ulm university universität
uulm

Ulm University | 89069 Ulm | Germany

Faculty of
Engineering, Computer Science and Psychology
Neural Information Processing

Instrumental Classification

Master Project at Ulm University

Submitted by:
Jonathan Wunden , Tanja Zast
jonathan.wunden@uni-ulm.de , tanja.zast@uni-ulm.de

Reviewer:
Prof. Dr. Friedhelm Schwenker

Supervisor:
Sebastian Gottwald

2023

Version from August 11, 2023

Abstract

In this report we investigate possibilities for the classification of musical instruments. We specifically focus on multi-label classification and how it can be applied to this domain. We use classifier chains as our primary method since we assume that there is a correlation of the different instruments in a musical excerpt. We also apply our multi-label models to a single-label testing dataset to investigate whether the vast domain of audio clips can be used to classify single instruments. Our results show promising results that are up to par with other papers that used a single-label dataset for training.

Contents

1 Introduction **1**

1.1 Inspiration and Setting 1

1.2 The Challenge at Hand 1

2 Exploring Related Works **3**

2.1 Methods of Feature Extraction 3

2.2 Key Deep Learning Architectures 4

2.3 Multi-label Classification 5

2.4 The Art of Optimization 5

3 Methodology **6**

3.1 Mel Frequency Cepstral Coefficient 6

3.2 Spectral Centroid Frequency 7

3.3 Spectral Rolloff 8

3.4 Spectral Bandwidth 9

3.5 Zero-Crossing Rate 10

3.6 Deep Learning Architectures 12

3.6.1 Multi Layer Perceptron 12

3.6.2 Support Vector Machine Learning 14

3.7 Multi-label Classification 16

3.8 Metrics 17

3.9 Bayesian Optimization 18

3.10 Dataset 19

4 Experiments and Results **21**

4.1 Procedure 21

4.2 Results 23

5 Discussion **25**

5.1 Evaluating the Results 25

5.2 Future Work 26

6 Conclusion **28**

1 Introduction

1.1 Inspiration and Setting

The complexity and expressiveness of music are deeply enriched by the variety and nuance of musical instruments. However, their accurate classification remains a considerable challenge. In this project, we aim to confront this challenge head-on, with the goal of creating a computational solution for instrument classification [8].

The domain of music information retrieval (MIR) is an ever-growing field, with an increasing focus on automatic recognition and classification of musical instruments. In our project, we are motivated to build upon the solid foundations laid by previous research in the field, leveraging strategies from conventional machine learning [18].

Our fundamental goal is to design, build, and assess a model capable of accurately distinguishing various musical instruments using audio signals. Recent breakthroughs in deep learning suggest promising prospects in tackling this challenge, given its impressive performances in areas like image and speech recognition [10].

1.2 The Challenge at Hand

In our study, we examine the feasibility of adapting instrumental classification methodologies for the processing of multi-label datasets. By incorporating acoustic features and additional labels, we anticipate improvements in the precision and reach of instrument classification. Our research approach involves the creation of a varied multi-label

dataset, extraction of pertinent acoustic features, and deployment of various classification algorithms. The spotlight here is on the efficiency of multi-label dataset classification.

2

Exploring Related Works

In this chapter, we delve deeper into various feature extraction methods, providing an introduction to crucial deep learning architectures significant for our work. Additionally, we will discuss multi-label classification, and provide an overview of the functionality of Bayesian optimization for hyper parameter tuning which we have employed in this work.

2.1 Methods of Feature Extraction

Mel Frequency Cepstral Coefficients (MFCCs) represent the short-term power spectrum of sound, derived from a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency. Due to their ability to capture timbral characteristics, MFCCs have been widely used as features in the arena of audio and speech processing, particularly in instrument recognition tasks [11].

The spectral centroid is a metric that showcases the "center of mass" for a sound's spectrum and is generally associated with the perceived brightness of a sound. It finds frequent use in music information retrieval (MIR) tasks and digital signal processing (DSP), specifically when classifying musical timbre [13].

Spectral bandwidth, the difference between the highest and the lowest frequencies in a continuous band, quantifies the spectral width of the sound. Owing to its capacity to capture the timbral texture of sounds, it's often employed in audio signal analysis and instrumental sound classification [23].

Spectral rolloff is the frequency below which a specific percentage of the total spectral energy lies. It is a feature commonly used in digital signal processing and MIR tasks,

helping differentiate between musical instruments by providing insights into the energy distribution of a sound [23].

The zero-crossing rate, the rate at which a signal changes from positive to zero to negative or vice versa, is widely used in both speech recognition and music information retrieval. It serves as a crucial distinguishing factor for different musical instruments [20].

2.2 Key Deep Learning Architectures

The Multilayer Perceptron (MLP) is a type of feedforward artificial neural network. It comprises at least three layers of nodes: an input layer, a hidden layer, and an output layer. Each node in one layer connects with a certain weight to every node in the following layer. MLP uses a supervised learning technique known as backpropagation for training the network. Its forte is in solving problems that are not linearly separable [7].

Convolutional Neural Networks (CNNs) are deep, feedforward artificial neural networks that excel at analyzing visual imagery. Thanks to their shared-weights architecture and translation invariance characteristics, they are also called shift invariant or space invariant artificial neural networks. CNNs have found their applications in image and video recognition, recommender systems, and natural language processing [10].

Support Vector Machine (SVM) is a powerful machine learning model. It is primarily used for classification and regression tasks, and it shines in handling high-dimensional data. The concept is simple yet effective: it finds the optimal hyperplane or set of hyperplanes in a high-dimensional space, which can be used to categorize new examples. At the heart of SVM is the principle of maximizing the margin to increase the model's generalization performance [5].

2.3 Multi-label Classification

In machine learning, Multi-label Classification (MLC) is a task where an instance can have multiple labels from a set [22]. This is a different ballgame from multi-class classification, where each instance gets assigned exactly one class or label [25]. MLC finds its applications in several areas like text categorization, music categorization, and gene functional prediction, where an object can simultaneously belong to several classes or carry multiple labels [21]. To tackle this problem, various strategies and algorithms have been proposed, including problem transformation methods and algorithm adaptation methods [25].

2.4 The Art of Optimization

Bayesian Search provides a hyper parameter optimization solution through Bayesian Optimization. The Bayesian Optimization is a sequential design strategy for global optimization of black-box functions that factors in past evaluation results when determining the next values to evaluate.

In the machine learning world, the black-box function is generally the generalization error of a learning algorithm with respect to the hyper parameter that is under optimization, and this function is unknown. The core concept behind Bayesian optimization is modeling this function using a Gaussian process, and then leveraging this model to identify the "optimal" hyper parameters for the next iteration [2].

3

Methodology

This chapter employs a combination of audio feature extraction techniques and machine learning models for in-depth audio signal analysis. We extract features like Mel Frequency Cepstral Coefficients (MFCC), Spectral Centroid, Spectral Rolloff, Spectral Bandwidth and Zero-Crossing Rate. We also define deep learning models like Multi Layer Perceptron and Support Vector Machine for multi-label classification tasks. Additionally, we utilize Bayesian search for hyper parameter tuning, and detail the dataset used for our experiments.

3.1 Mel Frequency Cepstral Coefficient

Mel Frequency Cepstral Coefficients (MFCCs) are a type of feature used for representing audio data and are particularly useful in speech and music recognition systems. The process of computing MFCCs involves several steps, each with its own associated formula.

In the first step the signal is framed into short frames. In this calculation we take a signal s and divide it into frames of length N.

In the next step we calculate for each frame the periodogram estimate of the power spectrum. Here the Fast Fourier Transform (FFT) is used to transform each frame from time domain to frequency domain, producing the periodogram estimate P_k for $k = 0, ..., N/2$:

$$P_k = \frac{1}{N}|FFT(s_k)|^2 \tag{3.1}$$

After that we apply the Mel filterbank to the power spectra, sum the energy in each filter. The Mel filterbank consists of M triangular filters, each represented as $H_m(k)$, for $m = 1, ..., M$. The filterbank output S_m is produced by multiplying P_k by $H_m(k)$, and summing

$$S_m = \sum_{k=0}^{N/2} H_m(k) P_k \tag{3.2}$$

Then we take the logarithm of all filterbank energies, formulated as

$$s_m = log(S_m) \tag{3.3}$$

In the last step wake the Discrete Cosine Transform (DCT) of the log filterbank energies:

$$MFCC_n = \sum_{m=0}^{M-1} s_m cos[\frac{\pi n}{M}(m - \frac{1}{2})](n = 0, ..., M - 1) \tag{3.4}$$

In these formulas, s represents the input signal, N is the number of samples in a frame, $FFT(s_k)$ represents the Fast Fourier Transform of the signal in the k^{th} frame, P_k is the periodogram estimate of the power spectrum, $H_m(k)$ represents the Mel filterbank, S_m is the filterbank output, s_m is the log filterbank energies, and $MFCC_n$ is the computed MFCC.

The output is a set of coefficients that collectively make up an MFCC, providing a compact representation of the sound [17].

3.2 Spectral Centroid Frequency

The Spectral Centroid is a measure used in digital signal processing to characterize a spectrum. It indicates where the "center of mass" for a sound's spectrum is located, and is commonly associated with the perceived "brightness" of a sound.

Mathematically, the spectral centroid (SC) for a given signal is calculated as follows:

$$SC = \frac{\sum_{n=0}^{N-1} f(n) \cdot |X(n)|}{\sum_{n=0}^{N-1} |X(n)|}$$

In this formula, $X(n)$ represents the magnitude of the Fourier transform at the frequency bin n, $f(n)$ is the frequency at bin n, and N is the total number of frequency bins. Essentially, the spectral centroid is computed as the weighted mean of the frequencies present in the signal, with their magnitudes as the weights.

The Spectral Centroid Frequency is just the frequency corresponding to the spectral centroid, i.e., the "center of mass" frequency, and can be obtained directly from the calculation of the spectral centroid.

It is important to note that to obtain a more perceptually relevant spectral centroid, it's common to convert the frequencies to a Mel scale or similar before computing the centroid.

These features are extensively used in music information retrieval and digital signal processing, specifically for classifying musical timbre [12].

3.3 Spectral Rolloff

The Spectral Rolloff is a measure used in digital signal processing to describe the shape of the power spectrum of a signal. It is defined as the frequency below which a specified percentage (typically between 85 $\%$ to 95 $\%$) of the total spectral energy is contained. The spectral rolloff can be an indicator of the presence of high-frequency noise or harmonics in a signal.

Mathematically, given a discrete Fourier transform $X(n)$ of a signal, and assuming $|X(n)|^2$ gives the power at frequency bin n, the spectral rolloff (R) is calculated as:

$$R = \min_n \left\{ f(n) : \sum_{k=0}^{n} |X(k)|^2 \geq \alpha \sum_{k=0}^{N-1} |X(k)|^2 \right\} \tag{3.5}$$

Where $f(n)$ is the frequency corresponding to bin n. And N is the total number of frequency bins. α is the specified percentage (e.g., 0.85 for 85 $\%$).

Here's a step-by-step breakdown of this formula:

First compute the power spectrum of the signal by squaring the magnitude of its Fourier transform: $|X(n)|^2$. After that calculate the total energy in the spectrum: $\sum_{k=0}^{N-1} |X(k)|^2$. Then multiply this total energy by the specified percentage α. Before the last step we have to start from the lowest frequency bin and accumulate the energy in each bin: $\sum_{k=0}^{n} |X(k)|^2$. In the end find the bin n where this accumulated energy just exceeds or equals the value from the step before. The frequency $f(n)$ corresponding to this bin is the spectral rolloff frequency.

The Spectral Rolloff is used in various audio processing applications, particularly in distinguishing between harmonic content and noise, and is often used in music information retrieval tasks to differentiate musical instruments or genres [24] [9].

3.4 Spectral Bandwidth

The spectral bandwidth is like a window showing the range of frequencies where you find most of the energy of a sound or signal. This is a important concept used in a bunch of areas.

Different fields might have slightly different definitions for spectral bandwidth, but a simple one is just the difference between the highest and lowest frequencies in a signal. If a signal has parts going from 300 Hz to 3 000 Hz, the spectral bandwidth is

$$3000Hz - 300Hz = 2700Hz \tag{3.6}$$

Another method is to compute the order-p spectral bandwidth. The order-p spectral bandwidth is a measure that describes the width of the power spectrum. It provides insights into the distribution of spectral content around the spectral centroid, and for different orders p, it highlights different moments of the spectral shape.

Given a discrete Fourier transform $X(n)$ of a signal, where $|X(n)|^2$ gives the power at frequency bin n, and the spectral centroid SC defined as:

$$SC = \frac{\sum_{n=0}^{N-1} f(n) \cdot |X(n)|}{\sum_{n=0}^{N-1} |X(n)|} \tag{3.7}$$

The order-p spectral bandwidth (BW_p) can be defined as:

$$BW_p = \left(\frac{\sum_{n=0}^{N-1} |f(n) - SC|^p \cdot |X(n)|}{\sum_{n=0}^{N-1} |X(n)|} \right)^{\frac{1}{p}} \tag{3.8}$$

Where $f(n)$ is the frequency corresponding to bin n. And N is the total number of frequency bins. Furthermore SC is the spectral centroid of the signal. In addition p is the order of the spectral bandwidth, and $p \geq 1$. Here's a brief understanding of the formula: First $f(n) - SC$ calculates the deviation of each frequency bin from the spectral centroid. This deviation is raised to the power p, which means for $p = 1$, it's a linear deviation, for $p = 2$, it's squared, and so forth. The resultant is multiplied by the magnitude of the Fourier transform to weight the deviation by the spectral content at each frequency. Then the summation across all frequencies gives a cumulative weighted deviation. Finally, taking the p-th root of this summation provides an averaged deviation in the same unit as frequency, giving the bandwidth.

The order-p spectral bandwidth gives insight into different moments of the spectral shape. For instance, BW_1 might be more sensitive to broad spectral shapes, while BW_2 might be more influenced by sharp peaks or troughs.

This measure is useful in audio signal analysis, particularly in characterizing timbral texture and identifying different musical instruments or sound types [12].

3.5 Zero-Crossing Rate

We define the zero-crossing rate (ZCR) as the number of times a signal changes from positive to negative or vice versa within a certain part of the signal, divided by the total number of samples in that part. We had to consider two types of noise that could

mess things up. One of these comes from very low-frequency notes from overlapping instruments, like a bass guitar. Since the signal we are working with is really short (usually less than 100 milliseconds), these low notes can mess up the average level of the signal. The other type of noise comes from high-frequency components of other instruments. Even though these sounds are usually quieter than the main percussive sound, they can still interfere with the ZCR calculation [6]. So the ZCR is calculated by checking if the signal changes from positive to negative:

$$ZCR = \frac{1}{N-1} \sum_{n=1}^{N-1} \mathbb{1}(s[n] \times s[n-1] < 0) \tag{3.9}$$

where N is the total number of samples in the signal. And $s[n]$ is the signal value at sample n. In addition $\mathbb{1}$ is the indicator function, which equals one if its argument is true and zero otherwise. The formula can be interpreted as follows the product $s[n] \times s[n-1]$ will be negative if and only if the signal has crossed zero between samples $n-1$ and n. The sum counts the number of zero-crossings in the signal. Finally, dividing by $N-1$ normalizes the count based on the length of the signal, resulting in a rate of zero-crossings. For continuous signals, the ZCR can be defined similarly by integrating over the duration of the signal and checking for sign changes. In many applications, especially in speech processing, ZCR serves as a simple measure for detecting voiced versus unvoiced speech segments, as unvoiced segments tend to have a higher ZCR compared to voiced segments. It's also employed in music information retrieval for understanding the nature of musical instruments or sound types [20]. ZCR can even be used as a simple way to figure out the pitch of a single-note sound. It is also used in voice activity detection, or VAD, which is a way to tell if someone's talking in a certain part of audio [19].

3.6 Deep Learning Architectures

3.6.1 Multi Layer Perceptron

We will look at a general multilayer perceptron (MLP) unit with l layers. Building a perceptron like this, with any number of layers, is just repeating what has to be done for one and two layers over and over. In simple terms, to build this unit, we combine the results of the $l-1$ layer units in a certain way, then run the combined result through an activation function.

A Multi-Layer Perceptron (MLP) is a feedforward artificial neural network, made up of multiple layers of nodes or neurons. Mathematically, each neuron processes an input to produce an output, often using a non-linear activation function. Let's break down the MLP mathematically. Each neuron receives inputs, multiplies each input by a corresponding weight, and then sums them up. If we consider the i-th neuron in layer l, its weighted sum $z_i^{(l)}$ is

$$z_i^{(l)} = \sum_j w_{ij}^{(l)} x_j^{(l-1)} + b_i^{(l)} \tag{3.10}$$

where $x_j^{(l-1)}$ is the output from the j-th neuron in layer $l-1$. So $w_{ij}^{(l)}$ is the weight connecting the j-th neuron from layer $l-1$ to the i-th neuron in layer l. In addition $b_i^{(l)}$ is the bias of the i-th neuron in layer l. It also consists of the activation function which means, once the weighted sum is calculated it is passed through an activation function to produce the neuron's output. The activation function introduces non-linearity into the model, which allows the network to capture complex patterns. A common activation function is the sigmoid function:

$$f(z) = \sigma(z) = \frac{1}{1+e^{-z}} \tag{3.11}$$

However, there are several others like ReLU (Rectified Linear Unit), tanh, etc. The output $a_i^{(l)}$ of the i-th neuron in layer l is:

$$a_i^{(l)} = f(z_i^{(l)}) \tag{3.12}$$

Where f is the activation function and $z_i^{(l)}$ is the weighted sum for the neuron. MLPs are typically trained using a process called backpropagation, which is an application of the chain rule from calculus to compute gradients of the loss function with respect to the weights and biases. These gradients are then used in optimization algorithms like gradient descent to adjust the weights and biases to minimize the loss.

For regression tasks, Mean Squared Error (MSE) might be used:

$$L(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{3.13}$$

Where y is the true output. And $\hat{y}$ is the predicted output by the MLP. Also n is the number of samples. For classification tasks, Cross-Entropy loss might be used:

$$L(y, \hat{y}) = -\sum_{i} y_i \log(\hat{y}_i) \tag{3.14}$$

Using the chain rule, the gradient of the loss with respect to a weight can be calculated as:

$$\frac{\partial L}{\partial w_{ij}^{(l)}} = \frac{\partial L}{\partial a_i^{(l)}} \times \frac{\partial a_i^{(l)}}{\partial z_i^{(l)}} \times \frac{\partial z_i^{(l)}}{\partial w_{ij}^{(l)}} \tag{3.15}$$

This process is repeated for every weight and bias in the network [17].
A typical architecture for this can be seen in Figure 3.1.

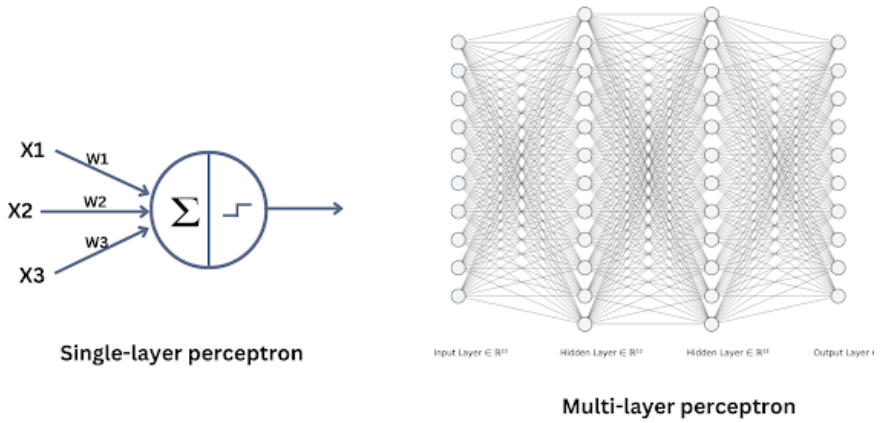

Figure 3.1: Single-layer and Multi-layer perceptron from [1]

3.6.2 Support Vector Machine Learning

Support Vector Machines (SVM) are a set of supervised learning algorithms used for classification and regression. We focus on the use of SVM for binary classification, as it is the classic scenario. The basic idea is, that SVMs aim to find the best hyperplane that separates data into two classes. The best hyperplane is the one that maximizes the margin between the two classes. Support vectors are the data points that are closest to this hyperplane. For a linearly separable dataset, we can define the hyperplane as

$$w \cdot x + b = 0 \tag{3.16}$$

where w is the weight vector, x is the input vector, and b is the bias. The decision function to classify an input x is:

$$f(x) = sign(w \cdot x + b) \tag{3.17}$$

For the two classes (+1 and -1), the equations of the hyperplanes are

$$w \cdot x + b = 1 \tag{3.18}$$

for the positive class, and

$$w \cdot x + b = -1 \tag{3.19}$$

for the negative class. The margin m between these hyperplanes is

$$m = \frac{2}{\|w\|}. \tag{3.20}$$

The objective is to maximize this margin, which is equivalent to minimizing $\|w\|$. Therefore, the optimization problem is:

$$\min \quad \frac{1}{2}\|w\|^2 \quad \text{subject to} \quad y_i(w \cdot x_i + b) \geq 1 \quad \text{for} \quad i = 1, ..., n \tag{3.21}$$

Where y_i is the class label of the i-th data point (either +1 or -1). And n is the number of training data points.

For non-linearly separable datasets, we introduce slack variables ξ_i to allow for misclassifications:

$$\min \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \quad \text{subject to} \quad y_i(w \cdot x_i + b) \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0 \quad \text{for} \quad i = 1, ..., n \tag{3.22}$$

Here C is a regularization parameter that determines the trade-off between maximizing the margin and minimizing classification error. And ξ_i measures the degree of misclassification for the i-th data point.

In cases where the data is not linearly separable, SVM can be extended using the kernel trick. Instead of finding hyperplanes in the original feature space, we implicitly map the input data into a higher-dimensional space using a kernel function and then find a separating hyperplane in this new space.

Popular kernel functions include

1. Linear Kernel: $K(x, x') = x \cdot x'$
2. Polynomial Kernel: $K(x, x') = (1 + x \cdot x')^d$
3. Radial Basis Function (RBF) Kernel: $K(x, x') = \exp(-\gamma \|x - x'\|^2)$
4. Sigmoid Kernel: $K(x, x') = \tanh(\alpha x \cdot x' + c)$

Where d is the degree for the polynomial kernel. And γ is a parameter for the RBF kernel. Also α and c are parameters for the sigmoid kernel.

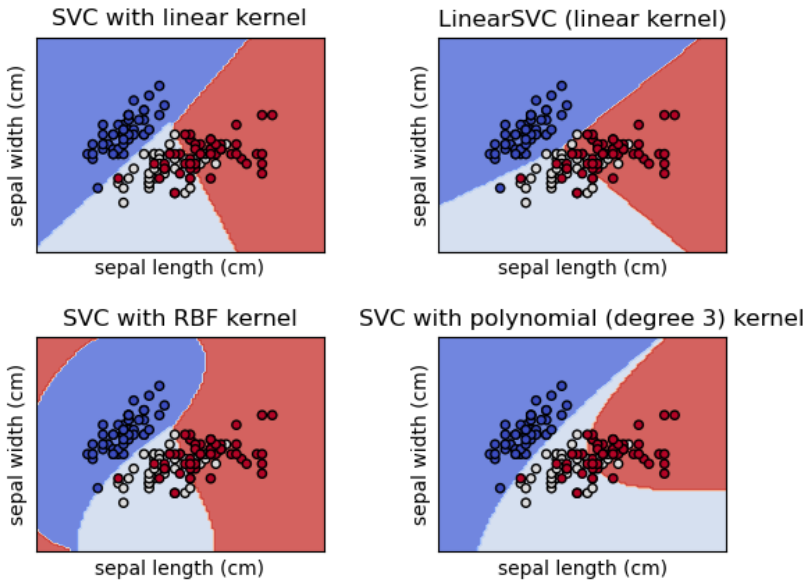

Figure 3.2: Support Vector Machine

With the kernel trick, the optimization becomes a dual problem, which can be solved in terms of dot products in the transformed space.

3.7 Multi-label Classification

Multi-label classification refers to the classification problem where a subset of all labels $\mathcal{Y}_i \subset Y = \{y_1, y_2, ..., y_n\}$ can be attributed to each sample $x \in X = \mathcal{R}^d$. For example

multiple topics which can be attributed to a news article or as in our case multiple instruments that play in one audio clip [26]. There are several methods that can be used for multi-label classification. The binary method is an intuitive method which assumes the independence of the labels and trains a binary classifier for each label.

Classifier Chains [16] use the binary method by linking all the single classifiers in a chain and using the labels of the previous classifiers in the chain as input along with the usual feature input. Therefore it assumes dependence of the labels in the chain to the previous labels. Since the ideal order of the chain is unknown, multiple chains are trained to determine the best order. The output of the first classifier is determined with

$$C_1(x_1, ..X_n) = y_1 \tag{3.23}$$

and the i-th classifier C_i in the chain can be described as

$$C_i(x_1, ..., x_n, y_1, ..y_{i-1}) = y_i. \tag{3.24}$$

3.8 Metrics

For measuring the quality of our classification, we use three metrics. We can judge the performance of our models.

Precision, also referred to as Confidence, shows us the proportion of predicted positive cases that are genuinely positive. This factor is quite significant in fields such as Machine Learning, Data Mining, and Information Retrieval, as it reflects the accuracy of our models positive predictions. The formula for Precision is [14] [15]:

$$Precision = \frac{TruePositives(TP)}{(TruePositives(TP) + FalsePositives(FP))} \tag{3.25}$$

On the flip side, the Recall, or referred as sensitivity, measures the ratio of actual positive cases that are predicted correctly as positive. Essentially, it is a way to see how well the model can identify true positive instances. While this is not always highly regarded in fields like Information Retrieval, it can be a vital tool in certain contexts such as Medicine,

where missing a positive case can have serious consequences [14]. In other words, the Recall value gives us an idea of the number of positive cases our model is able to catch. It is calculated by dividing the number of true positives by the sum of true positives and false negatives. This gives us:

$$Recall = \frac{TruePositives(TP)}{(TruePositives(TP) + FalseNegatives(FN))} \tag{3.26}$$

Both Recall and Precision focus solely on positive cases and predictions, giving us insights into the types and rates of errors made by the model. However, they do not provide any information on how well the model identifies negative cases. True Negatives are not considered in either of these measurements, nor in their means [14].

The F1 score, for instance, provides a mean of Recall and Precision, but references the True Positives to the Arithmetic Mean of Predicted Positives and Real Positives. Meanwhile, the Geometric Mean of Recall and Precision normalizes True Positives to the Geometric Mean of Predicted Positives and Real Positives [14]. It is important to note, however, that these measurements do not provide insights into how well the model handles negative cases. This leads to [15]:

$$F1 = \frac{2 \cdot (Precision \cdot Recall)}{(Precision + Recall)} \tag{3.27}$$

3.9 Bayesian Optimization

The bayesian optimization uses previous attempts to make educated guesses about what to try next. This process could save time and resources. Initially, it is trying to optimize a function $f(x)$. We do not know the perfect shape or structure of this function, but have seen its behavior at different points. It then constructs a mental model $P(f|D)$. This model encapsulates it is current understanding of the function based on the data D observed so far. To start, your it might assume a general pattern with an average behavior of zero and some predetermined variability. The next move is figuring out the next setting to try, which is done by optimizing a decision-making function $a(x|D)$, which

comes from $P(f|D)$. This decision-making function represents how much it expects to improve the model based on what it know so far. It is like weighing between taking a wild guess (sampling from areas in the setting space where the uncertainty is high) and going with a hunch (sampling from areas your friend thinks will yield high results). A common strategy your friend might use for this decision-making function is the Expected Improvement (EI) function. Once it chooses a setting and sees the result, they update their mental model $P(f|D)$ and repeat the process until a certain stopping condition is met, such as a maximum number of trials [4].

Moreover, we incorporate cross-validation during our employment of Bayesian optimization. This method can estimate the skill of machine learning models. It aims to test the ability of the model to predict new data that was not used in estimating it, in order to flag problems like overfitting or selection bias and to give an insight on how the model will generalize to an independent dataset. In simple terms, it works by dividing the dataset into a number of k folds. The model is trained on $k-1$ of these folds, and then tested on the remaining fold. This process is repeated k times, with each fold used exactly once as the test set.
The k results can then be averaged to produce a single estimation of the model performance.

3.10 Dataset

In this work we are using the Instrument Recognition in Musical Audio Signals (IRMAS) dataset. IRMAS is a dataset that is commonly used for the task of musical instrument recognition. It consists of 6705 short audio clips, each 3 seconds long, extracted from 2000 digital audio recordings with a sample rate of 44.1 kHz, originally recorded in professional studio conditions. The IRMAS dataset is designed to be used for training and testing methods for the automatic recognition of predominant instruments in polyphonic music. It covers a diverse range of musical genres, including classical, jazz, pop, rock, and others. The dataset is divided into a training set and a test set. The training set contains 6705 excerpts from distinct music pieces, each annotated with the predominant

instrument class, including piano, guitar, voice, strings, drums, and others. The audio clips in the training set are divided into "solo" (one instrument) and "ensemble" (more than one instrument, one being predominant) subsets. The test set, on the other hand, contains 2874 excerpts from 290 distinct music pieces and does not have public class annotations, as it is intended for evaluation purposes. Each audio clip in the dataset is mono, 16-bit depth, and lasts 3 seconds. The dataset also includes additional metadata about the audio clips, such as the instruments present in the clip, the starting time of the clip in the original recording, and others [3].

Instruments	Occurences
Cello	111
Clarinet	62
Flute	163
Guitar	535
E-Guitar	942
Organ	361
Piano	995
Saxophone	326
Trumpet	167
Violin	211
Voice	1044

Table 3.1: All the instruments and the amount of occurences of each instrument in the multi-label dataset.

4

Experiments and Results

4.1 Procedure

We loaded three different datasets along with their corresponding labels. These are then concatenated to form a single training set. The features are standardized. It reads audio files from a directory, and for each file, it extracts certain features including Mel-frequency cepstral coefficients (MFCC), spectral rolloff, spectral bandwidth, spectral centroid, and zero-crossing rate. These features are all combined into a single feature vector, which will be the input for the classifier. The output or the label for each audio file is a binary vector where the index corresponding to the instrument type is set to 1 and all others are set to 0. Before training, Bayesian hyperparameter optimization is performed on the binary classifier using the BayesSearchCV method from the scikit-optimize package. This is done to find the best hyperparameters that optimize model performance.

All the classifiers in the classifier chain are then built using the same previously determined hyperparameters and then they are trained with the training data. We build 10 classifier chains, each with a different random state for the order of the labels. The trained classifier chains are used to predict the labels for the test set. The predictions from all chains are averaged to get an ensemble prediction. Metrics such as the precision, recall and f1-score are computed to evaluate the performance of the classifier ensemble prediction.

This procedure is done twice with different classifiers. We use a binary MLP classifier and a binary SVM classifier where the optimal kernel is determined as a hyper parameter. The results for this experiment can be seen in Table 4.1.

	MLP			SVM		
Instruments	P	R	F1	P	R	F1
Cello	0.61	0.38	0.47	0.50	0.32	0.39
Clarinet	0.38	0.15	0.21	0.40	0.25	0.31
Flute	0.69	0.56	0.62	0.31	0.12	0.18
Acoustic Guitar	0.67	0.45	0.54	0.51	0.34	0.41
Electric Guitar	0.66	0.52	0.58	0.61	0.43	0.50
Organ	0.61	0.43	0.50	0.56	0.17	0.26
Piano	0.79	**0.59**	**0.68**	**0.71**	**0.68**	**0.70**
Saxophone	0.82	0.43	0.56	0.64	0.28	0.39
Trumpet	0.85	0.35	0.50	0.37	0.18	0.24
Violin	**0.86**	0.49	0.62	0.62	0.27	0.38
Voice	0.66	0.52	0.58	0.66	0.36	0.47

Table 4.1: Results for Multi-Label Classification

We also used similar models as mentioned above to classify instruments in audio files where only one instrument was present. For this we used the same features as for the multi-label classification experiment. This time the entire multi-label dataset will be used for training and the single label dataset for testing. We once reused the model with classifier chains and once we simply built a separate classifier for each instrument. The better result out of those experiment can be seen in Table 4.2.

	SVM			MLP		
Instruments	P	R	F1	P	R	F1
Cello	0.09	0.04	0.06	**0.50**	0.003	0.005
Clarinet	**0.18**	0.03	0.04	0.08	0.99	0.14
Flute	0.05	0.03	0.04	0	0	0
Acoustic Guitar	0.13	0.27	0.18	0.09	0.95	0.17
Electric Guitar	0.13	0.41	**0.20**	0	0	0
Organ	0.12	0.15	0.13	0	0	0
Piano	0.12	0.33	0.17	0	0	0
Saxophone	0.12	0.10	0.10	0.09	**1**	0.17
Trumpet	0.10	0.04	0.06	0.10	0.20	0.13
Violin	0.11	0.07	0.08	0	0	0
Voice	0.13	**0.46**	**0.20**	0.12	**1**	**0.21**

Table 4.2: Results for single instrument classification

4.2 Results

In the next step we will interpret the results and think about possible reasons. To visualize the results better, we can plot the results, too. In this work we only plotted the results of multi label classification for SVM and MLP because the visualization of these results is more meaningful than of single instrument classification, since here often values of 0 or very low generally. With this we obtained this plot:

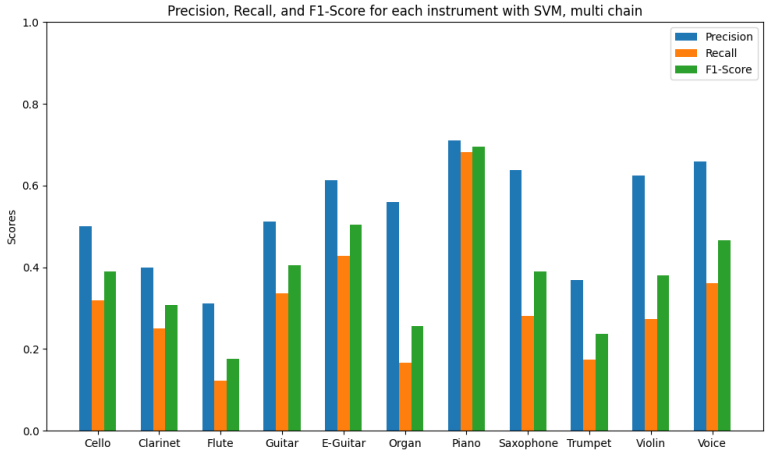

Figure 4.1: Scores SVM with multi label classification

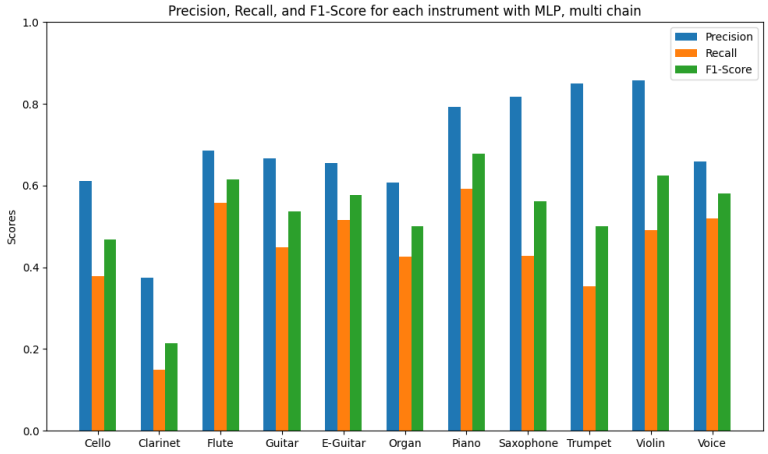

Figure 4.2: Scores MLP with multi label classification

In the upper part of the plot are scores SVM with multi label classification. The lower part of the plot shows scores MLP with multi label classification. In general, we can see

that the precision is always higher than the recall value for both parts. And for MLP the values are generally higher than for SVM. In addition you can see that the results vary for the different instruments. Both models achieve the highest F1 score for piano.

5
Discussion

5.1 Evaluating the Results

The main takeaway of our work is that multi-label classification works best with an MLP classifier and a classifier chain ensemble. It outperforms the SVM classifier on almost every instrument as can be seen in the figures 4.2 and in table 4.1. The reason for that lies most likely in the nature of the dataset and its complexity. We assume the dataset and the relationship between the labels is nonlinear and of complex nature, since it is even sometimes hard for humans to determine which instruments are playing at once. The use of the classifier chain could also be a factor since MLP is more flexible and it is easier to add another input feature while still keeping the pre-trained model. The optimization of the hyper parameters also plays a factor since it is generally harder to optimize SVM parameters and we used a stochastic approach which does not cover all variations of parameters and therefore could miss the optimal configuration.

For all results of the multi-label classification the precision score is higher than the recall score. This suggests that the classifiers in general are a little too conservative and that we haven't found an optimal threshold. Since this is consistent through both classifiers it could also be an effect of the classifier chain where false negatives of other classes impact the decision of classes later in the chain.

Another trend that is visible in our results for multi-label classification, is that underrepresented instruments in the dataset are harder to detect and therefore have lower overall scores than to other instruments. This can be seen for example for the clarinet which has the fewest occurrences in the dataset and also the lowest scores when using the MLP classifier. Since a separate classifier is trained for each instrument, the training data

for these particular instruments is not sufficient which leads to overfitting and a higher sensitivity to outliers. The difference between the instruments can also be caused by the nature of the instruments themselves since some instruments have a very clean sound and some have multiple harmonics and overtones, leading to more complex features.

Furthermore, it is important to emphasize that the parameters for the classifier chain were optimized for the chain as a whole and not for each individual classifier. This could potentially lead to suboptimal performances for some classifiers within the chain. This approach was chosen to ensure a consistent and streamlined optimization process for the classifier chain, but it might not be the most suitable method for every individual classifier. In addition optimizing for each classifier separately would result in better overall performance. However, doing so would considerably increase the computational load and potentially introduce new challenges, such as how to ensure that all classifiers in the chain work harmoniously after individual optimizations.

Our experiments for single-label classification were worse than for multi-label classification and mostly unusable. There are multiple possible reasons for this. First of all, the data used for testing is completely different data than the data used for training. The distribution of the instruments is different and obviously every clip only has one instrument. Although the results are mostly unusable, they suggest that there is a strong interdependence between the different instruments and that the use of classifier chains was therefore the right choice.

It is also worth noting that the high scores for precision and recall in Table 4.2 do not mean good results since the other corresponding score is low which means the classifier classifies the instrument to always positive or always negative.

5.2 Future Work

For future work, it would be worth investigating the benefits and trade-offs of optimizing parameters for each classifier within the chain separately. This investigation could reveal if the potential gains in the performance outweigh the increased complexity and computational demands. Furthermore, considering different ensemble methods and

using other models could help in achieving better results, especially for underrepresented instruments.

Since our results for the single-label classification were unusable, a future work could also investigate different methods for single-label classification with multi-label training data.

6

Conclusion

In conclusion our experiments have shown promising results for multi-label classification of instruments. The MLP model in connection with a classifier chain ensemble is a viable approach for the classification of instruments and its results are on par and even slightly better than other experiments that used different approaches. The model that we use does not need a lot of preprocessing of the dataset and it can be easily adapted to other domains.

Bibliography

[1] Multi-Layer Perceptron Explained: A Beginner's Guide — pycodemates.com. https://www.pycodemates.com/2023/01/multi-layer-perceptron-a-complete-overview.html#google_vignette. [Accessed 16-Jul-2023].

[2] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[3] J.J. Bosch, Jordi Janer, Ferdinand Fuhrmann, and Perfecto Herrera. A comparison of sound segregation techniques for predominant instrument recognition in musical audio signals. *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012*, pages 559–564, 01 2012.

[4] Eric Brochu, Vlad M. Cora, and Nando de Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning, 2010.

[5] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[6] Fabien Gouyon, Francois Pachet, and Olivier Delerue. On the use of zero-crossing rate for an application of classification of percussive sounds. 08 2002.

[7] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[8] Perfecto Herrera, Geoffroy Peeters, and Shlomo Dubnov. Automatic classification of musical instrument sounds. *Journal of New Music Research*, 32, 08 2010.

[9] Anssi Klapuri and Manuel Davy. *Signal Processing Methods for Music Transcription*. 01 2006.

[10] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[11] Beth Logan. Mel frequency cepstral coefficients for music modeling. *Proc. 1st Int. Symposium Music Information Retrieval*, 11 2000.

[12] Geoffroy Peeters. A large set of audio features for sound description (similarity and classification) in the CUIDADO project. Technical report, Icram, 2004.

[13] Geoffroy Peeters, Bruno Giordano, P. Susini, Nicolas Misdariis, and Stephen Mcadams. The timbre toolbox: Extracting audio descriptors from musical signals. *The Journal of the Acoustical Society of America*, 130:2902–16, 11 2011.

[14] D. M. W. Powers. Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation. *Journal of Machine Learning Technologies*, 2(1):37–63, 2011.

[15] Karthikeya Racharla, Vineet Kumar, Chaudhari Bhushan Jayant, Ankit Khairkar, and Paturu Harish. Predominant musical instrument classification based on spectral features. In *2020 7th International Conference on Signal Processing and Integrated Networks (SPIN)*, pages 617–622, 2020.

[16] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. *Mach. Learn.*, 85(3):333–359, 2011.

[17] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[18] Markus Schedl, Emilia Gómez, and Julián Urbano. *Music Information Retrieval: Recent Developments and Applications*. 01 2014.

[19] Drishti Sharma. Analysis of Zero Crossing Rates of Different Music Genre Tracks — analyticsvidhya.com. https://www.analyticsvidhya.com/blog/2022/01/analysis-of-zero-crossing-rates-of-different-music-genre-tracks/. [Accessed 16-Jul-2023].

[20] J. Sueur, T. Aubin, and C. Simonis. Seewave: a free modular tool for sound analysis and synthesis. *Bioacoustics*, 18:213–226, 2008.

[21] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3:1–13, 09 2009.

[22] Grigorios Tsoumakas, Ioannis Katakis, and I. Vlahavas. *Mining Multi-label Data*, pages 667–685. 07 2010.

[23] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10:293 – 302, 08 2002.

[24] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.

[25] Min-Ling Zhang and Zhi-Hua Zhou. A review on multi-label learning algorithms. *Knowledge and Data Engineering, IEEE Transactions on*, 26:1819–1837, 08 2014.

[26] Min-Ling Zhang and Zhi-Hua Zhou. A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1819–1837, 2014.

List of Figures

3.1 Single-layer and Multi-layer perceptron from [1] 14
3.2 Support Vector Machine 16

4.1 Scores SVM with multi label classification 23
4.2 Scores MLP with multi label classification 23

List of Tables

3.1 All the instruments and the amount of occurences of each instrument in the multi-label dataset. 20

4.1 Results for Multi-Label Classification 22

4.2 Results for single instrument classification 22

Name: Jonathan Wunden , Tanja Zast Matriculation number: 1063956 , 935593

Honesty disclaimer

I hereby affirm that I wrote this thesis independently and that I did not use any other sources or tools than the ones specified.

Ulm, 11.08.2023 .....

Jonathan Wunden , Tanja Zast