmend an Bedeutung. Mit Hilfe von Deep-Learning-Methoden wie Convolutional Neural Networks (CNN) und verschiedenen automatischen Quellenseparationsalgorithmen, die am Fraunhofer Institut für Digitale Medientechnologie (IDMT) entwickelt wurden [1, 2], untersucht diese Masterarbeit, wie die Leistungsfähigkeit der entsprechenden Klassifikationsalgorithmen durch unterschiedliche Vorverarbeitungsstufen verbessert werden kann. Mehrere Experimente wurden durchgeführt, um die Ergebnisse der Forschung Han et al. [3] reproduzieren und verbessern zu können. Im Rahmen dieser Forschung werden zwei Systeme vorgeschlagen: ein verbessertes System, das auf einen harmonic / percussive Separationsalgorithmus sowie Nachbearbeitung durch Klassenweise Entscheidungsschwellwerte aufbaut und ein kombiniertes System, das einen solo / accompaniment-Separationalgorithmus und Transfer Learning für den speziellen Anwendungsfall der Erkennung von Soloinstrumenten in Jazzaufnahmen verwendet. Für die Validierung der Ergebnisse wurden verschiedene Tests mit mehreren Musikdatensätzen mit unterschiedlicher Komplexität und Instrumentenauswahl durchgeführt.

Acknowledgements

Thanks to my family and Andrea.

Table of Contents

1 Introduction

In the group Semantic Music Technologies at the Fraunhofer IDMT, the main research focus is on innovative technologies from MIR that allow to extract semantic features such as key, notes, instruments, tempo, and time signature from music recordings. In this field, the task of automatically identifying musical instruments from polyphonic audio recordings is very desirable, due to the high amount of application scenarios including audio auto-tagging, instrument-specific audio equalization, music recommendation services, audio source separation, automatic music transcription, and genre classification.

Given these MIR applications and the new dimensions of music availability and consumption in recent years, new challenges arise in providing meaningful access to the information that music contains in the context of media technology. Regarding the specific scope of this thesis, the instrumentation of a musical piece is one of the critical factors while processing music, cognitively and perceptually, since it determines the timbre of the piece [4]. Timbre is a perceptual property of sound and is the *"attribute whereby a listener can judge two sounds as dissimilar using any criterion other than pitch, loudness, and duration"* [5]. Concretely, this attribute allows humans to differentiate the sounds produced by a clarinet and a guitar, regardless if the pitch and loudness of both sounds are identical. Research on instrument recognition investigates the correlation between timbre and objective physical sound characteristics (both spectral and temporal audio features), the existence of tonal and noise-like elements in the audio, or the energy distribution in the partials of a tone.

Automatic instrument recognition has been investigated for more than 20 years [6, 7, 8, 9, 10, 11, 12], but recent developments in the field of machine learning (particularly through the use of deep neural networks) have led to significant performance improvements. This is the case of the research by Han et al. [3], in which a deep convolutional neural network is used for predominant instrument recognition using real-world polyphonic and multitimbral music. This convolutional neural network was trained with the IRMAS data set of polyphonic musical audio excerpts (assembled by the Music Technology Group of the Universitat Pompeu Fabra in Barcelona, Spain [6]) to recognize the instrumentation of 11 pitched instruments in multitimbral mixtures.

1.1 Motivation

Deep learning techniques, concretely convolutional neural networks, have been widely and successfully used in the image processing field (e.g., image recognition, video analysis), and are now increasingly being used on music signals. CNNs have now outperformed previous state-of-the-art methods in MIR for automatic chord detection [13], blind source separation [14], and music structure analysis [13]. In the case of automatic instrument recognition, Han et al. [3] achieved a 23.1% and 18.8% performance improvement compared with previous state-of-the-art algorithms [6, 15]. In the case of automatic source separation, Uhlich et al. [16] used a deep neural network for instrument extraction from a polyphonic mixture, given prior knowledge of the type of instruments in the audio file. This approach reports an overall improved signal-to-distortion ratio than previous approaches.

Current research in automatic drum transcription of polyphonic music with recurrent neural networks [17] leads to very interesting application scenarios in the improvement of automatic music transcription tools through the use of instrument recognition and source separation algorithms. A music transcription system could take polyphonic audio material as input, separate it into independent instrument streams through source separation, and translate it into traditional music scores through pitch detection. The instrument recognition module could be used to generate a complete score, since the notes of different instruments should be notated on different staves. This module could also prove to be important characterizing musical pieces or recognizing genres, considering that names of musical compositions or ensembles are defined by the names of the instruments in the music (e.g. "*piano* sonata" or "*string* quartet")[18].

Furthermore, the increasing digitalization of raw audio signals has become critical in our listening music habits. Music digitalization can also gain benefits from the improvement of automatic instrument recognition, considering that the information could be included in the audio metadata or audio-tags. This metadata could be used by users to search for music with a particular instrument or equalize a mixture to give more clarity to a given instrument [3]. Different application scenarios in music education fields are countless and very exciting. An example is Songs2See [19], in which the development of a music game (based on pitch-detection, sound separation, music transcription, and audio analysis) can provide the user a more practical and enjoyable way to learn to play an instrument.

1.2 Objectives

Automatic instrument recognition has attempted to solve cases of different complexities: single-note scenarios [7], monotimbral musical phrases [8], and artificially generated polyphony [20] with satisfactory performance. The success in these previous examples leads to the challenge of this research: accurate instrument recognition in ”real-world” scenarios with polyphonic, multitimbral (simultaneous multiple instruments with intrinsic timbre features), and reverberant audio mixtures. The objectives completed in this Master thesis are the following:

1. Implementation of the state-of-the-art system developed by Han et al. [3] as a baseline system. The reference system shall be implemented in Python and the Keras Deep Learning package [21].
2. Reproduce the results reported by the reference system to validate a correct implementation.
3. Test data sets with different complexity levels and instrument selections for evaluation purposes.
4. Evaluate the use of automatic source separation algorithms as pre-processing to the system. The automatic source separation algorithms that were evaluated are phase-based harmonic/percussive separation [1] and pitch-informed solo/accompaniment separation [2].

1.3 Overview

This Master thesis contains six chapters. Chapter 2 (Scientific Background), presents the relationship between timbre recognition and physical properties of audio. It also includes the classification of instrument recognition systems based on pattern recognition methods. The methods applied for instrument recognition using machine learning consist of two categories: unsupervised and supervised learning algorithms. Finally, this chapter contains an introduction of the key concepts of deep learning that were used throughout this research: convolutional neural networks, activation functions, and transfer learning.

Chapter 3 (State of the Art) contains a collection of several approaches for automatic instrument recognition in polyphonic music. Regarding the input of the system, there is a clear differentiation between systems which employ handcrafted audio features, and those that carry out automatic feature learning. Additionally, the use of sound separation algorithms as a pre-processing stage has been implemented in several systems to improve the overall recognition performance.

Chapter 4 (Baseline Systems) introduces the algorithms that have been used in this Master thesis. There is an initial description of the instrument recognition framework developed by [3] separated into four stages: pre-processing, feature processing, classification, and post-processing [4]. Finally, the automatic audio source separation algorithms developed at the Fraunhofer IDMT [1, 2] are described and the parameters that were used to batch process the data sets.

Chapter 5 (Proposed Experiments and Results) consists on the proposed methods in this Master thesis. It presents an analysis of the different data sets used to test the system and the performance metrics used to evaluate the system. As part of the proposed experiments, the reproduction of the evaluation results of the reference system shows that the baseline system has been successfully recreated. Tests have been carried out with monotimbral data to motivate the use of source separation algorithms. The proposed experiments have been classified into the four stages introduced by Fuhrmann [4]. All of these experiments were carried out in order to improve the recognition performance of the system. The effect of the automatic audio source separation algorithms as a pre-processing stage is evaluated, considering monotimbral and multitimbral scenarios. Finally, an improved model is proposed with the results of previous experiments, and a use case for predominant solo instrument classification in jazz music is evaluated, using transfer learning methods.

Finally, Chapter 6 (Conclusions) presents a discussion of the results, final comments, and the outlook that could be implemented from this Master thesis.

2 Scientific Background

The relationship between timbre recognition and physical properties from the audio signals is the foundation of instrument recognition algorithms. Automatic instrument recognition can be explained from three different algorithmic perspectives: different audio feature extraction systems, different methodologies to recognize patterns of instrumentation, and different types of learning algorithms applied in the classification stage. This chapter concludes with the basic knowledge on deep convolutional neural networks used in the proposed experiments.

2.1 Timbre and Audio Features

The majority of music processing tasks in MIR involve the extraction of suitable audio features that contain relevant information and suppress irrelevant data. Müller [5] states that the information regarding the timbre of an instrument (in the most simple single-note scenario), can be described by the following physical properties:

- *Amplitude envelope*: contains the evolution of the sound over time, divided into four phases (ADSR). The attack phase is a short phase in which there is a sudden increase of acoustic energy. This phase also contains non-periodic information from the signal (transient) that establishes how the instrument sounds (e.g., distinguishing the sound of a hammer hitting the strings on a piano, a string from a guitar being plucked, and the string of a violin being bowed). The decay phase is the period in which the energy decreases to stability. The phase in which the energy is stable and has a roughly periodic is known as the sustain phase. Finally, the release phase is the state when the sound fades away. Refer to the amplitude envelope in Figure 2.1, plotted in red to see the temporal differences between a note played by a piano and a violin.

- *Modulation*: indicates periodic variations in sound. The variations can be in energy (amplitude modulation) or time (frequency modulation). In musical terms, amplitude modulation corresponds to a tremolo, while frequency modulation corresponds to a vibrato. Both tremolo and vibrato depend on the amount and rate of variation, and do not necessarily correspond to a perceived change in loudness or pitch. Refer to Figure 2.1 to see the tremolo in the violin sustain phase amplitude plot and the vibrato in the violin spectrogram plot.

- *Partial information*: the most important property to describe timbre is the dominant frequencies of a musical tone (known as partials) and their relative strengths. The lowest partial is defined as the fundamental frequency and inharmonicity is defined as the deviation of the partials to the ideal harmonics (integer multiples of the fundamental frequency). The variation of the relative strength amongst partials result in a perceived timbre variation between instruments. For example, a clarinet will present more energy in the odd harmonics, a bassoon will show less energy on the fundamental frequency compared to higher partials, bells will have significant amounts of inharmonicities, and stringed instruments will show sizable deviation between the higher partials and the ideal harmonics. Refer to the general differences of relative strength between the two spectrograms in Figure 2.1, which shows the intensity of frequencies on a time-frequency representation.

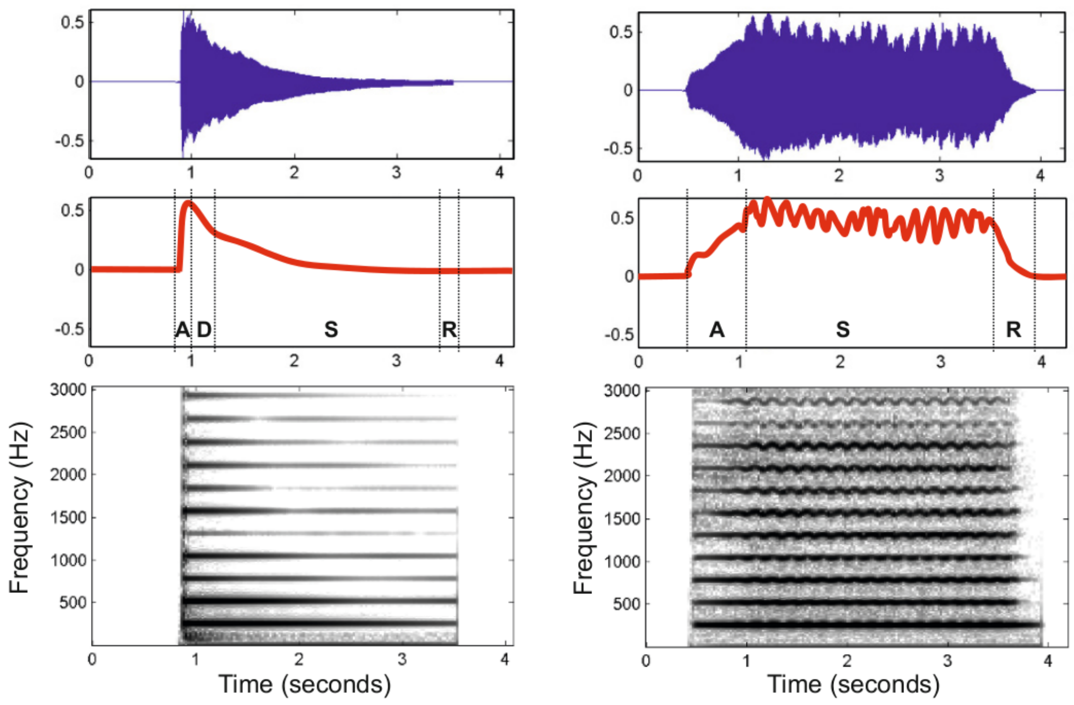


Figure 2.1: Waveform, amplitude envelope and spectrogram representation for a piano (left) and a violin (right) playing a C4 note (261.6 Hz) [5].

2.2 Pattern Recognition Methods

Fuhrmann [4] describes three types of methodologies for polyphonic and multitimbral instrument recognition: *pure pattern recognition*, *enhanced pattern recognition*, and *template matching*. The difference between these three methods varies in the complexity of the pre-processing stage and how the audio signal is used in the system:

- *Pure pattern recognition*: the classification is performed directly on the audio signal, releasing the constraints on the data or the labels. Concretely, these systems apply the knowledge derived from monophonic scenarios directly to the polyphonic audio signal, resulting in a non-existent or unsubstantial pre-processing stage.
- *Enhanced pattern recognition*: the recognition task is enhanced by the incorporation of additional knowledge about the source signals to solve the problem of source interference. In this methodology, techniques such as pitch or onset detection, and source separation are used as a pre-processing stage. The pattern recognition method is then applied on the resulting signals.
- *Template matching*: this methodology involves the use of predefined templates per musical instrument and measuring a distance metric between the predefined template and the polyphonic audio signal.

The algorithms used in this research can be classified using these methods for pattern recognition. In the case of the reference system by Han et al. [3], it can be considered a *pure pattern recognition* method, given that there is no incorporation of additional knowledge on the pre-processing stage. Both phase-based harmonic/percussive separation [1] and pitch-informed solo/accompaniment separation [2] are considered to be *enhanced pattern recognition* methods as a result of using the sound source separated audio streams as input to the system. Furthermore, it is important to mention that pitch-informed solo/accompaniment separation works with *a priori* information of pitch data, in order to accomplish the separation. This pitch data is obtained through an automatic music transcription algorithm, further discussed in section 4.2.2.

2.3 Machine Learning Algorithms

In the context of machine learning, there are three different types of learning algorithm: *supervised*, *unsupervised*, and *reinforcement learning algorithms* [22]. Since *reinforcement learning algorithms* are not currently used in automatic instrument recognition tasks, they are not discussed in this research. *Supervised learning algorithms* solve pattern recognition problems, in which the training data includes input vectors along with the corresponding target vectors. In the case of handwritten digit recognition or instrument recognition, the goal is to assign each input vector to a corresponding category from a finite number of labels. *Unsupervised learning algorithms* contain training data that consists of a set of input vectors without any corresponding target values. The aim of these type of algorithms is to detect input vectors that contain similar examples within the data [23].

2.3.1 Supervised Learning Algorithms

This type of learning algorithms use training data as input to the algorithm including the desired solutions, namely the labels. A classification task is a typical supervised learning problem, because it depends on information that is provided prior to the evaluation [24]. In general, the term *supervised learning* originates from the idea that the target is being provided by an instructor or teacher who shows the machine learning system what to do [25]. The following are supervised learning algorithms that have been used in automatic instrument recognition [22]:

- *Nearest neighbor algorithms*: predicts the membership of a sample to a given class based on a distance measure to different sets of training examples in a multi-dimensional feature space. Concretely, it determines the nearest k training instances to the target instance through the use of a distance measure, such as the Euclidean distance.

- *Support vector machines*: defines a multi-dimensional hyperplane that separates the different classes of the training samples, maximizing the distance from the margin and training instances.

- *Neural networks*: loosely based on models for biological neurons, artificial neurons have one or more binary inputs and one binary output [24]. These neurons are assembled into networks to model complex logical functions. This research relies on the use of convolutional neural networks, further explained in section 2.4

2.3.2 Unsupervised Learning Algorithms

On the othe hand, *unsupervised learning algorithms* have no labels that tell the target to the system, which learns to make sense of the data without a guide [25]. The following are unsupervised learning algorithms that have been used in instrument recognition: mixture models, hidden Markov models, independent component analysis, and non-negative matrix factorization [22]. Since none of these methods were implemented in this research, they are not further described.

2.4 Deep Learning

The idea to learn a task by gathering knowledge from experience allows a computer to learn complicated concepts, constructing them from more simple ones. As more simple concepts are built on top of each other, the structure becomes deeper and deeper. This is the reason why this approach is called *deep learning*[25]. Since this research uses specific concepts of deep learning, the rest of this chapter introduces convolutional neural networks, activation functions, and transfer learning.

2.4.1 Convolutional Neural Networks

Historically, the amount of connections between neurons in artificial networks has been constrained by hardware capabilities [25]. As one of the first successful implementation of convolutional neural networks, Krizhevsky et al. [26] developed a system that improved performance in the task of object recognition in the field of computer vision. Their system reduced the state-of-the-art top-5 error rate from 26.1% to 15.3%. Their convolutional neural network consisted of five convolutional layers, max-pooling, fully connected layers, and over 60 million trainable parameters, as seen in Figure 2.2.

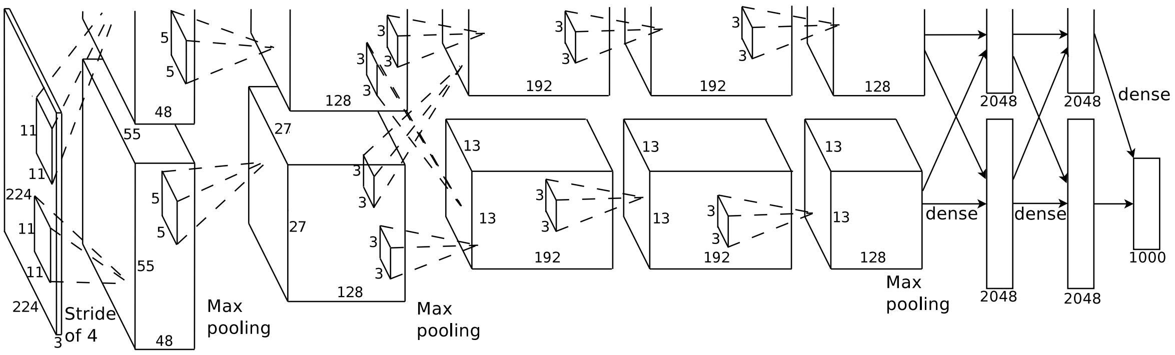


Figure 2.2: Deep convolutional neural network developed by Krizhevsky et al. [26] for object recognition between two GPUs.

In general, artificial neural networks are trained on data that travels through the input layer into the hidden layers, which process sections of the data, remove outliers, and ”distill” the information that finally reaches the output layer [25] (see Figure 2.3). The *input layer* receives the data as an extracted feature and transmits it to the hidden layers. In the case of [26], the input examples were color images with dimensionality of 224 pixels × 224 pixels × 3 channels. In the middle, *hidden layers* contain a significant amount of neurons depending on the type of model, and these neurons are arranged in layers with different functionalities. The feed-forward computation of the output layer is achieved by matrix multiplications from the inputs considering weights and activation functions, throughout the neural network [27]. The *output layer* is the predicted feature and depends on the type of model being built. In the case of [26], it was a fully-connected layer with 1000 activations, which predict the existence of a given object on an image. All cases of convolutional neural networks are supervised learning algorithms, since each training example contains an output label as an annotated ground truth (see Figure 2.3). The output layer is updated to a prediction probability after a *forward propagation* of the data. The prediction probability is then compared to the ground truth of the input. Through the calculation of prediction error and *backward propagation*, the weights of the neurons are adjusted on each iteration [28]. An optimization method is used to minimize the loss function based on the prediction error. Training a neural network is based on learning models with gradient descent like other machine learning algorithms. For this reason, different gradient-based optimizers are used to iteratively minimize the cost function. In this research, Stochastic Gradient Descent [29] and Adam [30] (an extension to SGD) optimizers were used.

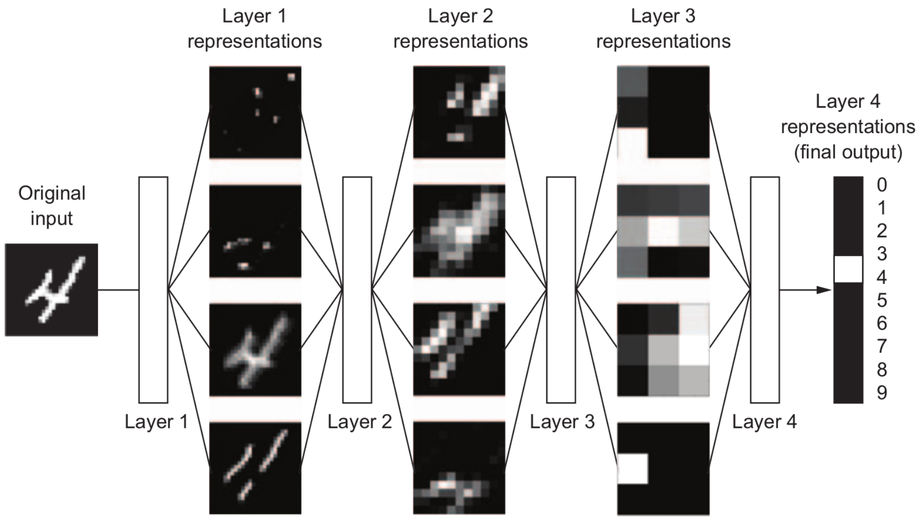


Figure 2.3: Deep convolutional neural network model for handwritten recognition with input, hidden and output layers simplification [28].

Figure 2.4 shows how the input and output of the neuron connects to other neurons throughout the model. All input connections are a weighted by ”strengths” (modeled by a weight w and a bias b), which are learnable and control the influence from one neuron to the next. Consequently, if the final sum carried out in the neuron is above a given threshold, it will then activate the output to the next neuron. In general, the activation functions are used to compute the hidden layer states, differentiate neural networks from other machine learning approaches, and introduce a non-linear transformation that origins non-convex loss functions [25]. An example of activation function is the *sigmoid function*, which takes the real-valued input and compresses it from 0 to 1.

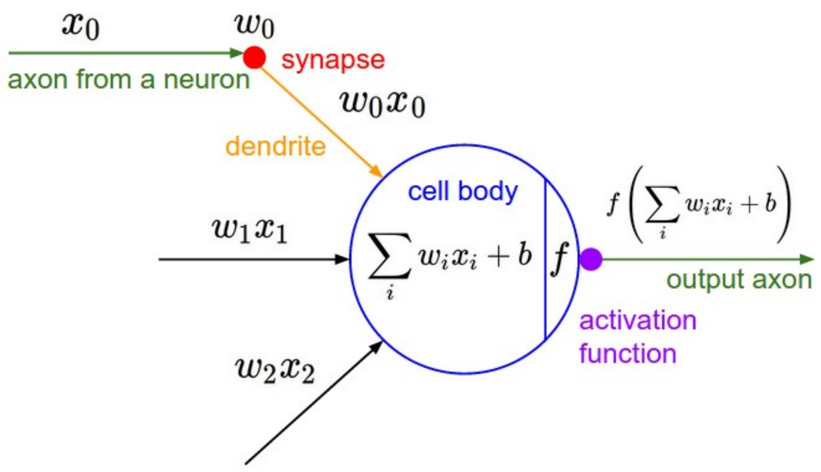


Figure 2.4: Mathematical model of a neuron, showing inputs, outputs, and activation function [31].

In the case of convolutional neural networks, the input is assumed to be an image, motivating the use of filters to extract information from it. A convolutional layer is made up of three stages during feed-forward propagation. In the first stage, several parallel convolutions take place to obtain a set of linear activations. The input can have a padding of zeros around the edges of the image to preserve spatial dimensionality of the convolution. In the case of computer vision, the convolution of an image extracts important information like borders, edges, and textures, as seen in Layer 1 of Figure 2.3. In the second stage, each linear activation is used as input to a detector stage, which introduces the non-linearity or activation function. The different activation functions that have been used in this research are described in the next section. In the third stage, a pooling function is used to reduce the dimensionality of the output for the next layer[25].

Given that a reduction of dimensionality results in progressively reducing the amount of parameters and computation in the networks, it can also control overfitting. Overfitting can appear in any machine learning system and refers to the state in which the algorithm reflects low prediction error on training data, but does not generalize correctly on testing or unseen data [27]. Overfitting can be also controlled by using the dropout technique [32]. This method is a computationally inexpensive approach of regularization, which is based on temporarily and randomly sampling neurons and their connections in the network. Consequently, this prevents neurons from co-adapting and results in improved generalization of the model on testing data. Finally, overfitting can also be further minimized through splitting the training data into training and validation, as seen in section 5.1. Depending on the network architecture, several convolutional layers are finally followed by fully-connected layers, which have full connections to all activations in the previous layer as in a regular neural network. This stage of the network is usually referred to as the classifier layer, which results in the output predictions of the system.

2.4.2 Activation Functions

The activation function introduces a non-linearity that takes the output of the neuron and performs a fixed-mathematical operation on it. Table 2.1 summarizes all the activation functions that were used in this research. Two types of activation functions are considered: the ones in input and hidden layers, and those that are used in the output layer.

With respect to input and hidden layers, two activation functions are used: rectified linear units (ReLU) and leaky rectified linear units (LReLU). The Rectified Linear Unit (ReLU) has become very popular because it greatly accelerates the convergence of SGD given that it is piece-wise linear [26]. On the other hand, ReLU units can be fragile during training and be activated in such a way that they turn to zero and are never activated again. Leaky Rectified Linear Units (LReLU) solve this problem by introducing a leakage parameter α that allows values below zero to pass [33].

Given that the output layers result in prediction probabilities, these values should range between 0 and 1. For this reason, softmax and sigmoid functions are used in the output layers. The softmax function has the property that the sum of all probabilities add up to 1, which makes it useful in models where only one label must be predicted (single-labeled testing data). Conversely, the sigmoid function can add up to different values in the output layer, which makes it useful in prediction scenarios where several activations should be present (multi-labeled testing data) [25, 31].

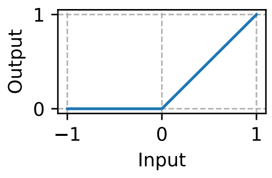
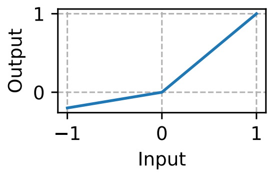
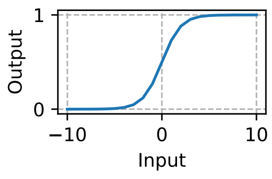
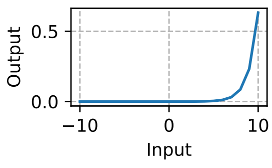
Activation Function	Equation	Plot
Rectified Linear Unit (ReLU)	$f(x) = \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases}$	
Leaky Rectified Linear Unit (LReLU)	$f(x) = \begin{cases} \alpha x & x < 0 \\ x & x \geq 0 \end{cases}$	
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$	
Softmax	$f(x) = \frac{e^x}{\sum^X e^x}$	

Table 2.1: Comparison of activation functions for neural networks.

2.4.3 Transfer Learning

The concept of transfer learning comes from the idea of taking the knowledge that a neural network has learned for a given task and applying it on a new related task. For example, a convolutional neural network trained for object recognition learns to classify between two categories: cats and dogs. This pre-trained model can also be used to classify new categories, such as ants and wasps [25]. Since the neural network has learned low-level representations of the images (e.g., edges, shapes, and patterns), the learned features can be used for a new task with a new training set. However, the relationship between the original task and the new one can vary depending on the input or output of the system. In the object recognition example, the features that are learned in early layers of the network are shared with the new task. In this scenario, the approach is to train the final layers with the new data, while leaving all other weights fixed. Conversely, a speech recognition recurrent neural network must be fine-tuned to recognize different versions of the same phoneme. In this scenario, the classification tasks share the output layers. the suggested approach for this scenario would be to fix the weights of the final layers and only train on the first layers of the network. Since transfer learning is a very novel approach, different approaches and research study how transferable are the features learned by a deep neural network [34].

Two initial approaches are suggested for implementing transfer learning: using the convolutional neural network as a fixed feature extractor, and fine-tuning the network completely. In the first approach, the last fully connected layer is removed and replaced by a new one. The weights of all other layers are maintained fixed and the network is trained only on the classifier output with the new data. In the second approach, the final layer is replaced and the complete network is retrained with the new data. Depending on the new training set, several considerations can be taken into account to perform transfer learning [31]:

- *Small and similar data set*: use the network only as a feature extractor, while using fine-tuning could result in overfitting.
- *Large and similar data set*: use both approaches, given the data set is large, it is less prone to overfit.
- *Small and different data set*: use the network as a feature extractor, while allowing training on early layers to adapt to the specific features of the data set.
- *Large and different data set*: do not implement transfer learning and train a new network from scratch.

3 State Of The Art

According to Fuhrmann [4], the task of automatic instrument recognition can be separated into four independent stages, which can be partially or completely included in the algorithm, depending on the specific approach. The algorithmic structure of an instrument recognition framework can contain:

- *Pre-processing stage*: prior information (such as number of sources in the mixture, pitch estimation, and onset time) can be input to the system to improve the classification. In this stage, automatic source separation algorithms can be used to produce independent audio streams that are then given to the system as input.
- *Feature processing stage*: the audio signal is transformed into a vector of features that describe the physical properties from the signal. The low-level information extracted from the audio is typically divided into short segments, from where the features are then calculated. In several instrument recognition systems, the extracted features can be further processed to decrease redundancy or dimensionality.
- *Classification stage*: a model that has been previously trained predicts the label probabilities for each audio instance presented to the system.
- *Post-processing stage*: the output of the classifier is weighted in order to produce a the final prediction labels of the system. This weighting can be performed using timbral information or classification decision based on neighboring predictions.

The scope of this research is focused on the recognition of instruments in polyphonic recordings, which is the case in real-world recorded music. In these type of mixtures, the interference of simultaneous sounds will very likely limit the performance of the recognition task [12]. As mentioned in the previous chapter, the way in which audio features are extracted results in a new classification of automatic instrument recognition systems: those systems that rely on carefully handcrafting audio features that describe timbre, and those that involve deep learning and use automatic feature learning to train a classifier (see section 3.1).

Additionally, the use of source separation techniques as pre-processing stage to the recognition task has been used for several years to significantly improve the performance metrics of the system. For this reason, several approaches for automatic source separation in polyphonic instrument recognition are mentioned in section 3.2.

3.1 Audio Features in Instrument Recognition

A significant amount of audio features modeling timbre can be extracted from audio segments and are processed to result in a probability of the existence of an instrument in a mixture. Early research in this field uses the case of monotimbral and monophonic audio. In the case of instrument recognition of isolated notes, classification is achieved with the use of spectral and temporal features. The motivation to use spectral and temporal features is to capture discriminant aspects of instruments timbres, such as amplitude envelope, temporal and spectral modulation, and partial information (see section 2.1). Examples of spectral features are: mean of spectral centroid, standard deviation of spectral centroid, fundamental frequency, Mel-Frequency Cepstral Coefficients (MFCCs) [7], modified group delay feature (instrument onset detection by means of the phase slope function) [9], and sparse coding (using sparse cepstral codes and power normalization as a compact representation of cepstral features) [10]. Temporal features include: amplitude envelope, duration of attack, duration of decay, time between end of attack and maximum rms-energy, and crest factor [7]. In the case of instrument recognition of musical phrases, classification is carried out with the use of line spectral frequencies and Gaussian mixture model (GMM) [8], and principal component analysis (PCA) [11].

3.1.1 Handcrafted Audio Features

As mentioned in section 2.2, these approaches carefully design the audio features that are extracted from the audio, use different weighting schemes to differentiate between them, and estimate an output probability. These systems usually apply the knowledge derived from monophonic scenarios directly to the polyphonic audio signal.

Kitahara et al. [18] created a model that weighted the features based upon quantity of overlapping from partials (harmonic components) in polyphonic audio. The polyphony in this research was achieved with artificial mixtures up to a polyphony of four, generated from the RWC instrumental library. They used several spectral, temporal and modulation features as input to linear discriminant analysis (LDA). Spectral features refer to spectral centroid, relative power of fundamental frequency, relative cumulative power from fundamental to partials, relative power in odd and even components, and relative lengths of partial duration. Modulation features included amplitude and frequency of amplitude modulation (AM) and frequency modulation (FM). Temporal features included approximating the power envelope to a line, the derivation of the power envelope from an onset time to different time intervals and time-varying power ratios. LDA was used to minimize within-class and maximize between-class variance, enhancing the extracted features. Principal component analysis (PCA) was used for feature dimensionality reduction, and pitch-dependent Gaussian prototypes were trained for all instruments. Artificial polyphony was achieved by mixing classical music recordings of piano, guitar, violin, clarinet, and flute. They achieved a recognition rate of 84% for a duo, 78% for a trio and 72% for a quartet.

Heittola et al. [20] proposed a system using a source separation model based on a non-negative matrix factorization (NMF) and, simultaneously, a multi-pitch estimator that calculates the fundamental frequencies of the incoming audio. Using the separated streams, a feature extraction component modulates the filters as a sum of elementary functions on the Mel-frequency scale and are recognized with the use of a Gaussian mixture model (GMM) classifier. As audio features, they used MFCCs to represent a coarse shape of the power spectrum and their first-order derivatives to describe the dynamic properties of the cepstrum. They achieved a 59% recognition rate for six polyphonic notes randomly generated from 19 different instruments (including accordion, bassoon, clarinet, contrabass, electric bass, electric guitar, and electric piano, among others).

In the context of single note instrument recognition, the work by Grasis et al. [12] is of particular importance to this Master thesis, given its potential capacity to extend towards polyphonic scenarios and that it was developed at the Fraunhofer IDMT. They proposed a multiple expert framework, which combines classification results of different feature categories. Through the use of an automatic transcription tool for note event detection, the onset, offset and pitch information are used to track the partials of the fundamental frequency and the first 9 overtones. This results in magnitude and frequency tracking for each harmonic component. Magnitude and frequency tracking is used to calculate three categories of audio features: partial-wise, frame-wise, and note-wise features. These features are then classified with the use of a Support Vector

Machine (SVM) algorithm. Finally, there is a two step combination scheme which results in a single vector of class probabilities, as a post-processing stage.

3.1.2 Automatic Feature Learning

Feature learning is the approach through which deep learning techniques allow the system to discover the representation needed for data classification. The concept of feature learning becomes very clear in the context of autoencoders, neural networks trained to attempt to copy their input to their output [25]. In general, an autoencoder has hidden layers that learn a code that represents the input. The goal of these systems is to learn useful properties from the input, in such a way that dimensionality reduction is achieved and the original input can be retrieved at the output of the system. This idea can generalize to convolutional neural networks, in which filters learn to extract key features from the input and train using backpropagation [26].

Only recently convolutional neural networks have been used in the task of instrument recognition. Park et al. [35] proposed a convolutional neural network to recognize instruments using single tone recordings. In their approach, the network has two types of inputs: conventional spectrograms and multi-resolution recurrence plots (MRPs). Their approach takes into account phase information with the MRPs, a square matrix whose elements correspond to the distance between two phase trajectories in time domain. The plots are multi-resolution because they are extracted for different time windows of the signal. They used the University of Iowa Musical Instrument Samples (MIS) database and piano samples from Virtual Studio Technology instruments to assemble a subset of 20 instrument classes, including piano, tuba, trumpet, horn, tenor trombone, bass trombone, violin, viola, among others.

Li et al. [36] used an end-to-end approach of feature learning and multi-label training data to feed their CNN. This system relies less on frequency domain knowledge through the use of raw audio signals. Their model contains three temporal convolutional layers, which are temporal because the length of the filter is the same of the length of the raw audio signal processed. They used the MedleyDB data set to train and evaluate their model, which was assembled into 11 different instrument classes such as electric bass, acoustic guitar, synthesizer, and drum set, among others. They trained the network with multitimbral audio mixtures and generated the labels upon the activation of the instrument in a 100 ms window. This approach achieved a 82.74% accuracy, and significant improvements against a benchmark model. The benchmark model used MFCCs and their first and second derivatives as audio features.

The approach that has been implemented in this Master thesis is the one developed by Han et al. [3] based on a convolutional neural network. This model considers predominant instruments in the mixture, which differs from [36] due to the use of single labeled predominant instruments in the polyphonic mixture. The network was trained with the IRMAS dataset [6] which contains approximately 10.000 audio excerpts from 11 musical instruments (cello, clarinet, flute, acoustic guitar, electric guitar, organ, piano, saxophone, trumpet, violin and voice). Further information regarding the reference system is discussed in Chapter 4.

Additionally, Pons et al. [37] proposed to improve the design of convolutional neural network architectures in different research tasks relating timbre (singing voice phoneme classification, instrument recognition, and music auto-tagging). Their approach is to use domain knowledge in the filter design stage. In this context, they used filters with different shapes that are musically motivated, as opposed to small rectangular filters as Han et al. [3]. In the case of instrument recognition, they achieved very similar performance metrics than baseline model, while reducing the number of trainable parameters from 1.446.000 to 743.000. This is evidence that the field of neural networks is constantly evolving, and several design strategies still have to be researched.

3.2 Sound Separation as Pre-processing Stage

Sound separation methods have been used in instrument recognition systems to further improve the models used for classification, by previously dividing the original multitimbral audio into several streams. This Master thesis uses two source separation algorithms developed at the Fraunhofer IDMT: phase-based harmonic/percussive separation [1] and pitch-informed solo/accompaniment separation [2].

As figure 3.1 shows, automatic source separation algorithms are used to divide polyphonic audio into independent streams. These streams are then used as input to multiple instances of an instrument recognition model. As output to every instance, the system obtains a probability for a given label which is processed again to obtain the final output labels from the mixture.

The work by Bosch et al. [6] is of particular relevance to this Master thesis, not only due to the research on pre-processing stages as a method to improve the system, but also for the creation of the IRMAS data set as a polyphonic, multitimbral, and public data set for instrument recognition. Regarding the use of automatic source separation, they carried out several experiments using two segregation methods: a simple LRMS (Left/Right-Mid/Side) separation and FASST (a Flexible Audio Source Separation

Framework) developed by Ozerov et al. [38]. LRMS separation is made with simple panning calculations, where $M = L + R$ and $S = L - R$. FASST separation is based on introducing constraints that are based on previous knowledge about the separation, separating the audio into four streams: ”drums”, ”bass”, ”melody”, and ”other”. The classification stage was performed with independent Support Vector Machine (SVM) classifiers trained with features extracted from real audio signals. They achieved improvements of 19% of recognition results with the simple panning separation and up to 32% when the model was trained with previously separated audio, taking into account the typical artifacts produced by source separation techniques. As previously mentioned, Heittola developed a system that uses a polyphonic pitch estimation algorithm as input to a NMF source separation algorithm [20].

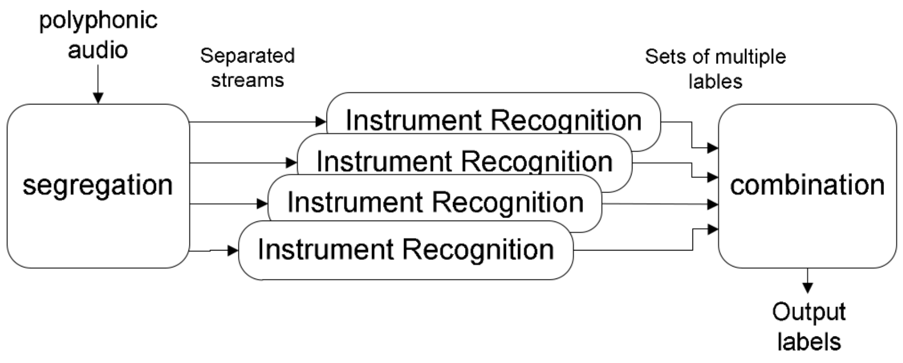


Figure 3.1: Generic model of sound source separation as pre-processing stage to instrument recognition [6].

4 Baseline Systems

The algorithms used in this Master thesis are the instrument recognition system developed by Han et al. [3] and the automatic sound separation algorithms developed at Fraunhofer IDMT [1, 2]. Regarding the implementation of the reference model, the parameters and network design are described thoroughly in section 4.1. Given that the source separation algorithms have only been used to process the data sets using available code, they are briefly described in section 4.2, along with the parameters used in the separation process.

4.1 Baseline Instrument Recognition Framework

The model that was implemented as a baseline is a deep convolutional neural network using high-level time-frequency representations of audio as input. The algorithmic structure introduced at the beginning of Chapter 3 is used to describe the reference system. The general system architecture is presented in Figure 4.1 and is explained in the following sections.

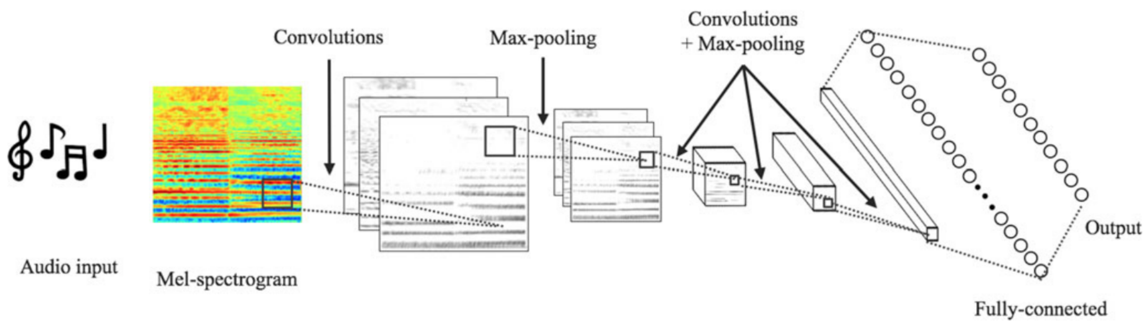


Figure 4.1: Deep convolutional neural network developed by Han et al. [3].

4.1.1 Pre-processing stage

Given that approaches that use deep neural networks have the goal to automatically learn the features needed for classification, the reference model has a very simple pre-processing stage.

1. *Conversion from stereo to mono*: the audio is converted to mono by taking the mean from left and right channels.
2. *Downsampling*: the original audio is downsampled from 44.1 KHz to 22.05 KHz. This results in a Nyquist frequency of 11.025 KHz, which is considered sufficient to contain the most harmonics from the musical instruments, while removing noises generated by higher frequencies.
3. *Time-domain normalization*: All audio signals are divided by their maximum value to obtain a normalized signal from -1 to 1.

4.1.2 Feature processing stage

Following the idea to use high-level representations of the data to train the model, the reference model uses mel-spectrograms as input. The implementation of this stage was performed with *Librosa* library [39], a tool for audio and music signal analysis for Python.

1. *Short Time Fourier Transform*: a linear frequency scale spectrogram is obtained by extracting the STFT of the audio signal using a window size of 1024 and a hop size of 512 time frames.
2. *Mel-spectrogram*: linear frequency scale is converted to mel-scale, using 128 mel-bands to represent frequency.
3. *Logarithmic compression*: the magnitude of the final spectrogram is compressed with a natural logarithm. The final spectrogram $\hat{S} = log(|S| + \epsilon)$, where $|S|$ is the magnitude of the spectrogram and ϵ is the smallest representable floating point type in Python.
4. *Framing*: the resulting spectrograms are finally separated into segments of 1 second, corresponding to 43 time frames. Han et al. [3] obtained the best results with this framing size, after experimenting with 0.5, 1, 1.5, and 3 seconds. The final dimensionality of each spectrogram is then 128 mel-bands by 43 time frames.

4.1.3 Classification stage

The network architecture of the classifier is inspired in image recognition structures like AlexNet and VGGNet, using repeated convolutional layers, max-pooling, and dropout layers that have shown best performance in the field of computer vision [3].

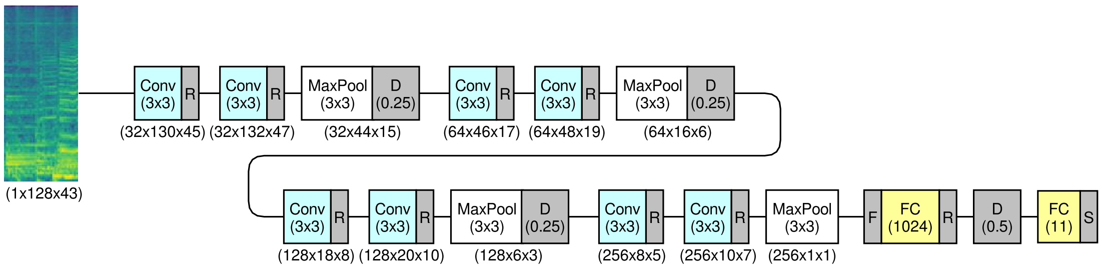


Figure 4.2: Reference model including input dimensionality for 43 time frames equivalent to 1 second (number of filters × frequency bins × time frames).

The general architecture consists of four double convolutional layers with filters that have a small 3×3 receptive fields and a stride size of 1. For each layer, the amount of filters is doubled starting from 32 up to 256, as seen in the dimensionality information of Figure 4.2. Prior to each convolutional layer, there is a 1×1 zero padding layer to apply same convolution. After each convolutional layer there is a 3×3 max-pooling layer for dimensionality reduction. Finally, each double convolutional layer ends with a dropout layer (0.25) to prevent overfitting. After the four double convolutional layers, there is a flattened fully-connected layer, a dropout layer (0.5), and a final output layer with 11 sigmoid outputs. As mentioned in section 2.4, sigmoid activation function allows multiple active labels in multi-labeled data. Han et al. [3] tried different activation functions: ReLU, leaky ReLU, and parametric ReLU (the leakage parameter α is also learned during training). They found best results using ReLU activation (denoted R in Figure 4.2) and sigmoid output (denoted S).

The model was trained using an Adam optimizer with a learning rate of 0.001 to minimize categorical cross-entropy, using mini-batch size of 128, early stopping after 10 epochs with no decrease in validation loss, and uniform-distributed Glorot-Bengio initialization on every layer [25].

4.1.4 Post-processing stage

The final stage is designed to optimize clip-wise predictions for the testing data. Obtaining a clip-wise prediction output refers to post-processing all segment predictions (which have a duration of 1 second), and obtain a final prediction for a complete audio clip. The test audio is separated into segments of 1 second with a 50% overlap, and the segment-wise predictions are aggregated, as shown in Figure 4.3. The authors designed two different aggregation strategies:

- *S1*: summation over all segment-wise predictions, averaging over all frames, normalization by dividing over the maximum value across labels, and thresholding. This strategy is based on the assumption that humans perceive the predominant instrument judging its relative strength to other instruments.
- *S2*: max-pooling is performed on a sliding window of 6 and hop size of 3 segments. *S1* is performed afterwards with the resulting max-pooled segment-wise predictions. This strategy is based on the assumption that sporadic activations could be suppressed when averaging, which is reduced by temporal local max-pooling.

Variable thresholding is used to identify if the instrument is labeled as predicted or not. Using a higher value of threshold will lead to a better precision and reduce the recall. Conversely, a lower threshold will lead to a lower precision and higher recall. The identification threshold has been varied from 0.0 to 1.0 to obtain an optimal setting, which is further discussed in section 5.2.

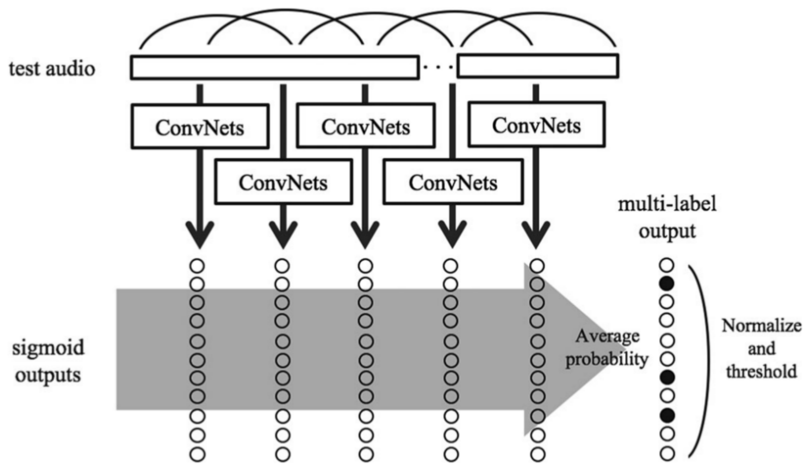


Figure 4.3: General aggregation scheme for clip-wise predictions of multi-labeled output [3].

4.2 Baseline Source Separation Algorithms

4.2.1 Phase-Based Harmonic-Percussive Separation

The method to separate harmonic and percussive elements in music recordings was developed by Cano et al. [1]. This algorithm takes into account that percussive components appear as vertical (broadband) elements in the spectrogram, while harmonic components are generally horizontal. The main idea behind the method is to use frame-wise phase expectation of the unwrapped phase, given the pitch of the source. Phase expectation can be accurate in the case that the signal varies smoothly. If the phase change between two frames lies on the calculated expected range, the source is assumed to be harmonic. On the other hand, if it falls outside the expected range, it is assumed to be percussive. For each time frame, the partials in the magnitude spectrum are detected and spectral masks are then calculated, depending on the resulting phase change. Finally, the percussive and harmonic time-domain signals are obtained through the Inverse Short Time Fourier Transform (ISTFT).

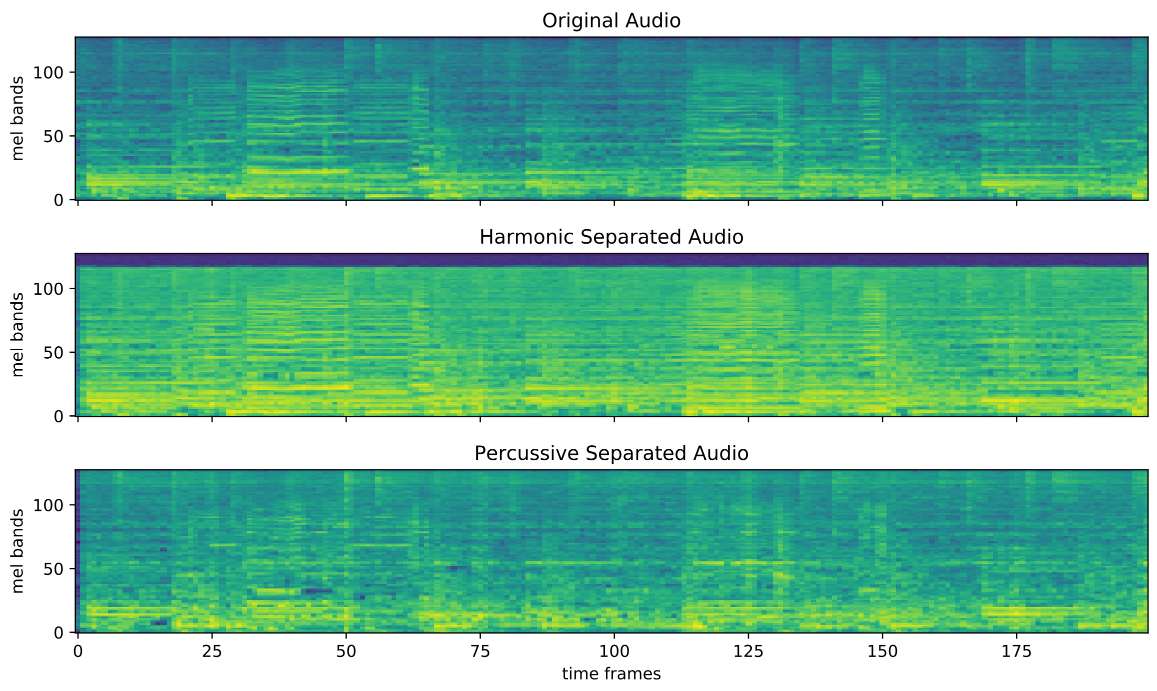


Figure 4.4: Mel-spectrogram of harmonic/percussive separated audio (John Coltrane - A Love Supreme) with a duration of 200 time frames ≈ 4.6 seconds

Both phase and magnitude spectra are calculated through a Short Time Fourier Transform (STFT) with a Hanning window of 2048 and hop size of 128 time frames.

The peak detection in the magnitude spectrum is only calculated for the range of 0-8000 Hz, to have a better separation of snare drum, as suggested by the author.

Figure 4.4 shows the mel-spectrograms of a jazz excerpt and its corresponding separated audio streams. Given that peak detection is performed up to 8000 Hz, time-frequency representation from mel-band 117 to 128 in the harmonic component has no energy. This image also shows that the harmonic separated audio contains principally horizontal elements, while the majority of vertical elements from the original spectrogram appear in the percussive separated component.

4.2.2 Pitch-Informed Solo-Accompaniment Separation

The use of this algorithm is based on a two stage approach: automatic pitch detection and performing source separation based on this information. Regarding the solo, it is defined as a single (monophonic) pitch sequence that the listener would regard as the "essence" of the music excerpt. The algorithm focuses on the monophonic case, where the solo instrument is assumed to only play one note at a time [2]. The pitch detection algorithm used is the one proposed by Dressler [40], which obtains a spectral representation from the signal, detects the possible pitch candidates, forms the tones and voices (through the use of onset detection and partials estimation), and selects the main melody. Concretely, if the pitch detection algorithm does not detect a note, these frames are not considered in the separation. As Figure 4.5 shows, the mel-spectrogram of the solo separated audio does not exist in the frames in which a pitch was not detected.

The source separation is performed through the use of a harmonic series estimation, that represents the solo instrument and is consistent with the tones estimated in the pitch detection stage. The two principles used for this estimation are: each partial is allowed to have independent inharmonicity from the ideal harmonic and differences in inharmonic properties are considered depending on the instrument family [2]. Even though there is the possibility of using the harmonic refinement mentioned in the second principle, given that the algorithm was used to process the complete data sets which contain multiple different instruments, the information of the instrument family was not taken into account when performing the separation. The magnitude spectrum is calculated through a STFT with a Hanning window of 2048 and hop size of 256 time frames. Maximum 20 harmonics are estimated for each pitch. The magnitude spectrum is then masked, with the same principle as in harmonic/percussive separation. As a post-processing step, there is an attempt of removing attacks from the filtered audio

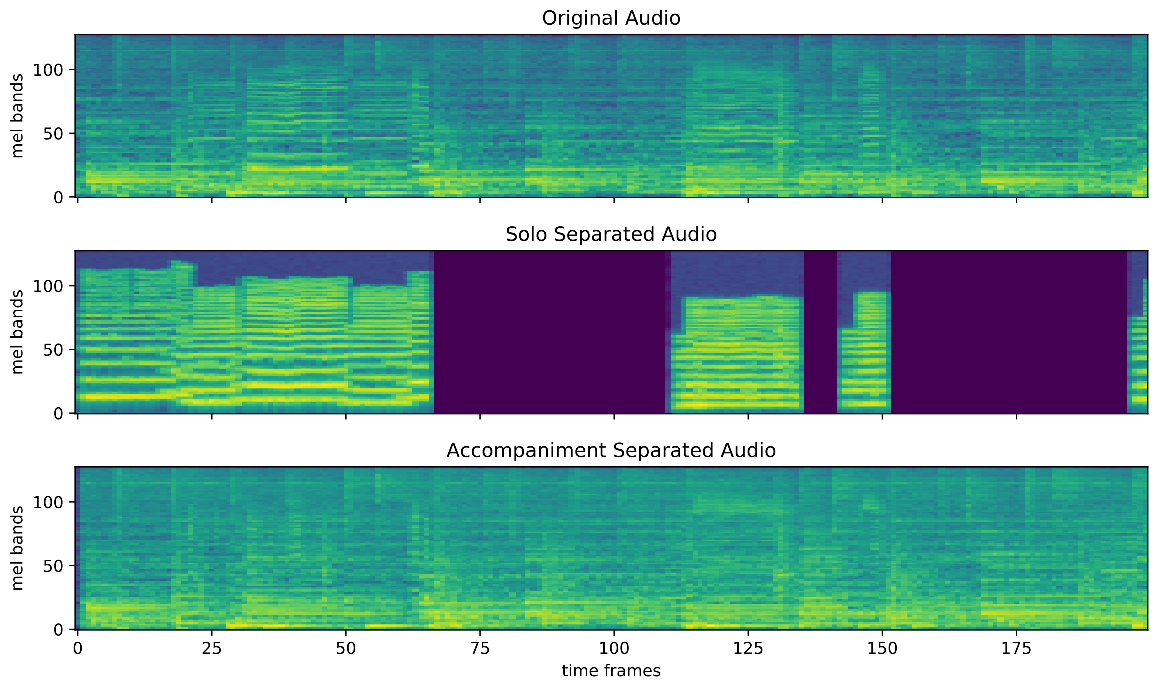


Figure 4.5: Mel-spectrogram of solo/accompaniment separated audio (John Coltrane - A Love Supreme) with a duration of 200 time frames ≈ 4.6 seconds

that can be percussive elements from the original mix that are being leaked into the separated audio. Finally, the solo and backing tracks are retrieved through an ISTFT to re-synthesize the separated audio.

Figure 4.5 shows how the extracted partials constitutes the solo separated audio, while the harmonic elements that are in the original signal corresponding to accompaniment instruments are maintained in the backing separated audio.

5 Proposed Experiments and Results

The baseline instrument recognition model was developed and evaluated using the IRMAS data set [3]. This research has used alternative data sets with different complexity and instrument classes, in order to test the validity and generalization of the obtained results. Section 5.1 includes the information about the data sets used and how they have been split into training and testing data. In order to reproduce the results from the baseline model, the same evaluation metrics have been estimated as those used in previous research. Section 5.2 contains a description of the performance metrics and confusion matrices employed. In order to guarantee a successful implementation of the baseline system, section 5.3 compares the original results reported by Han et al. [3] with the ones obtained in this research. The baseline model has been trained with the Monotimbral data set, to evaluate the impact of data complexity, activation functions, and aggregation strategies. Section 5.4 contains the proposed experiments that have been carried out to evaluate the impact of the source separation algorithms as pre-processing stage. Sections 5.5, 5.6, and 5.7 present additional experiments that have been performed in other stages of the instrument recognition algorithm, in order to improve the overall performance of the baseline system. Section 5.8 presents a final model using harmonic/percussive separation and post-processing to improve recognition performance. Finally, section 5.9 outlines the use case of the Jazz Solo Instrument data set, in order to evaluate transfer learning methods.

5.1 Data sets

5.1.1 IRMAS Data set

The IRMAS data set (Instrument Recognition in Music Audio Signals) was created at the Universitat Pompeu Fabra for training and testing predominant instrument recognition systems [6]. It was previously compiled as two independent sets: training and testing data sets. The complete data set has a bit depth of 16 bits per sample and sampling frequency of 44.1 KHz. The training data set contains 6705 stereo audio files with a fix duration of 3 seconds, obtained from more than 2000 recordings. This data set contains music in which only one instrument is predominant, namely single-labeled data. The amount of samples between classes is unevenly distributed, ranging from 388 to 778 audio files on different classes. The testing data set consists of 2874 stereo audio files with variable duration (ranging from 5 to 20 seconds) and 1-5 labels per sample. Given that an audio file can be annotated with several labels, this data set is considered multi-labeled. This data set is also unevenly distributed ranging from 62 to 1044 audio samples on different classes.

The 11 instruments that are annotated and their abbreviation in this data set are: cello (cel), clarinet (cla), flute (flu), acoustic guitar (gac), electric guitar (gel), organ (org), piano (pia), saxophone (sax), trumpet (tru), violin (vio), and voice (voi). The data has previously been distributed between classes, as seen in Figure 5.1.

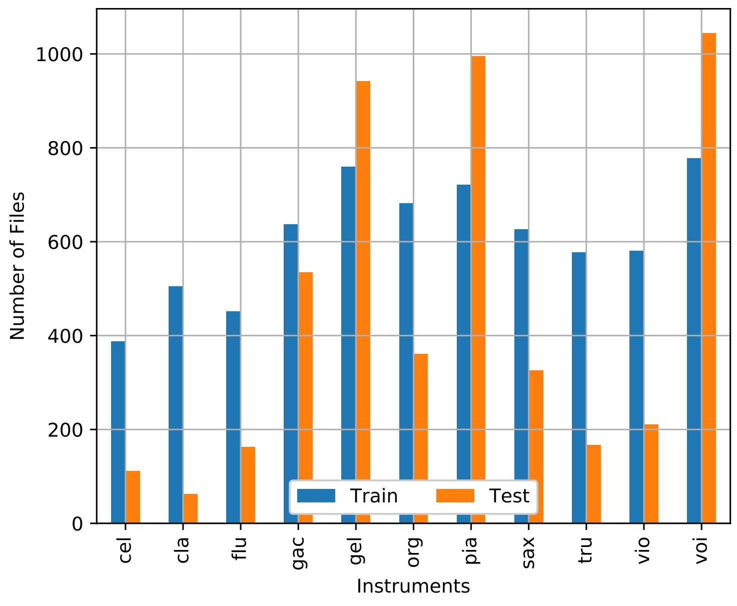


Figure 5.1: Distribution of audio files in IRMAS data set.

5.1.2 Monotimbral Data set

The Monotimbral data set has been created at the Fraunhofer IDMT for different timbre research tasks. Through the use of a script, the audio files were collected from YouTube with bit depth of 16 bits per sample and sampling frequency of 44.1 KHz. The data set contains 412 stereo audio files with variable duration from 10 to 120 seconds. Additionally, the audio that has been collected contains only monotimbral audio, which means that the annotated instrument is the only one that appears in the audio mixture (single-labeled). This leads to lower complexity in comparison with the IRMAS and Jazz Solo data sets (described in the following section), which are polyphonic and multitimbral. It is important to mention that this data set is weakly labeled, therefore instrument labels are given only on the audio clip level.

The 15 instruments that are annotated and their abbreviation in this data set are: acoustic guitar (acg), clarinet (cla), double bass (dob), electric guitar clean (egc), electric guitar distorted (egd), electric piano (epi), flute (flu), hammond organ (hor), piano (pia), saxophone (sax), singing voice female (svf), singing voice male (svm), synthesizer (syn), trumpet (tru), and violin (vio). Since it had no previous separation into training and testing data sets as the IRMAS case, it has been distributed equally based on the number of files for training and testing data sets, as seen in Figure 5.2.

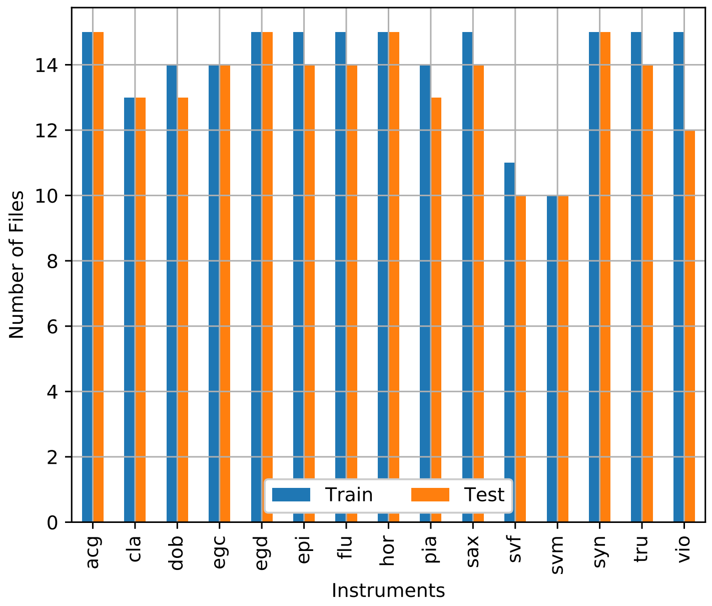


Figure 5.2: Distribution of audio files in Monotimbral data set.

5.1.3 Jazz Solo Instrument Data set

As a particular use case to classify popular brass and woodwind instruments in jazz solos, the Jazz Solo data set was collected from random solos of the Weimar Jazz Database and completed to have at least 25 solos for each instrument [41]. The objective of the Weimar Jazz Database is to have precise annotations on the pitches and durations (onset and offset times), chord changes, form sections, and phrase boundaries from different jazz solos. The data set contains 185 audio files collected from YouTube with bit depth of 16 bits per sample and sampling frequency of 44.1 KHz. Since the solos were extracted from real recordings, the material is polyphonic and polytimbral.

The 6 instruments that are annotated and their abbreviation in this data set are: alto saxophone (as), clarinet (cl), soprano saxophone (ss), trombone (tb), trumpet (tp), and tenor saxophone (ts). While the number of instruments is smaller compared to the IRMAS and Monotimbral data sets (as seen in Figure 5.3), the samples have been chosen to maximize diversity of performing artists. Samples from each class have been randomly selected to have the same total duration, achieving equal distribution of class samples (see Table 5.1).

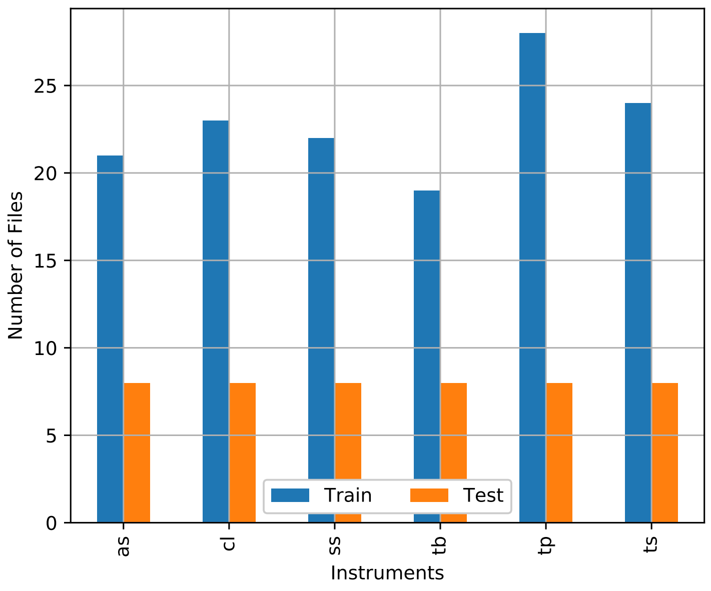


Figure 5.3: Distribution of audio files in Jazz data set.

5.1.4 IRMAS Wind Data set

This data set is a subset of the original IRMAS data set containing only brass and woodwind instruments. This reduces the original amount of samples to the following four instruments: clarinet (cla), flute (flu), saxophone (sax), and trumpet (tru). The amount of samples and durations from the original IRMAS data set were maintained.

This means that the data distributions of the audio files from Figure 5.1 are the same, while only considering these instruments. Given that this data set was only used for the Jazz Solo use case, it will be further discussed in section 5.9.

5.1.5 Summary and data splits

Table 5.1 summarizes the information about number of audio files and duration across data sets and labels. It is important to mention that the Monotimbral data set introduces new non-predominant instrument classes (e.g., double bass, synthesizer, electric piano). Additionally, the Jazz Solo data set contains sub-classes from the original data set (e.g, soprano, alto, and tenor saxophone).

Instrument		IRMAS		MONO		JAZZ	
Class	Subclass	num.	hr.	num.	hr.	num.	hr.
Cello		499	0.87				
Clarinet		567	0.71	26	0.32	31	0.53
Flute		614	1.17	29	0.42		
Acoustic Guitar		1172	3.08	30	0.38		
Electric Guitar		1702	5.00				
	Clean			28	0.43		
	Distorted			30	0.34		
Organ		1043	2.25				
Hammond Organ				30	0.44		
Piano		1716	5.40	27	0.38		
Electric Piano				29	0.31		
Saxophone		952	2.16	29	0.34		
	Soprano					30	0.53
	Alto					29	0.53
	Tenor					32	0.53
Trombone						27	0.53
Trumpet		744	1.29	29	0.35	36	0.53
Violin		791	1.56	27	0.47		
Voice		1822	5.38				
	Female			21	0.26		
	Male			20	0.26		
Double Bass				27	0.28		
Synthesizer				30	0.77		
TOTAL		11622	28.87	412	5.75	185	3.18

Table 5.1: Global distribution of used data sets, including number of audio files (num.) and duration in hours (hr.) of annotated audio per instrument label.

This table includes the number of files and total duration in seconds of each class and data set, including both training and testing data sets. Given that the testing data set from IRMAS is multi-labeled, a file that is annotated with both voice and piano results in adding the duration of this file to both labels in the table. For this reason, instruments like acoustic guitar, electric guitar, piano, and voice result in higher total duration compared to other instruments.

To avoid overfitting, the training data set is split into a training and a validation data set, as mentioned in section 2.4. The *training data set* is used by the model to learn the weights and biases of the neurons, and the *validation data set* to estimate the generalization error during training, and update the hyperparameters accordingly. Following the reference model, 15% of the training data was randomly selected and used for validation. To implement the identification thresholding at the post-processing stage, the testing data set is split into a development and a pure test data set, as seen in section 4.1. The *development test data set* is used to calculate performance metrics while varying the threshold from 0.0 to 1.0, in order to obtain the optimum identification threshold. The *pure test data set* is used to obtain the final performance metrics using the optimum threshold calculated in the previous step. Following the reference model, 50% of the original test data set was used as development data set and the remaining 50% was used for to calculate the final performance metrics. Table 5.2 shows the resulting data split of mel-spectrograms in each data set (with a dimensionality of 128 mel-bands $\times$ 43 time frames). Following the reference model, the testing mel-spectrograms were extracted with a 50% overlap as seen in Figure 4.3.

	Training Data Set (85/15)		Testing Data Set (50/50)	
	Train	Validation	Development	Pure
IRMAS Data set	17094	3021	48064	48055
Monotimbral Data set	8676	1539	10620	10610
Jazz Solo Data set	7206	1275	1678	1271
IRMAS Wind Data set	5486	970	10447	10446

Table 5.2: Number of mel-spectrograms for each data set split into train, validation, development, and pure data sets.

5.2 Performance metrics

To compensate for the effect of random initialization, each model was trained three times and the performance metrics were calculated and averaged over all model iterations. An initial cue of the system performance is the training history plot. This plot shows the minimization of the loss function, the maximization of accuracy using training and validation data sets, and the amount of epochs required for early stopping. Figure 5.4 shows an example of model loss and accuracy through all training epochs. The parameters `loss` and `acc` refer to the loss and accuracy of the model when using training data, while `val_loss` and `val_acc` are calculated using the validation data set. The solid line shows the mean values across training iterations, while the shaded area represents their standard deviation. This representation of mean and standard deviation of training iterations is maintained throughout the remain of this thesis. As mentioned in section 4.1, early stopping is implemented such that the model stops training after 10 epochs with no decrease in `val_loss`. Although the validation accuracy measures the prediction performance on the validation data, it is necessary to evaluate the recognition performance of the model on unseen data. For this reason, recognition metrics and confusion matrices are extracted on the testing data set.

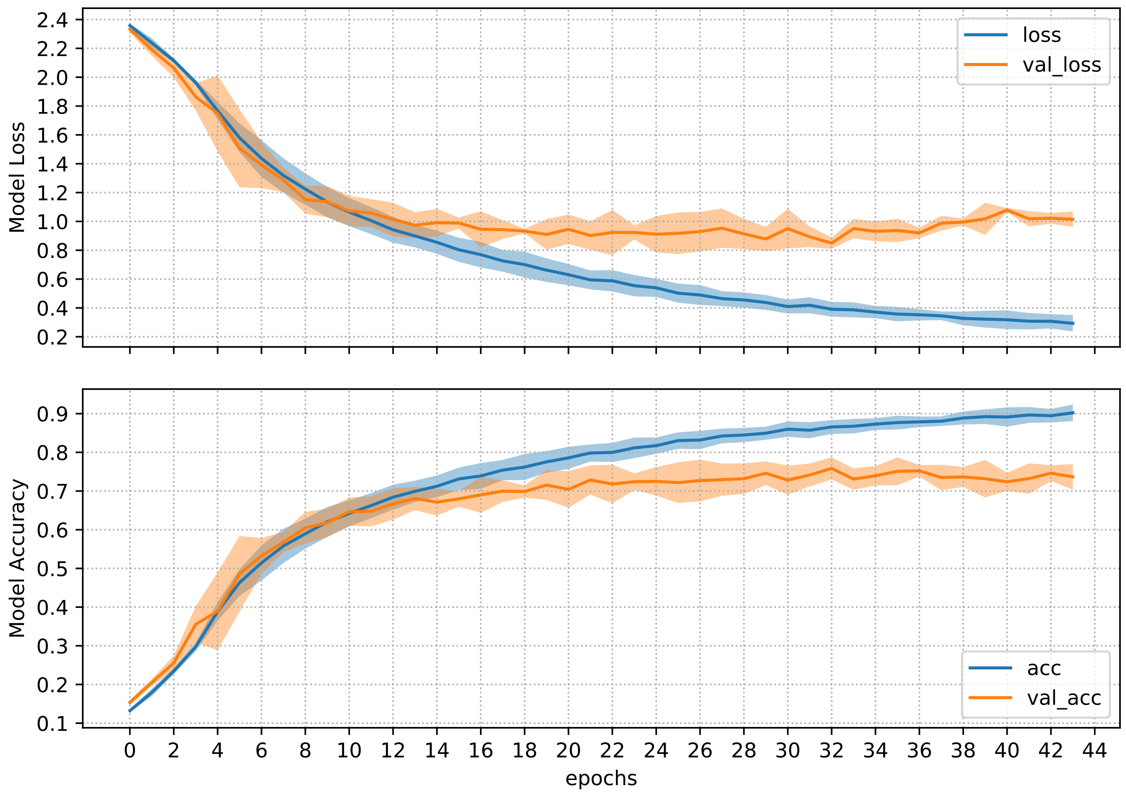


Figure 5.4: Example of model loss and accuracy during training, with calculated `loss` and `acc` from the training data set and `val_loss` and `val_acc` from validation data set.

5.2.1 Recognition metrics

Following evaluation conventions in other instrument recognition systems [3, 6, 15], precision, recall and f-scores are calculated for both micro and macro averages. Micro averaging gives more weight to instrument classes that have a higher number of appearances in the data distribution. For each label l in the total instrument classes L, tp_l corresponds to the true positives, fp_l to the false positives, and fn_l to the false negatives. The micro-averaged performance metrics are defined as:

$$\begin{aligned} P_{micro} &= \frac{\sum_{l=1}^{L} tp_l}{\sum_{l=1}^{L} tp_l + fp_l} \\ R_{micro} &= \frac{\sum_{l=1}^{L} tp_l}{\sum_{l=1}^{L} tp_l + fn_l} \\ F_{micro} &= \frac{2P_{micro}R_{micro}}{P_{micro} + R_{micro}} \end{aligned}$$

Conversely, macro averaging is calculated for each label, representing the overall performance of all classes. P_l and R_l are precision and recall calculated per label l as:

$$\begin{aligned} P_l &= \frac{tp_l}{tp_l + fp_l} \\ R_l &= \frac{tp_l}{tp_l + fn_l} \end{aligned}$$

The macro-averaged performance metrics are defined as:

$$\begin{aligned} P_{macro} &= \frac{1}{|L|}\sum_{l=1}^{L} P_l \\ R_{macro} &= \frac{1}{|L|}\sum_{l=1}^{L} R_l \\ F_{macro} &= \frac{2P_{macro}R_{macro}}{P_{macro} + R_{macro}} \end{aligned}$$

As mentioned in section 5.1, the performance metrics are measured in two stages of the evaluation phase: once to obtain the optimum global threshold with the development test data set, and then to finally obtain the final metrics using this threshold on the pure test data set. In the first step, the identification threshold θ is modified from 0.0 to 1.0 in steps of 0.05, while evaluating f-score using both averaging methods and both aggregation strategies (see Figure 5.5). Given the training iterations, the optimum threshold is estimated as the maximum of the mean f-score across all training iterations. In the second step, this threshold is used for a final evaluation using the pure test data set.

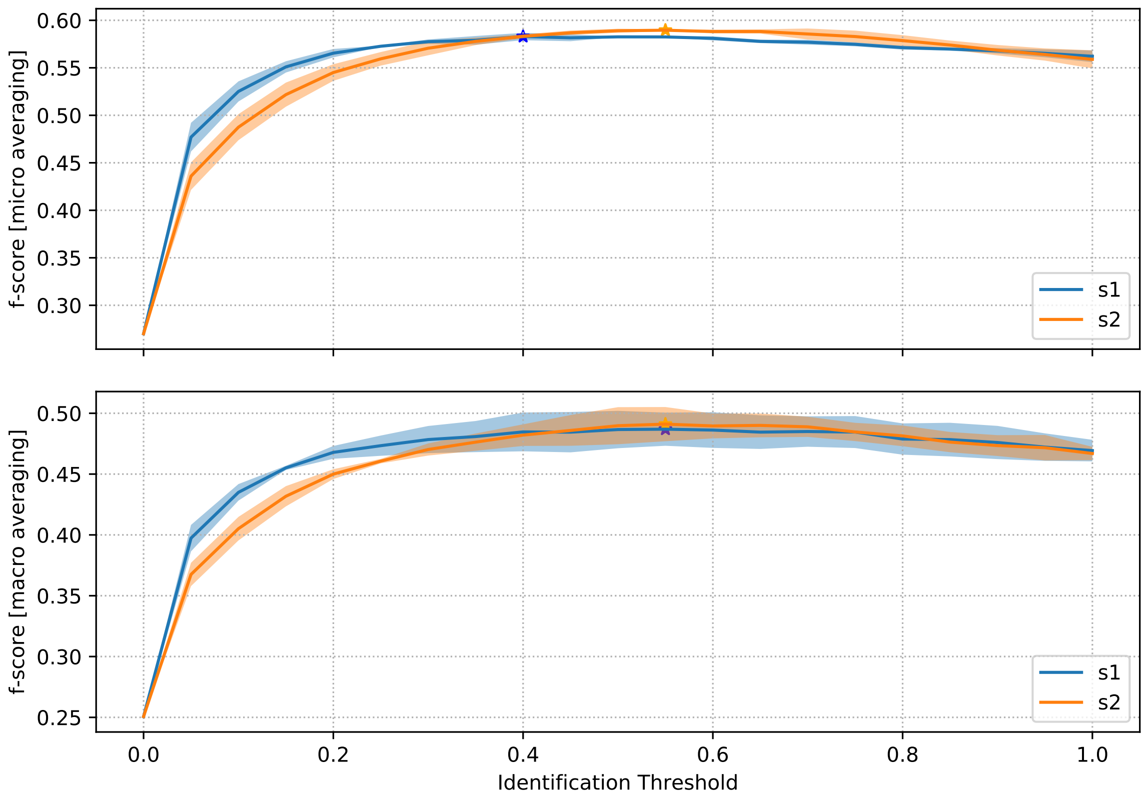


Figure 5.5: Example of identification of optimal threshold with development test data set using micro (top), macro (bottom) averaging, and both aggregation strategies (s1 and s2).

5.2.2 Confusion matrices

Confusion matrices allow to evaluate the performance of a network by counting the amount of times a given label is predicted correctly and incorrectly [24]. According to Han et al. [3], they represent a more ”pure” performance of the network, given that they only predict one predominant instrument from each sample. Two different confusion matrices are evaluated depending on the nature of the data sets:

- *IRMAS data set*: given that the testing data set has multiple simultaneous labels, the confusion matrices have been calculated by using the training data set. To achieve this, the training set was split into halves: the first for training the model again, and the second to only evaluate confusion matrices.

- *Monotimbral and Jazz Solo data sets*: given that both of these data sets are single-labeled for both the training and testing data sets, it is possible to obtain confusion matrices on the pure testing data set (see Table 5.2).

5.3 Reproduction of reference system evaluation

Using the optimum identification threshold θ, Table 5.3 shows the achieved results from the implemented system using the pure test data set. The results reported by [3] are labeled *Baseline* - *IRMAS*, and were obtained using the ReLU activation function and *S2* aggregation. These results were obtained from an ensemble model, which combines the outcome of different predictors. The implemented system trained with the IRMAS data set is labeled *Reproduction* - *IRMAS* and the one trained with the Monotimbral data set is labeled *Experiment* - *MONO*. Experiments were carried out by modifying the activation function of the hidden layers and the aggregation strategy on the post-processing stage. With respect to the activation function, the implemented models were trained using ReLU and LReLU activations (with leakage parameter $\alpha = 0.33$), as in [3].

The implemented system that was trained with the IRMAS data set shows similar performance metrics using *S2* aggregation strategy. These results show exactly the same optimum identification threshold using ReLU activation, as reported by [3]. The reduced performance of the implemented system is possibly due to the effect of the ensemble model, which was not implemented in this research.

In the case of the Monotimbral data set, best performance is achieved by using the LReLU activation function and the *S1* aggregation strategy. *S2* was implemented to recognize sporadic activations in the case of multi-labeled testing data [3]. Using *S1* results in better performance in the case of the Monotimbral data set, possibly due to the nature of the data set: a predominant instrument is recognized with respect to its relative strength compared to the most active instrument (see section 4.1). In the case of this data set, there are no other ”competing” instruments in the audio mixture. This motivates to continue evaluating the effect of the aggregation strategy, which is further discussed in the proposed use case in section 5.7. Moreover, the overall performance metrics obtained with the Monotimbral data set improve approximately between 5-10% in all scores, with respect to the model trained with the IRMAS data set. This indicates that using monotimbral input to the system can effectively improve the overall performance, motivating the use of source separation algorithms as a pre-processing stage. The optimum identification threshold obtained by training on the

Monotimbral data set results in a higher value (0.7 - 0.8). As mentioned in section 4.1, a higher threshold results in higher precision and lower recall scores. Since the model has been trained with a more "pure" timbral representation of the instruments and is less likely to wrongly predict outputs, a higher threshold is more likely to improve the overall performance metrics. The effect of thresholding is further analyzed with the proposed post-processing experiments in section 5.7.

Model	Data set	Activation Function	Agg.	Micro Averaging P	R	F	Macro Averaging P	R	F	Opt. θ
Baseline [3]	IRMAS	**ReLU**	**S2**	**0.657**	**0.603**	**0.629**	**0.540**	**0.547**	**0.517**	**0.55**
Reproduction	IRMAS	ReLU	S1	0.591	0.548	0.568	0.530	0.477	0.471	0.40
	IRMAS	**ReLU**	**S2**	**0.609**	**0.544**	**0.574**	**0.501**	**0.507**	**0.475**	**0.55**
Experiment	MONO	**LReLU**	**S1**	**0.645**	**0.678**	**0.661**	**0.685**	**0.681**	**0.657**	**0.8**
	MONO	LReLU	S2	0.619	0.695	0.655	0.657	0.690	0.649	0.7

Table 5.3: Reproduction of results from the baseline system using IRMAS and Monotimbral data sets.

Class-wise performance metrics were extracted to compare the baseline results with the ones achieved by using the best strategies for each data set (see Figure 5.6). Concerning the implemented system trained with the IRMAS data set, the overall f-scores for each instrument are consistent with the ones reported by the authors. Low performance of cello and clarinet could be due to the low amount of testing data for these instruments (see Figure 5.1). Moreover, the low performance metrics obtained on the clarinet could also be justified by the wrongful annotation of training labels in the IRMAS data set. Concretely, several training examples labeled as clarinet were actually saxophone or oboe. This was reported to the creators of the data set in an effort to continuously clean the data set. On the other hand, even though flute also has a low amount of testing data, its sine-like signal is distinctive from all other instruments in the data set. This is likely to result in better classification performance. Additionally, both the reference and the implemented system report best performance for voice, possibly due to its unique spectral characteristics, which contains more inharmonicities and natural vibrato than other instruments [3]. In contrast, the evaluation of the Monotimbral data set results in overall better class-wise performance, likely due to the complexity of the data set. Given its monotimbral content, the classification task (and more concretely, the filters) can learn from a more "pure" representation of the timbre of each instrument. Synthesizer and violin are the only instruments that result in f-scores that are lower than 50%, which is further discussed when analyzing the obtained confusion matrix.

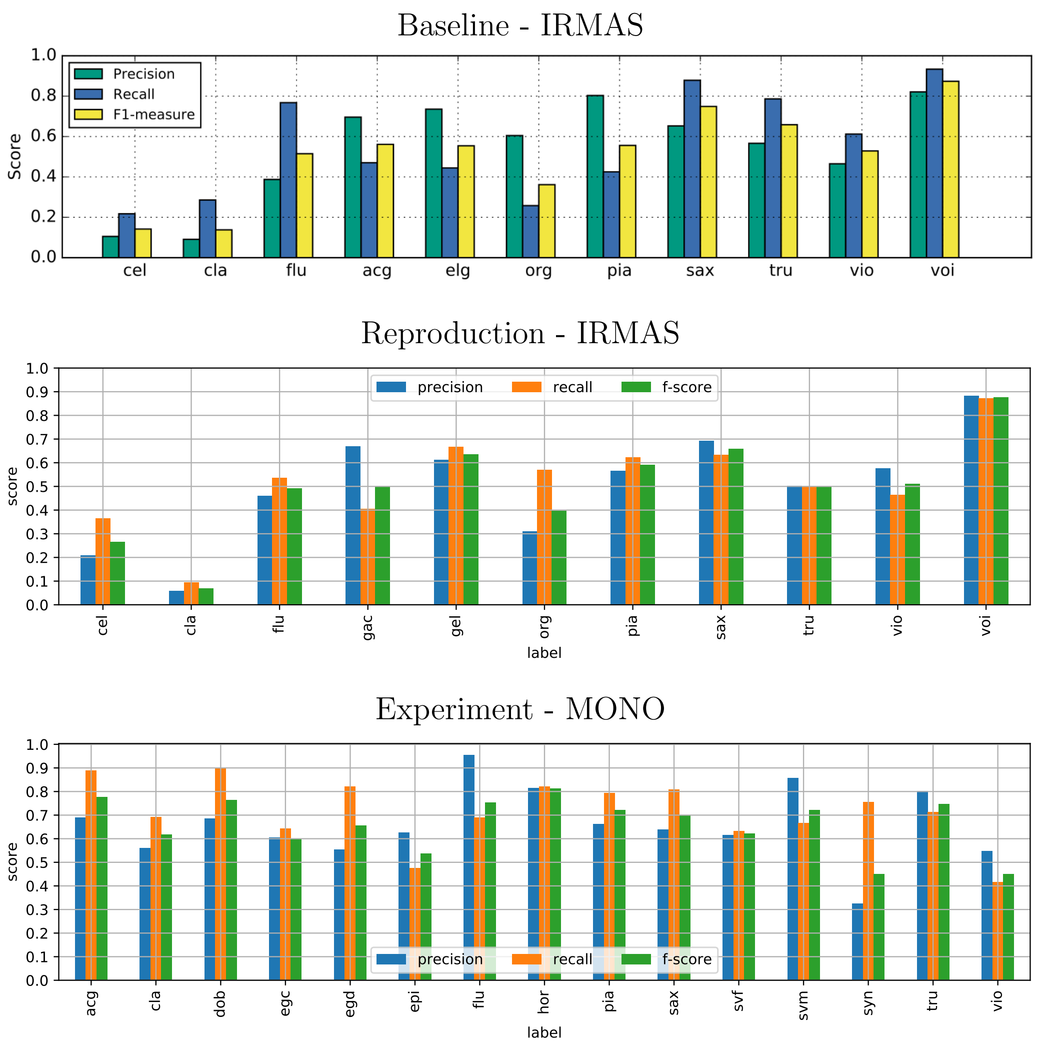
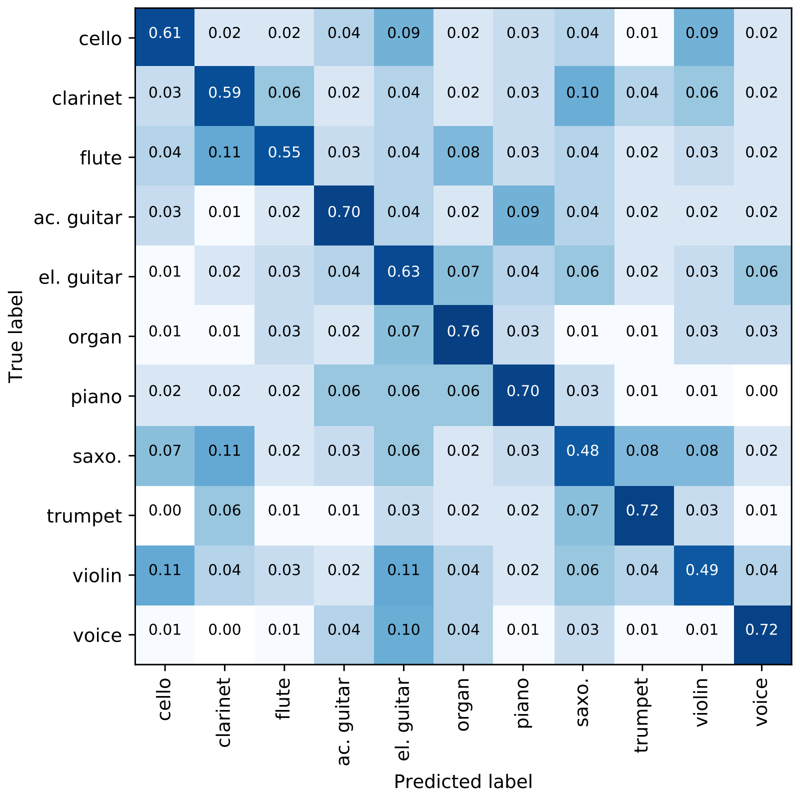


Figure 5.6: Class-wise performance metrics of baseline system [3] (top), achieved class-wise performance metrics using the IRMAS data set with aggregation strategy S2 (middle), and the Monotimbral data set with aggregation strategy S1 (bottom).

After evaluating the confusion matrices the authors report an overall mean accuracy of 63.18%, while the implemented system obtained 68.13%, as shown in Figure 5.7. This difference is possibly due to early stopping: the baseline system was trained to stop after 5 epochs with no decrease in validation loss, while the implemented system had a patience of 10 epochs. The reason to implement a longer patience is that longer training could potentially result in a lower minimum in the loss function and higher accuracy, given the backpropagation algorithm.



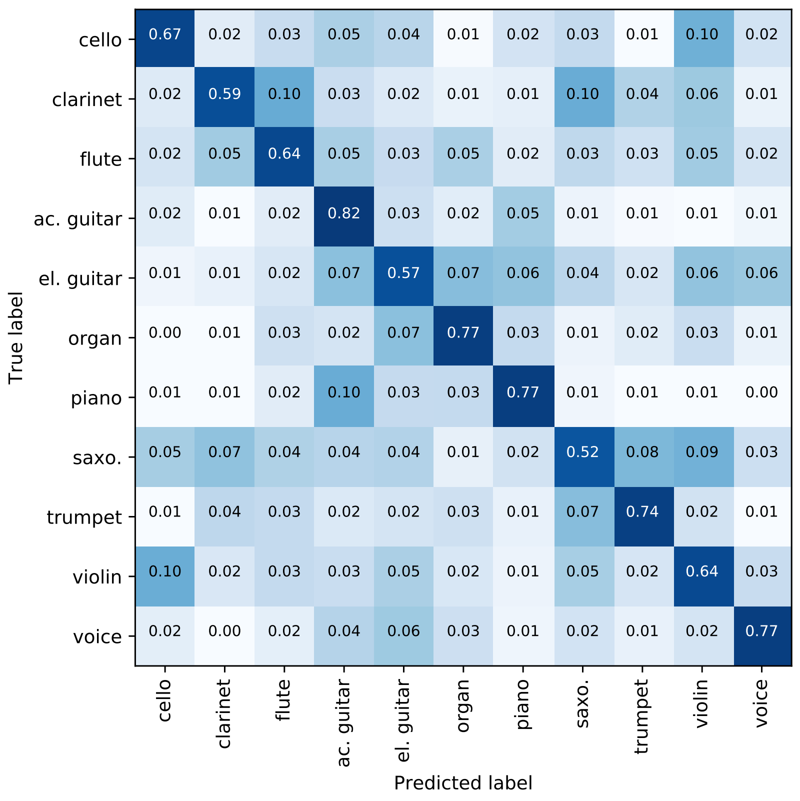
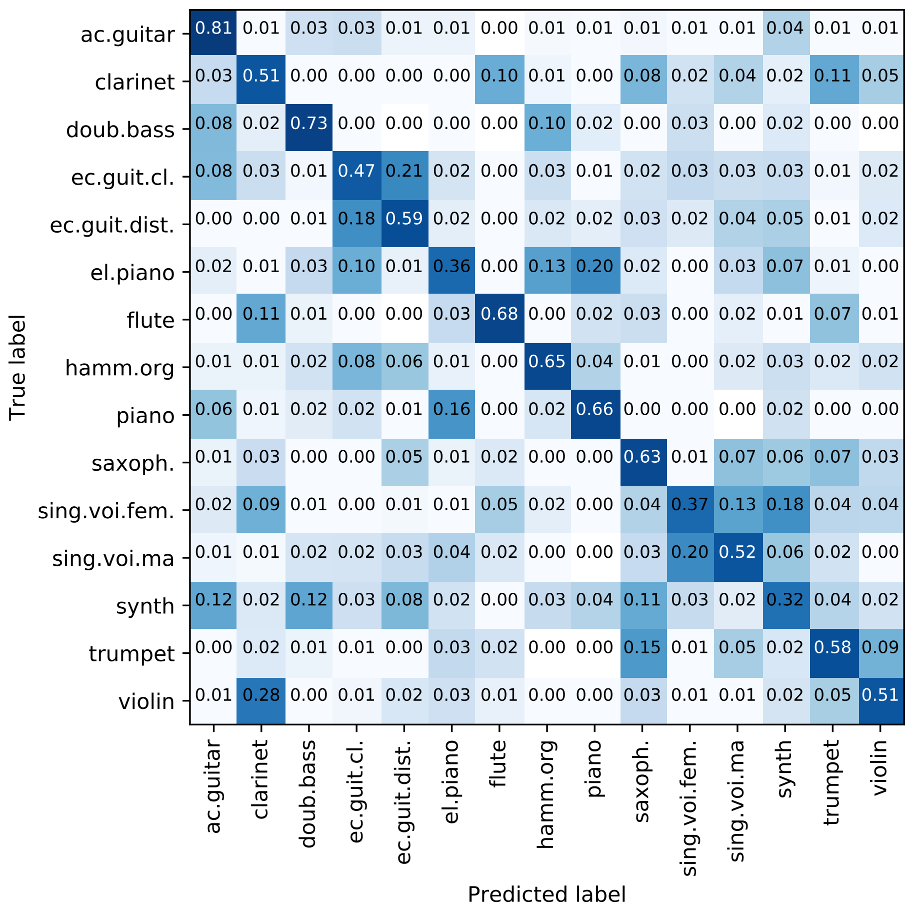


Figure 5.7: Comparison of confusion matrices for IRMAS data set, as reported by the baseline system - mean accuracy 63.18% (top), and obtained with the implemented system - mean accuracy 68.16% (bottom).

Both the baseline and the implemented system reflect common confusions such as violin - cello, saxophone - trumpet, acoustic guitar - electric guitar, and flute - clarinet. Likewise, both systems present low accuracy in the recognition of saxophone and clarinet. The implemented system obtained improved accuracy of flute, acoustic guitar, and violin. This improvement is possibly due to the implementation of the train-validation split. In the case of the baseline system, the train-validation data split was completely random. In the case of the implemented system, a balanced class-wise train-validation split was implemented to take into account the uneven distribution of the IRMAS data set. Class-wise train-validation split is introduced in section 5.4.



	ac.guitar	clarinet	doub.bass	ec.guit.cl.	ec.guit.dist.	el.piano	flute	hamm.org	piano	saxoph.	sing.voi.fem.	sing.voi.ma	synth	trumpet	violin
ac.guitar	0.81	0.01	0.03	0.03	0.01	0.01	0.00	0.01	0.01	0.01	0.01	0.01	0.04	0.01	0.01
clarinet	0.03	0.51	0.00	0.00	0.00	0.00	0.10	0.01	0.00	0.08	0.02	0.04	0.02	0.11	0.05
doub.bass	0.08	0.02	0.73	0.00	0.00	0.00	0.00	0.10	0.02	0.00	0.03	0.00	0.02	0.00	0.00
ec.guit.cl.	0.08	0.03	0.01	0.47	0.21	0.02	0.00	0.03	0.01	0.02	0.03	0.03	0.03	0.01	0.02
ec.guit.dist.	0.00	0.00	0.01	0.18	0.59	0.02	0.00	0.02	0.02	0.03	0.02	0.04	0.05	0.01	0.02
el.piano	0.02	0.01	0.03	0.10	0.01	0.36	0.00	0.13	0.20	0.02	0.00	0.03	0.07	0.01	0.00
flute	0.00	0.11	0.01	0.00	0.00	0.03	0.68	0.00	0.02	0.03	0.00	0.02	0.01	0.07	0.01
hamm.org	0.01	0.01	0.02	0.08	0.06	0.01	0.00	0.65	0.04	0.01	0.00	0.02	0.03	0.02	0.02
piano	0.06	0.01	0.02	0.02	0.01	0.16	0.00	0.02	0.66	0.00	0.00	0.00	0.02	0.00	0.00
saxoph.	0.01	0.03	0.00	0.00	0.05	0.01	0.02	0.00	0.00	0.63	0.01	0.07	0.06	0.07	0.03
sing.voi.fem.	0.02	0.09	0.01	0.00	0.01	0.01	0.05	0.02	0.00	0.04	0.37	0.13	0.18	0.04	0.04
sing.voi.ma	0.01	0.01	0.02	0.02	0.03	0.04	0.02	0.00	0.00	0.03	0.20	0.52	0.06	0.02	0.00
synth	0.12	0.02	0.12	0.03	0.08	0.02	0.00	0.03	0.04	0.11	0.03	0.02	0.32	0.04	0.02
trumpet	0.00	0.02	0.01	0.01	0.00	0.03	0.02	0.00	0.00	0.15	0.01	0.05	0.02	0.58	0.09
violin	0.01	0.28	0.00	0.01	0.02	0.03	0.01	0.00	0.00	0.03	0.01	0.01	0.02	0.05	0.51

Figure 5.8: Confusion matrix obtained with the implemented system on the Monotimbral data set - mean accuracy 55.97%.

Figure 5.8 shows the confusion matrix obtained on the implemented system trained with the Monotimbral data set. The instrument selection of this data set results in the confusion of instruments with higher similarity: clean - distorted electric guitar, female - male singing voice, and electric piano - hammond organ - piano - synthesizer. The confusion of these instruments result in an overall mean accuracy of 55.97% for this data set. In the case of the synthesizer, it is important to mention that both training and testing data sets include audio files where it emulates the sound of other instruments. This possibly leads to its confusion with other instruments, such as female singing voice, acoustic guitar, double bass, and saxophone. The confusion of the synthesizer is further analyzed in the following section. Finally, the obtained accuracy on instruments in common with the IRMAS data set was similar or improved marginally (namely, acoustic guitar, flute, piano, saxophone, trumpet, and violin).

5.4 Proposed Experiments: Pre-processing stage

Three experiments have been carried out in the pre-processing stage to improve the classification results: balanced class-wise train-validation split, phase-based harmonic/percussive separation, and pitch-informed solo/accompaniment separation. Following research by [20] on automatic source separation algorithms for instrument recognition, independent models were trained for each separated audio stream.

5.4.1 Balanced class-wise train-validation split

In contrast to the baseline model, that randomly selected mel-spectrograms to perform the train-validation split, this implementation considers balancing the separation of this data. The implemented data split takes into account the amount of extracted mel-spectrograms of each class. By this means, every class on each data set was randomly split into 85% for the training data set and 15% for the validation data set. This can account for the improvement of the accuracy of flute, acoustic guitar, and violin recognition, when comparing the baseline and reproduced results on the IRMAS data set.

Additionally, an experiment has been conducted by calculating the data distribution and adding it to the `class_weight` mapping from Keras [21]. This can be used to inform the model of under-represented classes, giving them higher weight during training. This experiment did not result in significant classification improvements.

5.4.2 Phase-based Harmonic/Percussive Separation

The IRMAS and Monotimbral data sets have been pre-processed with the harmonic/percussive separation algorithm by Cano [1], resulting in diverse outcomes. The split that has been implemented in section 5.1 is maintained, in such a way that the samples in both training and testing data sets remain exactly the same. Using the extracted harmonic component, new models were trained for the IRMAS and Monotimbral data sets. In the case of the IRMAS data set, the harmonic component contains the predominant instrument and results in a marginal improvement (1-3%) of the overall recognition performance in both averaging methods and both aggregation strategies, as shown in Figure 5.9. This figure shows the difference between all the performance metrics (precision, recall, and f-score) obtained by the baseline system and the modified system, considering both averaging methods and both aggregation strategies. This representation of score improvement is maintained throughout the remain of this thesis.

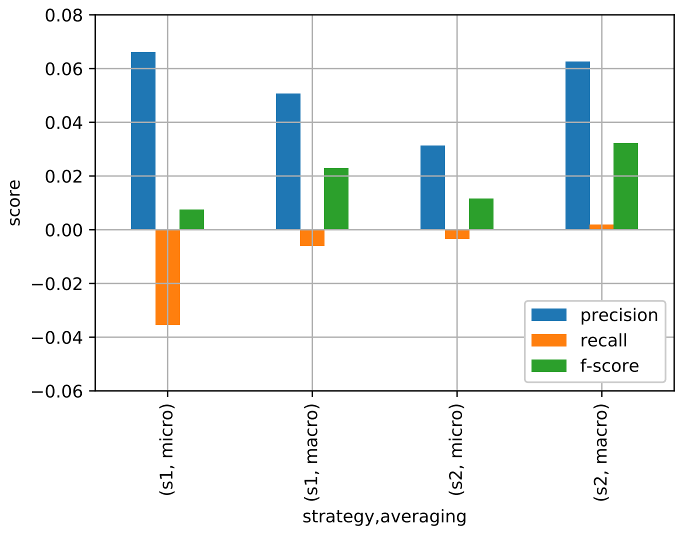
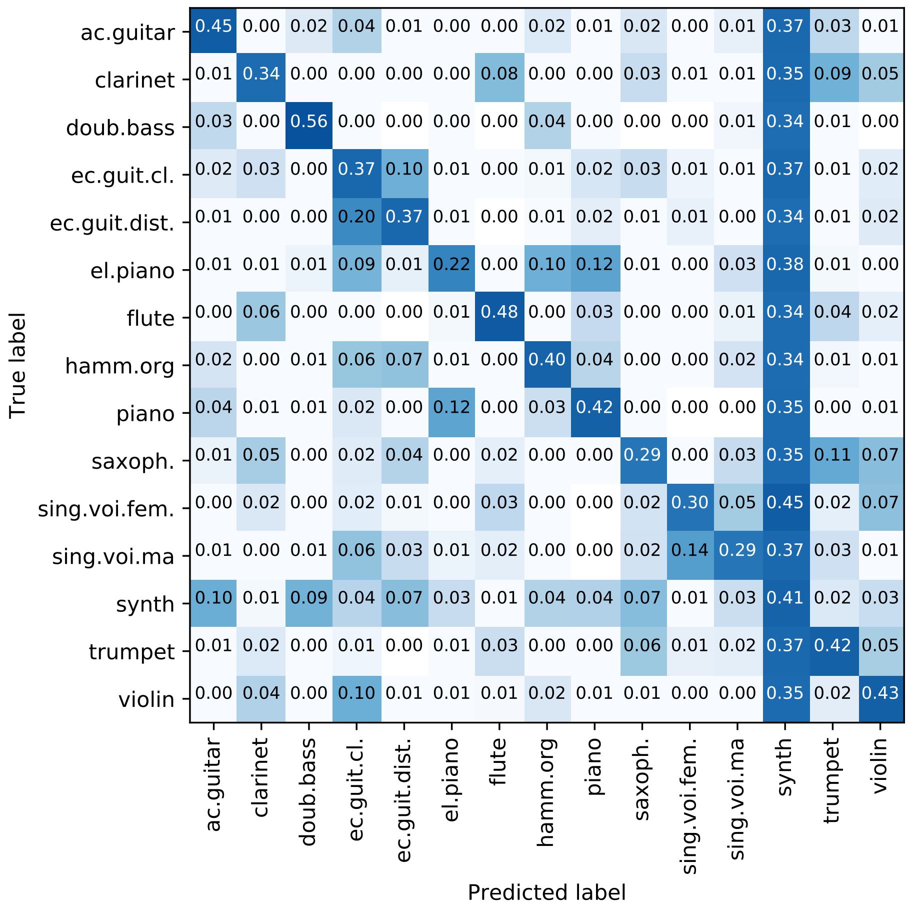


Figure 5.9: Score improvement of using harmonic/percussive separation on the IRMAS data set and training the baseline model on the harmonic component.

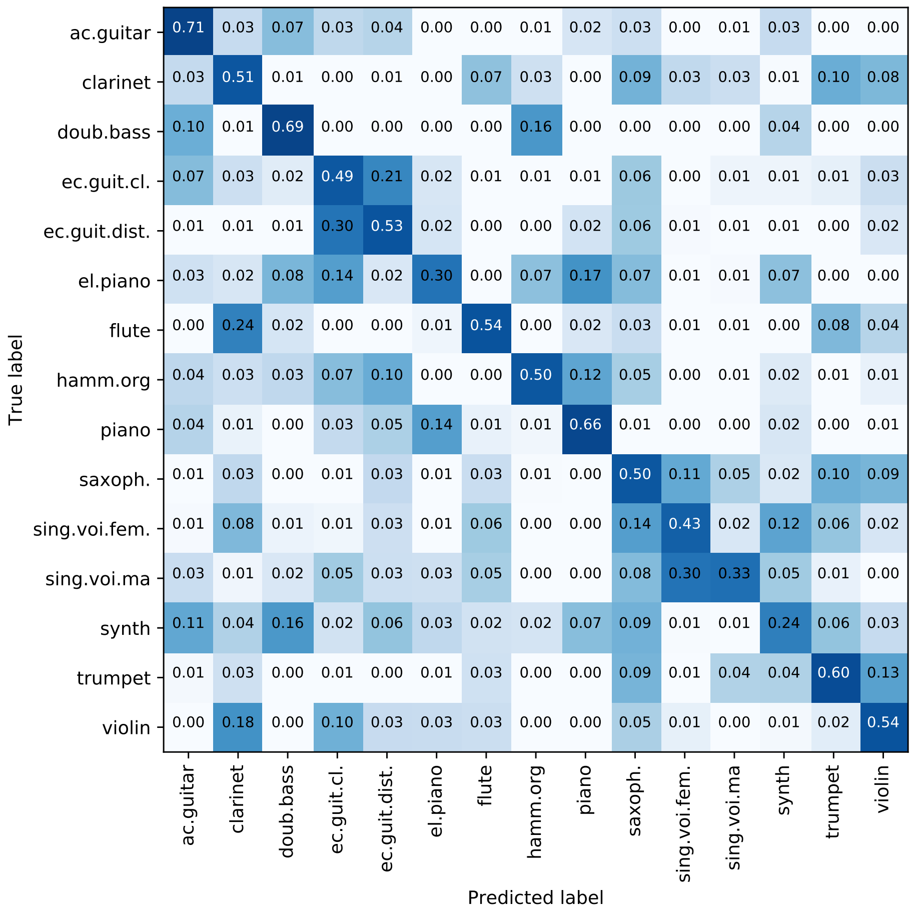
In the case of the Monotimbral data set, overall performance decreased significantly when using the harmonic/percussive separation algorithm (from 55.97% to 38.33% mean accuracy, and decrease of 1-2% in all f-scores). Given the nature of the data set, using the harmonic component results in removing a significant portion of the attack of the extracted harmonic component. This is consistent with Figure 5.10, where the synthesizer is confused with all instruments. As mentioned in section 5.3, the synthesizer examples of the testing data set are very diverse. Certain audio files emulate the sound of a flute or voice, while others have a strong ”electronic” quality. This leads to a discussion regarding human and automatic instrument recognition, which will be further explored in the remain of this section. Concretely, approximately one third of every prediction was wrongly labeled as a synthesizer, reducing the overall performance of the system. The use of the harmonic component on monotimbral data as input to the instrument recognition system was performed to evaluate the effect of this pre-processing algorithm, since the "real-world" scenario could contain both multi- and monotimbral material.



	ac.guitar	clarinet	doub.bass	ec.guit.cl.	ec.guit.dist.	el.piano	flute	hamm.org	piano	saxoph.	sing.voi.fem.	sing.voi.ma	synth	trumpet	violin
ac.guitar	0.45	0.00	0.02	0.04	0.01	0.00	0.00	0.02	0.01	0.02	0.00	0.01	0.37	0.03	0.01
clarinet	0.01	0.34	0.00	0.00	0.00	0.00	0.08	0.00	0.00	0.03	0.01	0.01	0.35	0.09	0.05
doub.bass	0.03	0.00	0.56	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.01	0.34	0.01	0.00
ec.guit.cl.	0.02	0.03	0.00	0.37	0.10	0.01	0.00	0.01	0.02	0.03	0.01	0.01	0.37	0.01	0.02
ec.guit.dist.	0.01	0.00	0.00	0.20	0.37	0.01	0.00	0.01	0.02	0.01	0.01	0.00	0.34	0.01	0.02
el.piano	0.01	0.01	0.01	0.09	0.01	0.22	0.00	0.10	0.12	0.01	0.00	0.03	0.38	0.01	0.00
flute	0.00	0.06	0.00	0.00	0.00	0.01	0.48	0.00	0.03	0.00	0.00	0.01	0.34	0.04	0.02
hamm.org	0.02	0.00	0.01	0.06	0.07	0.01	0.00	0.40	0.04	0.00	0.00	0.02	0.34	0.01	0.01
piano	0.04	0.01	0.01	0.02	0.00	0.12	0.00	0.03	0.42	0.00	0.00	0.00	0.35	0.00	0.01
saxoph.	0.01	0.05	0.00	0.02	0.04	0.00	0.02	0.00	0.00	0.29	0.00	0.03	0.35	0.11	0.07
sing.voi.fem.	0.00	0.02	0.00	0.02	0.01	0.00	0.03	0.00	0.00	0.02	0.30	0.05	0.45	0.02	0.07
sing.voi.ma	0.01	0.00	0.01	0.06	0.03	0.01	0.02	0.00	0.00	0.02	0.14	0.29	0.37	0.03	0.01
synth	0.10	0.01	0.09	0.04	0.07	0.03	0.01	0.04	0.04	0.07	0.01	0.03	0.41	0.02	0.03
trumpet	0.01	0.02	0.00	0.01	0.00	0.01	0.03	0.00	0.00	0.06	0.01	0.02	0.37	0.42	0.05
violin	0.00	0.04	0.00	0.10	0.01	0.01	0.01	0.02	0.01	0.01	0.00	0.00	0.35	0.02	0.43

Figure 5.10: Confusion matrix for Monotimbral data set using the harmonic component - mean accuracy 38.33%.

Seminal research on human timbre recognition by Berger [42] states that both attack and release sections of the amplitude envelope give the listener cues to successfully recognize a set of ten wind instruments. As a comparison with automatic instrument recognition, a model was trained using only the percussive component of the Monotimbral data set. The extracted component contains mainly the attack of the audio mixtures. Confirming the authors results, using the percussive component results only in a decrease in accuracy of 5.56% compared to the model trained with the Monotimbral data set, as shown in Figure 5.11. Even though the performance metrics have marginally decreased, the percussive component of the Monotimbral data set can still be used as a cue to the system, motivating experiments in section 5.6. Given the improvements on the IRMAS data set, Chapter 6 proposes an outlook regarding the use of harmonic/percussive separation in both multi- and monotimbral audio.



	ac.guitar	clarinet	doub.bass	ec.guit.cl.	ec.guit.dist.	el.piano	flute	hamm.org	piano	saxoph.	sing.voi.fem.	sing.voi.ma	synth	trumpet	violin
ac.guitar	0.71	0.03	0.07	0.03	0.04	0.00	0.00	0.01	0.02	0.03	0.00	0.01	0.03	0.00	0.00
clarinet	0.03	0.51	0.01	0.00	0.01	0.00	0.07	0.03	0.00	0.09	0.03	0.03	0.01	0.10	0.08
doub.bass	0.10	0.01	0.69	0.00	0.00	0.00	0.00	0.16	0.00	0.00	0.00	0.00	0.04	0.00	0.00
ec.guit.cl.	0.07	0.03	0.02	0.49	0.21	0.02	0.01	0.01	0.01	0.06	0.00	0.01	0.01	0.01	0.03
ec.guit.dist.	0.01	0.01	0.01	0.30	0.53	0.02	0.00	0.00	0.02	0.06	0.01	0.01	0.01	0.00	0.02
el.piano	0.03	0.02	0.08	0.14	0.02	0.30	0.00	0.07	0.17	0.07	0.01	0.01	0.07	0.00	0.00
flute	0.00	0.24	0.02	0.00	0.00	0.01	0.54	0.00	0.02	0.03	0.01	0.01	0.00	0.08	0.04
hamm.org	0.04	0.03	0.03	0.07	0.10	0.00	0.00	0.50	0.12	0.05	0.00	0.01	0.02	0.01	0.01
piano	0.04	0.01	0.00	0.03	0.05	0.14	0.01	0.01	0.66	0.01	0.00	0.00	0.02	0.00	0.01
saxoph.	0.01	0.03	0.00	0.01	0.03	0.01	0.03	0.01	0.00	0.50	0.11	0.05	0.02	0.10	0.09
sing.voi.fem.	0.01	0.08	0.01	0.01	0.03	0.01	0.06	0.00	0.00	0.14	0.43	0.02	0.12	0.06	0.02
sing.voi.ma	0.03	0.01	0.02	0.05	0.03	0.03	0.05	0.00	0.00	0.08	0.30	0.33	0.05	0.01	0.00
synth	0.11	0.04	0.16	0.02	0.06	0.03	0.02	0.02	0.07	0.09	0.01	0.01	0.24	0.06	0.03
trumpet	0.01	0.03	0.00	0.01	0.00	0.01	0.03	0.00	0.00	0.09	0.01	0.04	0.04	0.60	0.13
violin	0.00	0.18	0.00	0.10	0.03	0.03	0.03	0.00	0.00	0.05	0.01	0.00	0.01	0.02	0.54

Figure 5.11: Confusion matrix for Monotimbral data set using the percussive component - mean accuracy 50.41%.

5.4.3 Pitch-Informed Solo/Accompaniment Separation

The solo/accompaniment source separation algorithm by Cano et al. [2] works particularly well on jazz music. For this reason, it was not used process the complete IRMAS or Monotimbral data sets. Training on the solo component from the Monotimbral data set would result in introducing artifacts to the audio. Consequently, a new model was trained using the IRMAS Wind data set described in section 5.1. Two models have been implemented: one trained with the original multitimbral data and one trained with the solo component obtained from pre-processing the IRMAS Wind data set with this algorithm. Figure 5.12 shows the improvement produced (roughly 3-4% improvement in all f-scores). These results motivate the creation of the use case of Jazz Solo recognition scenario, which is further discussed in section 5.9.

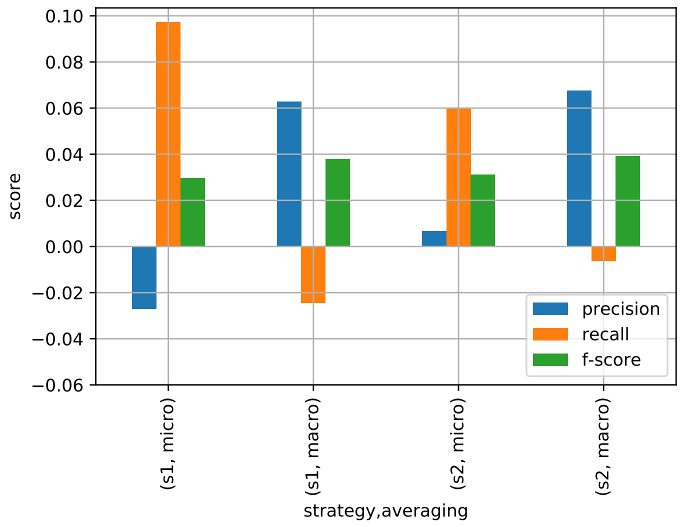


Figure 5.12: Score improvement of using solo/accompaniment separation on the IRMAS Wind data set and training the implemented baseline model on the solo component.

Although an improvement in performance metrics has been achieved, mean accuracy reports 74.26% for the IRMAS Wind data set and 69.64% for the solo separated audio. This results from the decreased recognition of flute after source separation, and wrongly classifying it as clarinet or saxophone. Reviewing the solo separated flute audio exposes that this can be due to three reasons. Firstly, the pitch detection algorithm is detecting notes from other non-predominant instruments from the mixture, making the solo/accompaniment algorithm extract other instruments instead of the flute. Secondly, certain train samples include piccolo flute, which results in artifacts when adding up to 20 harmonics to an original high pitch, as explained in section 4.2.2. Finally, the pitch detection algorithm can extract the pitch of other instruments in the mix, which have shorter attack.

5.5 Proposed Experiments: Feature processing stage

The feature processing stage includes two experiments that have an impact on the mel-spectrograms used as input to the instrument recognition system: improving the frequency resolution and using batch normalization.

5.5.1 Frequency resolution enhancement

Introducing a change in the window size of the STFT in the feature processing stage (see section 4.1), results in reducing the spectral leakage in the resulting mel-spectrogram. The original window size used by Han et al. [3] is 1024 time frames. By generating the training samples using an STFT with window size of 2048 and maintaining the hop size to preserve temporal dimensionality. The resulting mel-spectrograms have ”enhanced” frequency resolution (see Figure 5.13). Even though the frequency resolution is constant given the amount of mel-bands, this experiment results in reducing the spectral leakage between mel-bands.

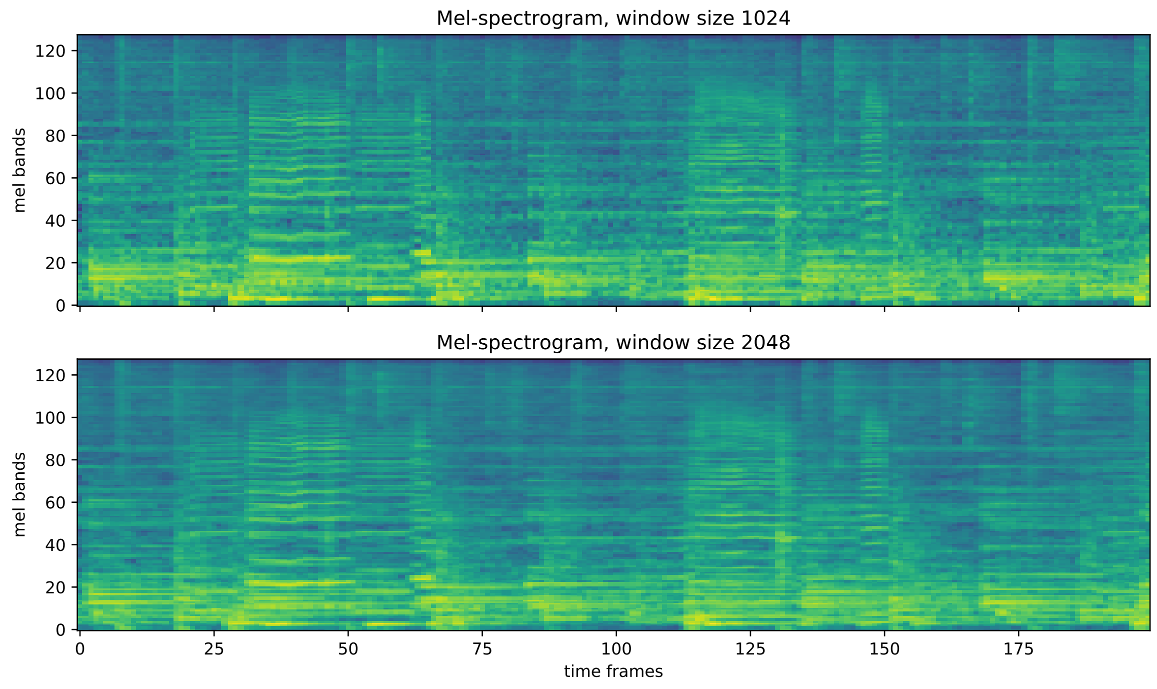


Figure 5.13: Comparison of window size variation in STFT to reduce spectral leakage between mel-bands.

The experiment results in diverse outcomes: in the case of the IRMAS data set, a marginal improvement of 1-2% in the mean f-scores was achieved. Conversely, using the Monotimbral data set results in a decrease of 1-3% in all mean f-scores. Given this inconsistency, further tests were carried out in section 5.6.

5.5.2 Normalization through standarization

Batch normalization is a recent innovation of adaptive reparametrization, motivated to improve training of very deep models [25]. Concretely, it permits more efficient backpropagation learning, reaching faster convergence [43]. It can be applied to any input and has the aim to normalize the data to have zero mean and unit variance. Normalization can be achieved through standarization, which uses the mean and standard deviation of the features (see Figure 5.14). The standardized data $X' = (X - \mu)/\Sigma$, where X is a mel-spectrogram, μ is the mean, and Σ the standard deviation over all training samples. Standarization was implemented following [44] on data normalization in convolutional neural networks. The authors concluded that other methods, like Zero Component Analysis and Mean Normalization can result in performance improvements over standarization, but this is still a researched topic and standarization is easily implemented.

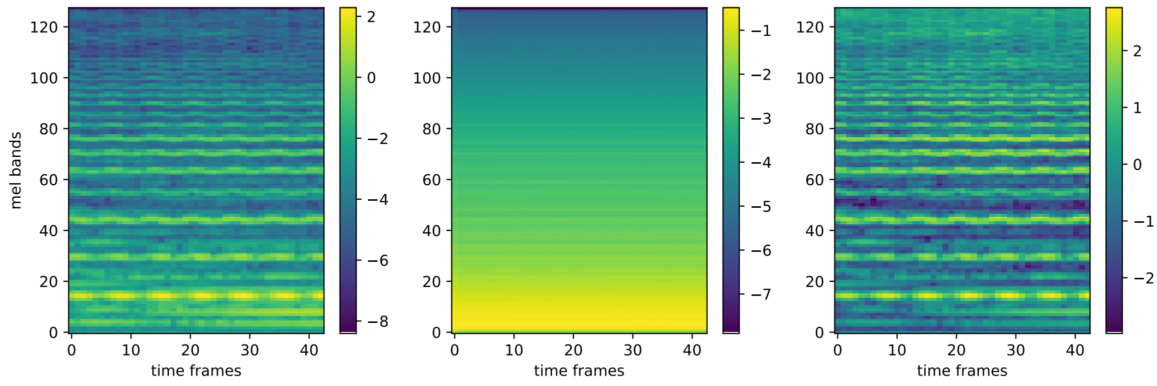


Figure 5.14: Effect of standarization: original mel-spectrogram (left), mean over all training instances (center), and resulting mel-spectrogram (right).

Figure 5.15 presents the improvement of using normalization: less variance in training loss and accuracy across training iterations, compared to Figure 5.4 which has been trained without normalization. Given that the data is normalized, gradient descent now results into similar directions during gradient descent, reducing the variance between iterations. Additionally, the amount of epochs required to stop is reduced from a range of 40-50 epochs to 30-40 epochs per training.

Opposing results have been obtained on the IRMAS and Monotimbral data sets with respect to the obtained performance metrics. While the IRMAS data set shows a general improvement of 3% in all f-scores and 3% in the mean accuracy, the Monotimbral data set shows a decrease of 8-9% in all f-scores and 3% in the mean accuracy.

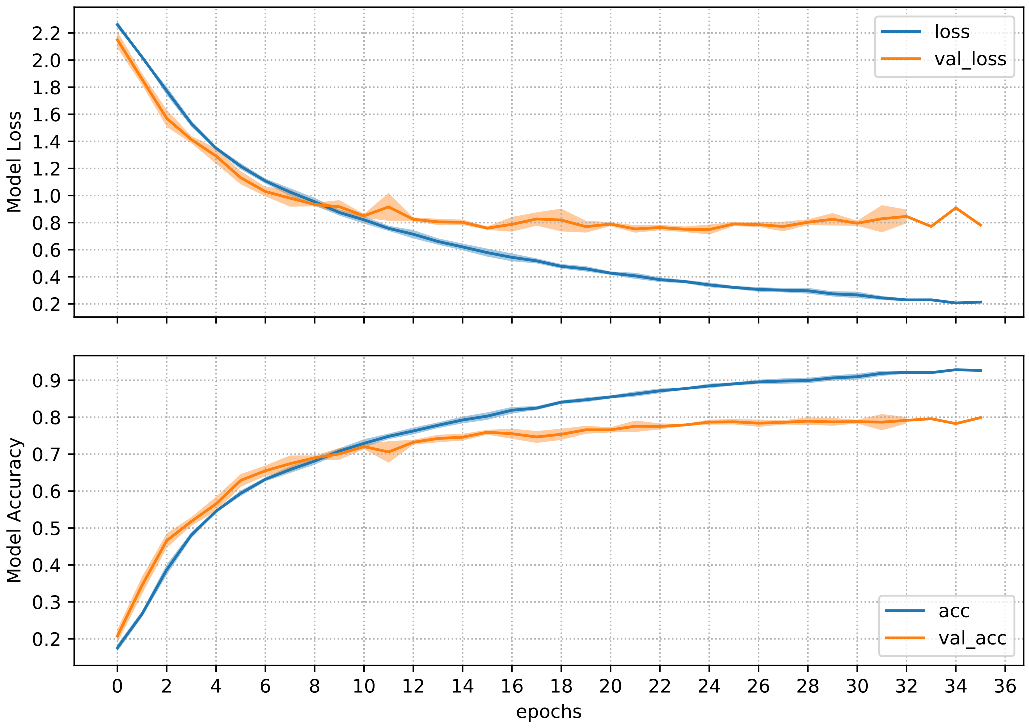


Figure 5.15: Effect of standarization in training performance.

5.6 Proposed Experiments: Classification stage

The classifier was tested with different configurations to improve performance metrics, but none of these tests resulted in enhancement of the recognition task. Nonetheless, they have been documented for future work.

5.6.1 Optimizers: Adam and Stochastic Gradient Descent

Following the baseline system [3], training has been performed with Adam optimizer and learning rate of 0.001. The use of different optimizers can lead to improvement of performance metrics. The model created by Pons et al. [37] for instrument recognition was trained using stochastic gradient descent. SGD has been used to train the implemented baseline model with the same learning rate and Nesterov momentum of

0.9. The difference between Nesterov and standard momentum is where the gradient is evaluated, which introduces a correction factor to the standard momentum [25].

Due to the sizable amount of samples in the data sets and trainable parameters in the convolutional neural network, using Adam results in early stopping after 30-40 epochs. Conversely, SGD reaches early stopping after 80-120 epochs. Since every experiment requires training a new model, experimentation with SGD was only implemented for transfer learning experiments in section 5.9.

5.6.2 Network architecture

Recent research by Grill and Schlüter for music boundary detection altered the architecture of a convolutional neural network, to combine input features as information fusion [45]. Their results improved by using fusion in a hidden dense layer or in the a convolutional layer. In the case of instrument recognition, a new model was created with two branches of convolutional layers that converge by the concatenation of tensors [21] (which is denoted as C) in a flatten layer previous to the fully connected layers, as shown in Figure 5.16. With this architecture it is possible to input both harmonic and percussive separated elements simultaneously. This results in doubling the amount of trainable parameters to 2.8 million.

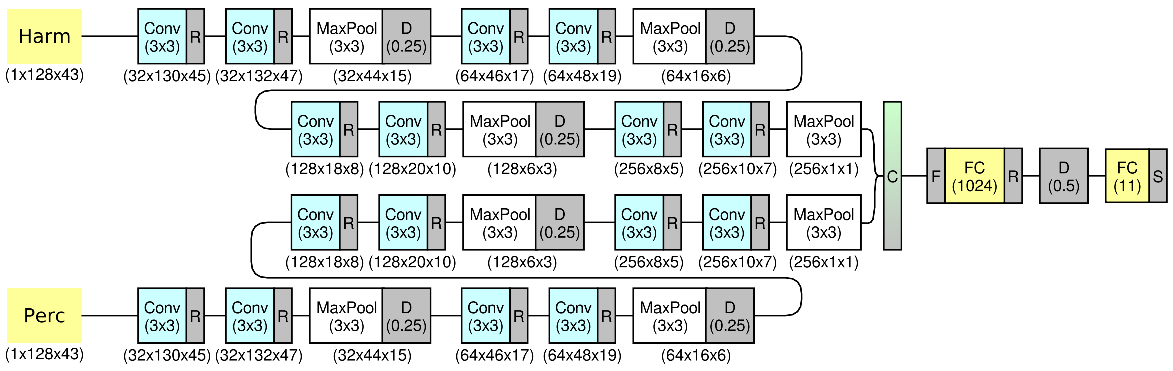


Figure 5.16: Model with fusion in fully connected layer to input simultaneous harmonic and percussive components.

Input fusion after the convolutional layers results in a overall decrease of performance metrics in both data sets. Testing this model results in an average loss of 1-2% in all f-scores using IRMAS data set, and 6-7% in all f-scores using Monotimbral data set.

5.6.3 Three dimensional mel-spectrograms input

Schüter and Böck have researched musical onset detection with the use of convolutional neural networks [46]. They stacked multi-resolution spectrograms obtained by altering the window size of the STFT. The use of this ”three-channel” input results in significantly improving the note onset detection task. This approach has also been used by Takahashi et al. [47] for acoustic event detection. As shown section 5.5, an enhancement of frequency resolution of mel-spectrograms results in ambiguous outcome depending on the data set used (improvements for the IRMAS data set and loss for the Monotimbral data set). A new model was trained by creating stacked mel-spectrograms that were extracted with increasing window size: 1024, 2048, and 4096 time frames. This results in a change in the input dimensionality of the zero padding in the first convolutional layer (see Figure 5.17).

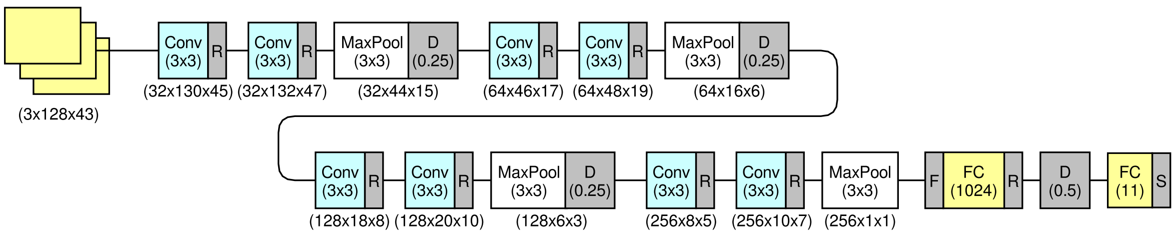


Figure 5.17: Model with three dimensional mel-spectrograms as input.

Using three-dimensional input results in a overall decrease of performance metrics in both data sets. This model results in an average loss of 3-4% in all f-scores using the IRMAS data set, and 4-5% in all f-scores using the Monotimbral data set.

5.7 Proposed Experiments: Post-processing stage

One experiment was implemented in the post-processing stage: an extension of the optimal global thresholding implemented by Han et al. [3].

5.7.1 Class-wise thresholding

As mentioned in section 4.1 and shown in Figure 5.5, the authors calculated an optimum global identification threshold to obtain the final performance metrics using the pure test data set. This operation was extended to obtain optima class-wise identification thresholds for each instrument. As Figures 5.18 and 5.19 show, the identification threshold was varied from 0.0 to 1.0 while extracting class-wise performance metrics. The figures show the mean f-score across training iterations, and optima thresholds are plotted as *. Moreover, the plots are arranged in such a way that the first plots (upper left) have the lowest optimum threshold and the last (lower right) present the highest. Figure 5.18 shows that instruments more commonly prominent in the mix (e.g., violin, saxophone, flute, trumpet, and voice) report higher optima thresholds. Conversely, instruments as piano, organ, and electric guitar are usually less salient and present lower optima thresholds, as accompaniment instruments.

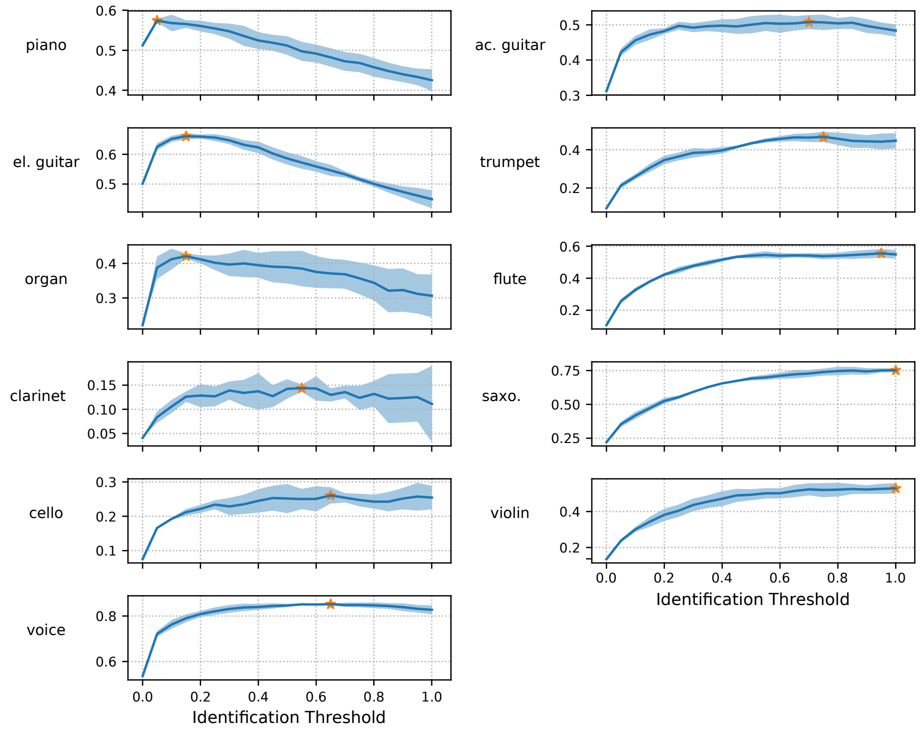


Figure 5.18: Optima class-wise identification thresholds calculated on f-scores from IRMAS development test data set.

It is important to remark that the previous analysis is only valid for the IRMAS data set and not for the Monotimbral data set. Figure 5.19 shows that instruments that should have a similar saliency (e.g., male and female singing voice) report completely different optima identification thresholds. Nonetheless, both data sets report improvements in performance metrics as shown in figure 5.20. IRMAS data set reports 2-4% improvements in all f-scores and Monotimbral data set has less than 1% variation, improving recall.

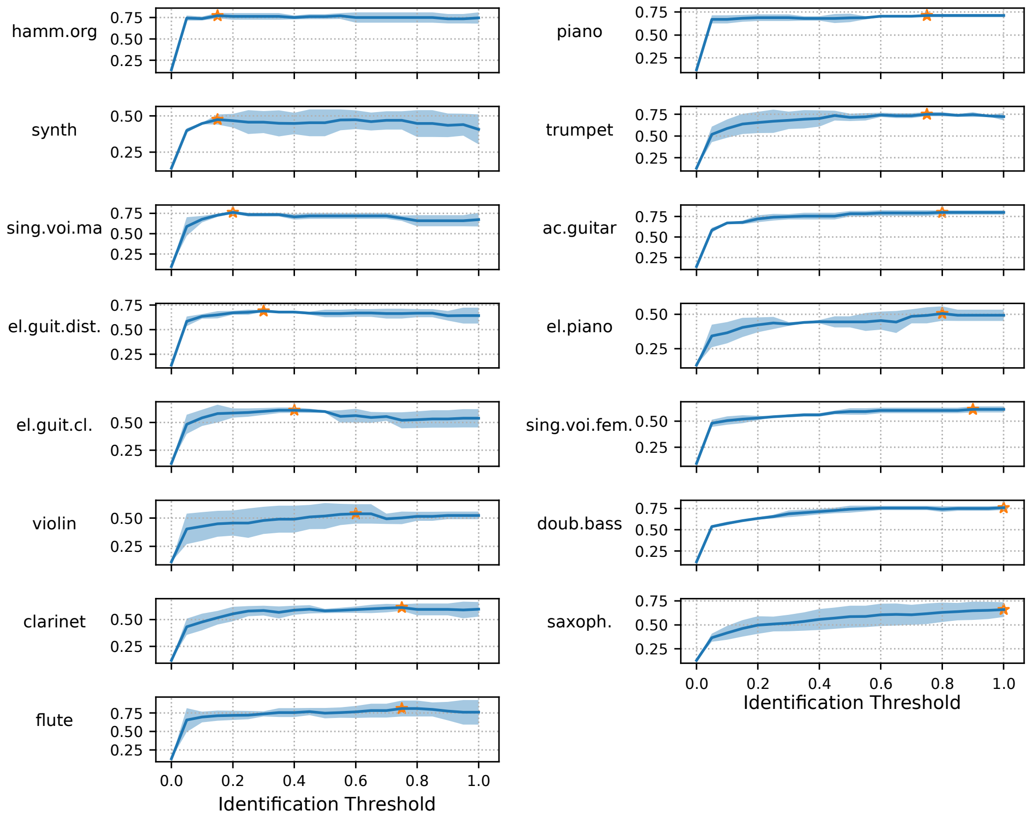


Figure 5.19: Optima class-wise identification thresholds calculated on f-scores from Monotimbral development test data set.

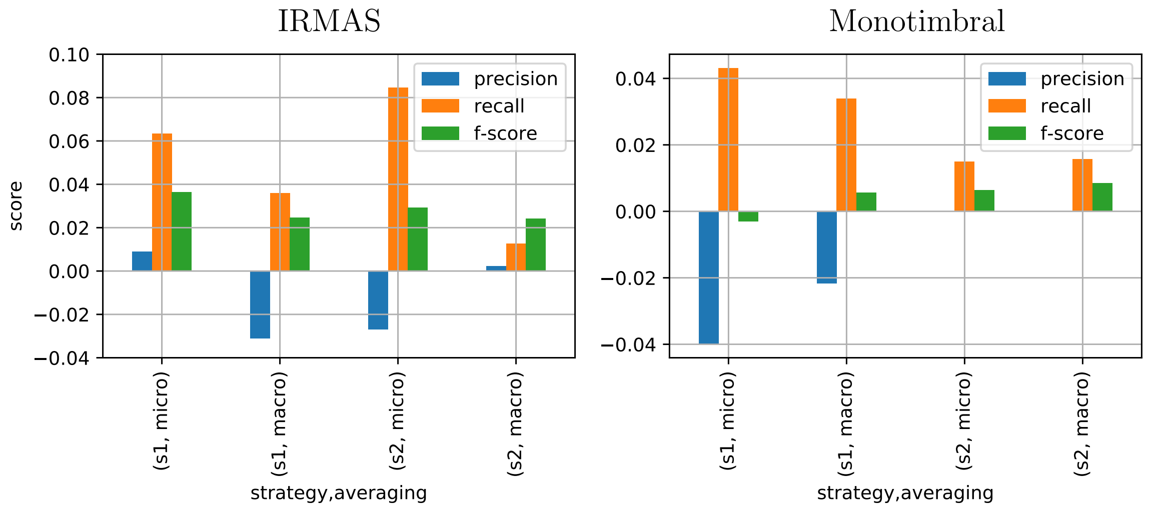


Figure 5.20: Score improvement by using optima class-wise thresholds on each data set: IRMAS (left) and Monotimbral (right).

5.8 Proposed System

Taking into consideration all the experiments that were carried out, an instrument recognition model is proposed. Given that the task is *predominant* instrument classification, the improvement gain on evaluation of the IRMAS data set is more significant than with the Monotimbral data set. Several experiments were conducted to find the combination of procedures that result in optimum performance. With the aim of maximizing recognition metrics, the proposed model employs *phase-based harmonic/percussive separation* and *class-wise thresholding*. The use of these two pre- and post-processing stages leads to an overall improvement of 4% in all f-scores of the IRMAS data set (see Figure 5.21). Regarding the Monotimbral data set, decline in performance results from the experiments in section 5.4. In this case and considering macro averaging, the use of class-wise thresholding increases the f-score enough to obtain a similar performance. Additionally, both data sets show the tendency of decreasing precision, while increasing recall, also shown in Figure 5.20. This suggests that the amount of *false positives* may be increasing, while the *false negatives* are decreasing. This tendency appears also using only class-wise thresholding, which suggests that it is responsible for this behavior (see Figure 5.20). Conversely, phase-based harmonic/percussive separation results the opposite outcome: increasing precision and decreasing recall (see Figure 5.9). Both of these effects appear to balance out to obtain an overall improvement in the processing of multitimbral music, as the IRMAS data set.

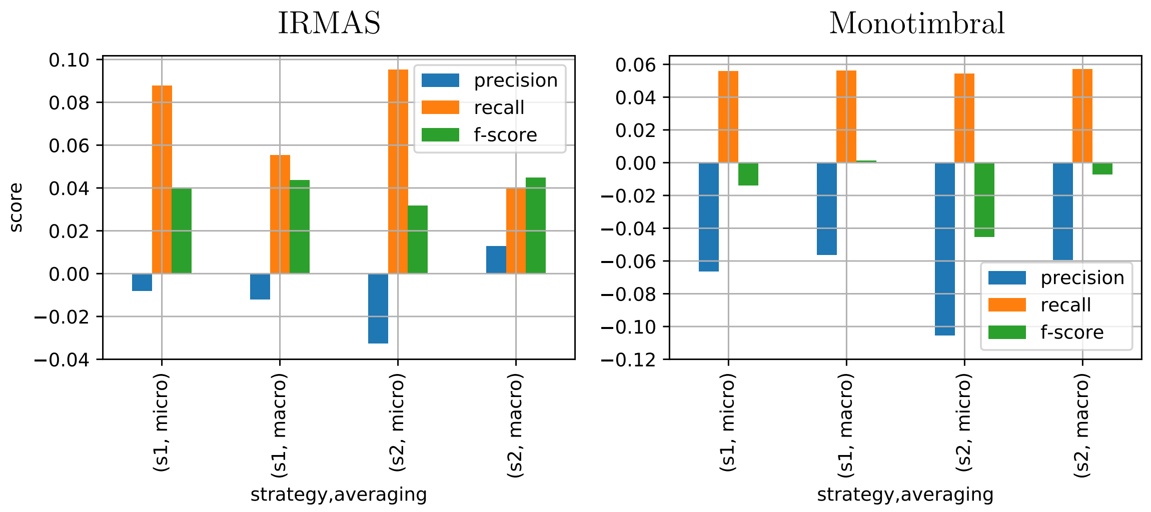


Figure 5.21: Score improvement for the proposed system on each data set: IRMAS (left) and Monotimbral (right).

5.9 Proposed use case: Jazz Solo Instrument Recognition

A concrete use case has been proposed in the context of jazz solo instruments using the pitch-informed solo/accompaniment separation algorithm. This task has two principal challenges: timbral similarity between sub-classes (e.g., alto, soprano, and tenor saxophone), and the large variety of recording conditions that influence the overall sound of the recording.

Baseline models have been trained with the IRMAS Wind data set and improvements have been obtained when employing solo/accompaniment separation, as shown previously in section 5.4. To extend its functionality for this use case, transfer learning was implemented to continue training with new data. As a benchmark, the baseline model was also trained with the Jazz Solo data set and its solo separated version, with the same parameters established in section 5.3. Table 5.4 reports the f-scores obtained in both cases using aggregation strategy *S1*. Concretely, using solo/accompaniment separation with the IRMAS Wind data set results in an improvement of 3-4%, while using the Jazz data set improves 14-15% by training each model from scratch. With respect to the aggregation strategy, *S1* results in improved performance metrics. This is possibly due to the prominence of jazz solo instruments in the music, as argued in section 5.3. It is important to mention that both micro- and macro-averaging scores are very similar, consistent with using a balanced split of data of this particular data set (see section 5.1).

		F-Score	
Data set	S/A Separation	Micro	Macro
IRMAS Wind	-	0.684	0.598
IRMAS Wind	✓	0.713	0.636
Jazz	-	0.657	0.669
Jazz	✓	**0.805**	**0.803**

Table 5.4: Performance metrics obtained by training the baseline model with mentioned data sets.

Two transfer training approaches were tested. Firstly, replacing the last layer of the neural network and train with a learning rate of 0.01, while keeping the remaining network weights fixed, referred to as ”one-pass”. Secondly, perform the previous training and then fine-tune all layers using a lower learning rate (0.001), referred to as ”two-pass”. Table 5.5 shows performance scores when using different optimizers

(Adam and Stochastic Gradient Descent), for different approaches (one pass and two pass), and different activation functions for the final layer. The columns with the data sets contain: pre-training refers to the training data that has been used in the initial training stage and transfer learning refers to the training data for retraining.

During the transfer learning stage, each model has been further trained with consistent data to the initial training stage (see S/A column in Table 5.5). For example, a model that has been trained with the IRMAS Wind data set (containing polytimbral mixtures), has been further trained with the Jazz data set (without source separation pre-processing). Conversely, a model that has been trained with the solo component of the IRMAS Wind data set (which contains a monotimbral mixture), has been further trained with the solo component of the Jazz data set. Additionally, tests have been carried out by applying transfer learning to the model trained with the complete IRMAS data set (containing all instruments), resulting in a loss of performance.

				Data sets		F-score	
Optimizer	Approach	Activation	S/A	Pre-Training	Transfer Learning	Micro	Macro
Adam	One pass	Softmax	-	IRMAS Wind	Jazz	0.612	0.635
			✓	IRMAS Wind	Jazz	0.729	0.735
		Sigmoid	-	IRMAS Wind	Jazz	0.605	0.621
			✓	IRMAS Wind	Jazz	0.738	0.748
	Two pass	Sigmoid	-	IRMAS Wind	Jazz	0.583	0.610
			✓	**IRMAS Wind**	**Jazz**	**0.751**	**0.768**
			-	IRMAS	Jazz	0.417	0.440
SGD	One pass	Softmax	-	IRMAS Wind	Jazz	0.590	0.611
			✓	IRMAS Wind	Jazz	0.741	0.751
		Sigmoid	-	IRMAS Wind	Jazz	0.599	0.618
			✓	IRMAS Wind	Jazz	0.566	0.608
	Two pass	Sigmoid	-	IRMAS Wind	Jazz	0.604	0.595
			✓	IRMAS Wind	Jazz	0.524	0.514
			-	IRMAS	Jazz	0.391	0.419

Table 5.5: Performance metrics obtained modifying optimizer, transfer learning approach, output activation function, and different pre-trained models.

Comparison between the best models obtained in table 5.4 and 5.5 shows performance loss by implementing transfer learning. While training a model from scratch results in an improved overall performance, applying transfer learning to a model trained with the solo component of the IRMAS Wind data set reaches similar performance metrics (with an approximate loss 5%). Moreover, tests using the softmax activation function did not result in significant improvement, so they were not implemented with the ”two-pass” approach.

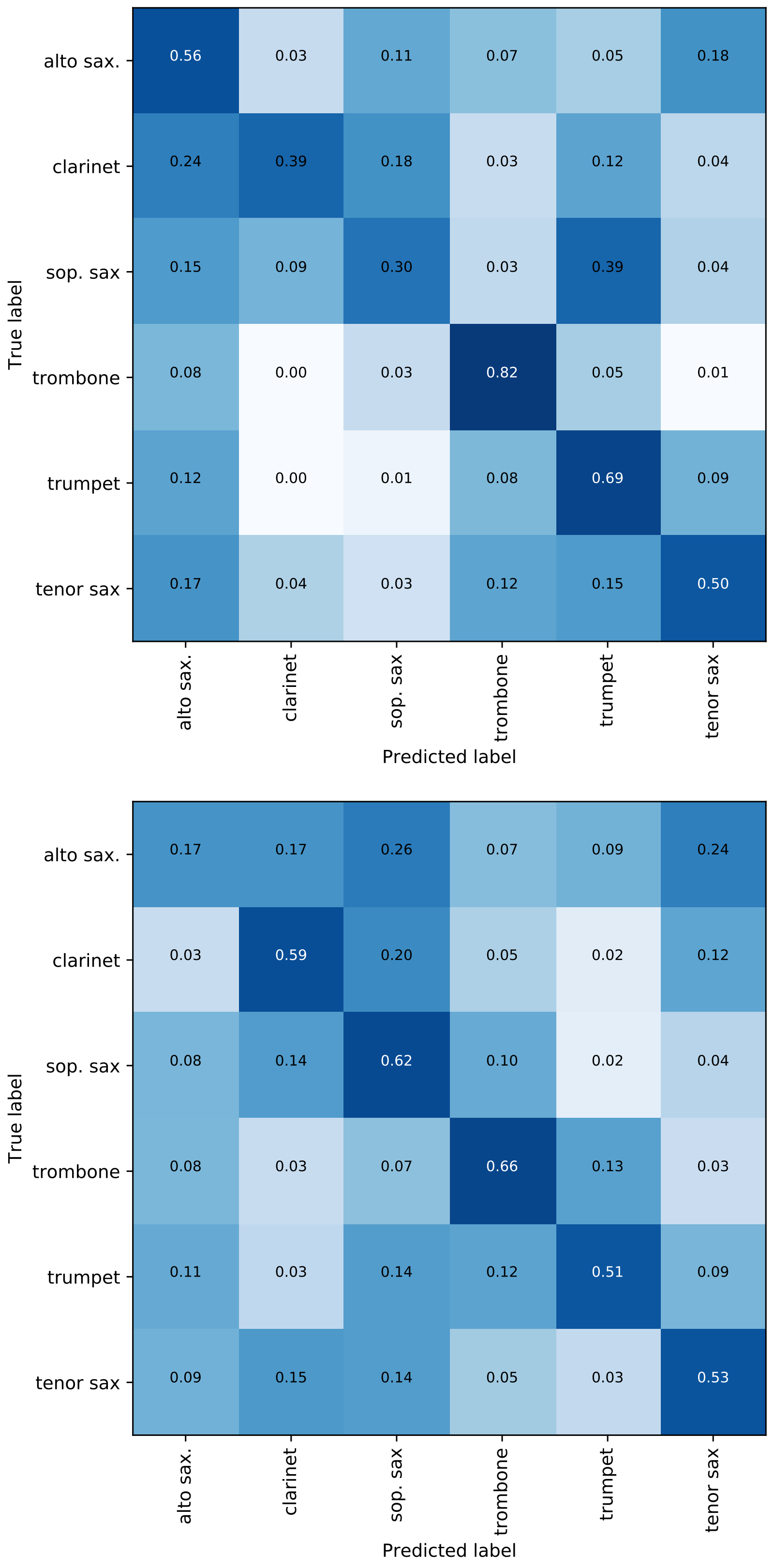


Figure 5.22: Comparison of confusion matrices: fully trained baseline with Jazz SS (top) and transfer trained model trained initially with IRMAS Wind SS (bottom).

The fully trained model results in a mean accuracy of 54.43%, while the transfer trained model reports 51.28%. The confusions between both models can be seen in Figure 5.22. Firstly, regarding saxophone in the case of the fully trained network (top), the alto saxophone is best classified in the case of transfer learning (bottom). Since the latter was pre-trained with four classes (clarinet, flute, saxophone, and trumpet), the training saxophone examples can be similar to the ones contained in the new soprano and tenor saxophone labels. Concretely, the model pre-trained with the IRMAS Wind data set improves the recognition of soprano and tenor saxophone, possibly due to that it contains these samples.

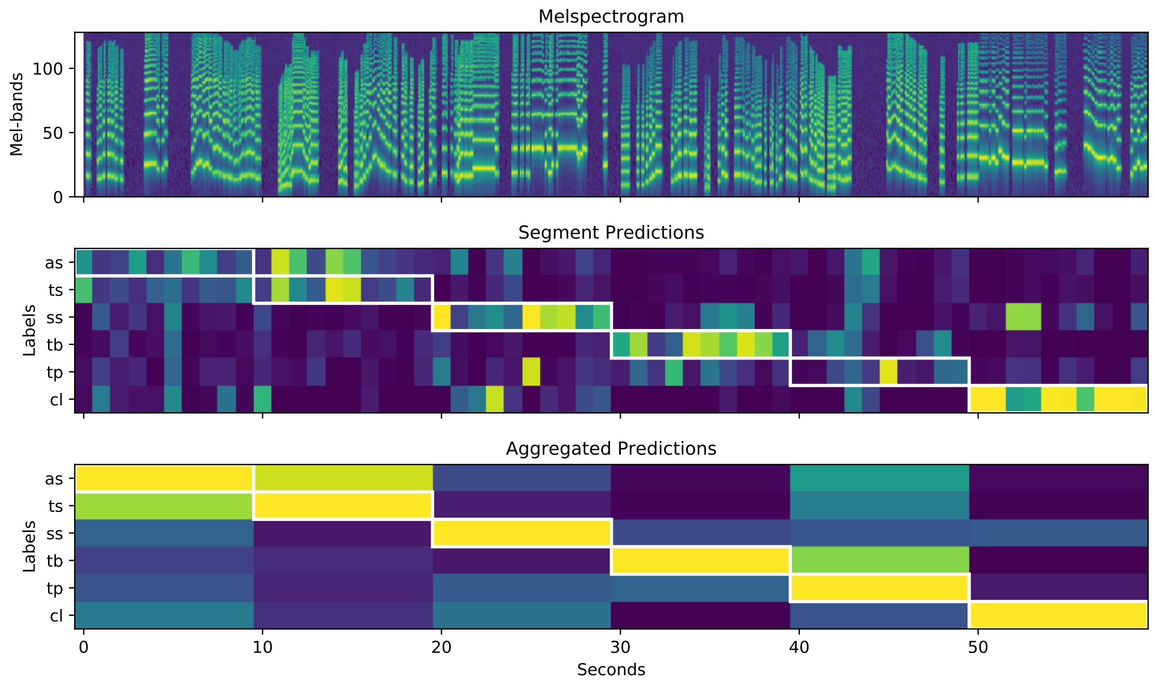


Figure 5.23: Melspectrogram of six jazz solos (top), segment-wise predictions using transfer learning two-pass model (middle), and aggregated predictions with strategy S1 (bottom). Clip-wise ground truth is plotted in white rectangles.

Regarding transfer training, the recognition of clarinet is improved due to the amount of training examples in the pre-training data set. Although trumpet has also been previously used for training, its confusion with soprano saxophone increases. Since both instruments have a very similar pitch range (soprano saxophone: B♭3 - F♯6, trumpet: F♯3 - F♯6), its possible that the model takes into account pitch range during classification, which leads to a possible outlook in post-processing stages (see Chapter 6). In general, both transfer training and training from scratch result in overall performance improvements with respect to the baseline models for this particular use case.

Figure 5.23 shows segment- and clip-wise predictions obtained from six jazz solos classified with the best model using transfer learning and solo/accompaniment separation. One solo of each instrument was used as input to the system: Ornette Coleman - Ramblin (alto saxophone), Buddy DeFranco - Autumn Leaves (clarinet), John Coltrane - My Favorite Things (soprano saxophone), Frank Rossolino - Moonlight in Vermont (trombone), Lee Morgan - The Sidewinder (trumpet), and Michael Brecker - African Skies (tenor saxophone). Each solo had a duration of 10 seconds and predictions were aggregated using the S1 strategy to obtain a clip-wise prediction. Comparison between segment- and clip-wise predictions show the importance of the post-processing stage to improve the overall recognition performance.

6 Conclusions

6.1 Summary

The baseline model for automatic instrument recognition developed by Han et al. [3] was implemented and the results reported by the authors were reproduced. Both phase-based harmonic/percussive [1] and pitch-informed solo/accompaniment [2] separation algorithms were used to pre-process the data. Different data sets with variable amount of instruments and complexities have been used to evaluate the system. The data sets used include: the IRMAS data set (which has been used in the reference model), the Monotimbral data set (that results in improvements in performance metrics and motivates the use of a pre-processing stage), the IRMAS Wind data set (a brass and woodwind subset of the original IRMAS data set that was used on the solo/accompaniment separation algorithm), and the Jazz Solo data set (that was used to test transfer learning applications and develop a concrete use case).

Several experiments have been carried out in all stages of the algorithmic structure of instrument recognition systems. These experiments have led to the proposal of a system which implements harmonic/percussive separation as a pre-processing stage and class-wise thresholding as post-processing. The use of this system must be further developed to gain previous knowledge on the amount of instruments of the mixture, and adapt the activation of the harmonic/percussive separation algorithm accordingly. The post-processing experiments suggest that the aggregation strategies developed by [3] should also be used depending on the scenario. In the case where the output is multi-labeled, strategy *S2* results in best performance metrics. Conversely, strategy *S1* performs best on single-labeled data. Additionally, a use case has been proposed to evaluate Jazz Solo instrument recognition with the use of the solo/accompaniment separation algorithm. The use case results in improvement of the performance metrics, with respect to the baseline model. Additionally, transfer learning techniques do not appear to improve the classification of instruments sub-classes. Conversely, the models trained with transfer learning improve in the classification of instruments already existent on the pre-training stage. Both approaches (training from scratch and using transfer learning) result in overall improvements of performance metrics, with respect to the baseline model.

6.2 Outlook

Several tasks can be outlined as an outlook to continue researching automatic instrument recognition:

- *Annotation improvement*: wrongful annotations on the training and testing data sets result in an overall loss of performance of the system. Further research should include analysis and purging on weakly-labeled data sets.
- *Adaptive harmonic/percussive separation*: since the use of this pre-processing stage can lead to performance decline with monotimbral material as input, the implementation of a system should be able to adapt the activation of the source separation algorithm, estimating the number of distinct instruments in the mixture.
- *Post-processing enhancement*: the output of the pitch estimation algorithm used in solo/accompaniment separation could be used in the post-processing stage. This approach should be able to weight class-wise predictions, depending on the natural pitch range of the instruments.
- *Aggregation strategies*: the reference model results in best performance when using aggregation strategy S2, while all results in the use case have best performance using S1. This motivates to continue research on how to improve the post-processing stage.
- *Stability test*: data perturbations (e.g., adding white noise to the audio) could be introduced to the time-domain signal to further test the stability of the system. A suggested approach is the one developed by Zheng et al. [48], who added perturbations to the data and the convolutional layers to improve the generalization of the model, regardless random disturbances.

Bibliography

[1] Estefanía Cano, Mark D. Plumbley, and Christian Dittmar, "Phase-based harmonic/percussive separation," in *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, Singapore, 2014, pp. 1628–1632.

[2] Estefanía Cano, Gerald Schuller, and Christian Dittmar, "Pitch-informed solo and accompaniment separation towards its use in music education applications," *EURASIP Journal on Advances in Signal Processing*, vol. 2014, pp. 23, 2014.

[3] 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*, vol. 25, no. 1, pp. 208–221, Jan 2017.

[4] Ferdinand Fuhrmann, *Automatic musical instrument recognition from polyphonic music audio signals*, Ph.D. thesis, Universitat Pompeu Fabra, 2012.

[5] Meinard Müller, *Fundamentals of music processing: audio, analysis, algorithms, applications*, Springer, 2015.

[6] Juan Bosch, Jordi Janer, Ferdinand Fuhrmann, and Perfecto Herrera, "A comparison of sound segregation techniques for predominant instrument recognition in musical audio signals," in *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, Porto, Portugal, 2012, pp. 559–564.

[7] Antti Eronen and Anssi Klapuri, "Musical instrument recognition using cepstral coefficients and temporal features," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing(ICASSP)*, Istanbul, Turkey, 2000, vol. 2, pp. 753–756.

[8] A. G. Krishna and T. V. Sreenivas, "Music instrument recognition: from isolated notes to solo phrases," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Quebec, Canada, May 2004, vol. 4, pp. 265–268.

[9] Aleksandr Diment, Padmanabhan Rajan, Toni Heittola, and Tuomas Virtanen, "Modified group delay feature for musical instrument recognition," in *Proceedings of the 10th International Symposium on Computer Music Multidisciplinary Research*, Marseille, France, 2013, pp. 431–438.

[10] Li-Fan Yu, Li Su, and Yi-Hsuan Yang, "Sparse cepstral codes and power scale for instrument identification," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Florence, Italy, May 2014, pp. 7460–7464.

[11] Slim Essid, Gael Richard, and Bertrand David, "Musical instrument recognition on solo performances," in *Proceedings of the 12th European Signal Processing Conference (EUSIPCO)*, Vienna, Austria, Sept 2004, pp. 1289–1292.

[12] Mikus Grasis, Jakob Abeßer, Christian Dittmar, and Hanna Lukashevich, “A multiple-expert framework for instrument recognition,” in *Proceedings of the 10th International Symposium on Computer Music Multidisciplinary Research (CMMR)*, Marseille, France, October 2013, pp. 619–634.

[13] Sharaj Panwar, Arun Das, Mehdi Roopaei, and Paul Rad, “A deep learning approach for mapping music genres,” in *Proceedings of the 12th System of Systems Engineering Conference (SoSE)*, Waikoloa, USA, June 2017, pp. 1–5.

[14] 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 *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, USA, March 2017, pp. 261–265.

[15] Ferdinand Fuhrmann and Perfecto Herrera, “Polyphonic instrument recognition for exploring semantic similarities in music,” in *Proceedings of the 13th International Conference of Digital Audio Effects (DAFx-10)*, Graz, Austria, September 2010, pp. 1–8.

[16] Stefan Uhlich, Franck Giron, and Yuki Mitsufuji, “Deep neural network based instrument extraction from music,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brisbane, Australia, April 2015, pp. 2135–2139.

[17] Richard Vogl, Matthias Dorfer, and Peter Knees, “Drum transcription from polyphonic music with recurrent neural networks,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, USA, March 2017, pp. 201–205.

[18] Tetsuro Kitahara, Masataka Goto, Kazunori Komatani, Tetsuya Ogata, and Hiroshi Okuno, “Instrument identification in polyphonic music: Feature weighting to minimize influence of sound overlaps,” *EURASIP Journal on Advances in Signal Processing*, vol. 2007, no. 1, Nov 2006.

[19] Sascha Grollmisch, Estefanía Cano Cerón, and Christian Dittmar, “Songs2see: Learn to play by playing,” in *Audio Engineering Society Conference: 41st International Conference: Audio for Games*, Feb 2011.

[20] Toni Heittola, Anssi Klapuri, and Tuomas Virtanen, “Musical instrument recognition in polyphonic audio using source-filter model for sound separation,” in *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR)*, Kobe, Japan, 2009, pp. 327–332.

[21] François Chollet et al., “Keras,” `https://github.com/keras-team/keras`, 2015, Accessed: 2017-10-10.

[22] Andrew Trahan, “Musical instrument recognition in polytimbral polyphonic music audio signals,” 2013.

[23] Christopher M. Bishop, *Pattern recognition and machine learning*, Springer, New York, 2006.

[24] Aurélien Géron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, O’Reilly Media Inc., 1st edition, 2017.

[25] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning*, The MIT Press, 2016.

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems 25*, pp. 1097–1105. Curran Associates Inc., 2012.

[27] Deng Yu and Li Deng, *Automatic Speech Recognition: A Deep Learning Approach*, Signals and Communication Technology. Springer London, 2014.

[28] François Chollet, *Deep Learning with Python*, Manning Publications Company, 2017.

[29] Joe Kiefer and J. Wolfowitz, “Stochastic estimation of the maximum of a regression function,” *The Annals of Mathematical Statistics*, vol. 23, no. 3, pp. 462–466, Sept 1952.

[30] Diederik P. Kingma and Jimmy Ba, “Adam: A method for stochastic optimization,” *CoRR*, vol. abs/1412.6980, 2014.

[31] “Cs231n convolutional neural networks for visual recognition - stanford uniersity,” `https://cs231n.github.io/`, Accessed: 2018-03-16.

[32] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” *CoRR*, vol. abs/1502.01852, 2015.

[34] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson, “How transferable are features in deep neural networks?,” *CoRR*, vol. abs/1411.1792, 2014.

[35] Taejin Park and Taejin Lee, “Musical instrument sound classification with deep convolutional neural network using feature fusion approach,” *CoRR*, vol. abs/1512.07370, 2015.

[36] Peter Li, Jiyuan Qian, and Tian Wang, “Automatic instrument recognition in polyphonic music using convolutional neural networks,” *CoRR*, vol. abs/1511.05520, 2015.

[37] Jordi Pons, Olga Slizovskaia, Rong Gong, Emilia Gómez, and Xavier Serra, “Timbre analysis of music audio signals with convolutional neural networks,” *CoRR*, vol. abs/1703.06697, 2017.

[38] Alexey Ozerov, Emmauel Vincent, and Frederic Bimbot, “A general flexible framework for the handling of prior information in audio source separation,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 4, pp. 1118–1133, May 2012.

[39] Brian McFee, Colin Raffel, Dawen Liang, Daniel P. W. Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto, “librosa: Audio and music signal analysis in python,” 2015.

[40] Karin Dressler, *Automatic transcription of the melody from polyphonic music*, Ph.D. thesis, TU Ilmenau, Germany, Jul 2017.

[41] Martin Pfleiderer, Klaus Frieler, Jakob Abeßer, Wolf-Georg Zaddach, and Benjamin Burkhart, Eds., *Inside the Jazzomat - New Perspectives for Jazz Research*, Schott Campus, 2017.

[42] Kenneth W. Berger, “Some factors in the recognition of timbre,” *The Journal of the Acoustical Society of America*, vol. 36, no. 10, pp. 1888–1891, 1964.

[43] Yann LeCun, Leon Bottou, Genevieve B. Orr, and Klaus Robert Müller, *Efficient BackProp*, pp. 9–50, Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[44] Kuntal Kumar Pal and K. S. Sudeep, "Preprocessing for image classification by convolutional neural networks," in *Proceedings of the IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, Bangalore, India, May 2016, pp. 1778–1781.

[45] Thomas Grill and Jan Schlüter, "Music Boundary Detection Using Neural Networks on Spectrograms and Self-Similarity Lag Matrices," in *Proceedings of the 23rd European Signal Processing Conference (EUSIPCO)*, Nice, France, 2015, pp. 1296–1300.

[46] Jan Schlüter and Sebastian Böck, "Improved Musical Onset Detection with Convolutional Neural Networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2014)*, Florence, Italy, May 2014, pp. 6979–6983.

[47] Naoya Takahashi, Michael Gygli, Beat Pfister, and Luc Van Gool, "Deep convolutional neural networks and data augmentation for acoustic event detection," *CoRR*, vol. abs/1604.07160, 2016.

[48] Stephan Zheng, Yang Song, Thomas Leung, and Ian J. Goodfellow, "Improving the robustness of deep neural networks via stability training," *CoRR*, vol. abs/1604.04326, 2016.

List of Figures

List of Tables

List of Abbreviations and Symbols

F	F-Score
P	Precision
R	Recall
ADSR	Attack-Decay-Sustain-Release
AM	Amplitude Modulation
CNN	Convolutional Neural Network
FASST	Flexible Audio Source Separation Framework
FFT	Fast Fourier Transform
FM	Frquency Modulation
Fraunhofer IDMT	Fraunhofer Institute for Digital Media Technology
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
Hz	Hertz
IRMAS	Instrument Recognition in Music Audio Signals
ISTFT	Inverse Short Time Fourier Transform
KNN	k-Nearest Neighbour
LDA	Linear Discriminant Analysis
LReLU	Leaky Rectified Linear Unit
LRMS	Left/Right-Mid/Side Separation
MFCC	Mel-Frequency Cepstral Coefficient
MIR	Music Information Retrieval
MRP	Multi-resolution Recurrence Plots
NMF	Non-Negative Matrix Factorization
PCA	Principal Component Analysis
PReLU	Parametric Rectified Linear Unit
ReLU	Rectified Linear Unit
SGD	Stochastic Gradient Descent
STFT	Short Time Fourier Transform
SVM	Support Vector Machine

Declaration of Originality

I hereby declare that this thesis was created autonomously without using other than the stated references. All parts which are cited directly or indirectly are marked as such. This thesis has not been used in the same or similar forms in parts or total in other examinations.

Ilmenau, 31.03.2018

Juan Sebastián Gómez Cañón