UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Computer Science Curriculum

Villem Tõnisson

Automatic Description of Music in Natural Estonian

Master's Thesis (30 ECTS)

Supervisor: Sven Aller, MSc

Tartu 2021

Automatic Description of Music in Natural Estonian

Abstract:

Recording and creating music is easier than ever and thanks to the internet, much of this music is publicly available. Due to the difficulty and time consumption of describing songs manually, automatic systems are necessary that could do it for us. The field of music information retrieval (MIR) can help us with this problem. MIR focuses on extracting useful features from music. Most of these features are not usable directly by humans. Natural language descriptors that are created based on these features could provide users with easily understandable ways to receive information about music.

We provide an overview of several tasks in the field of MIR. Our research shows that neural networks are state-of-the-art for almost all looked at tasks in music information retrieval. As a result of this thesis a web application was created that performs tempo estimation, chord recognition, key recognition, form discovery, genre classification and instrument recognition. Based on the retrieved information a description in natural Estonian is generated for the user. We found that the only feasible way to create descriptions was by using templates. For fully automatic description generation, more descriptive textual data about music is necessary.

Keywords:

Music information retrieval, Web application, Python, Music information retrieval modules

CERCS: P175

Muusikateose automaatne kirjeldamine loomulikus eesti keeles

Lühikokkuvõte:

Muusika salvestamine ja loomine on lihtsam kui iial varem ja tänu internetile on palju sellest muusikast vabalt kättesaadav. Laulude kirjeldamine käsitsi on keeruline ja aegavõttev. Selle tõttu on vaja automaatseid süsteeme, mis seda meie eest teeks. Sellega saab meid aidata muusika infootsingu uurimisvaldkond. Muusika infootsing keskendub muusikast kasulike tunnuste kätte saamisele. Enamik neist tunnustest ei ole otseselt inimeste poolt kasutatavad. Loomulikus keeles kirjeldused muusika kohta võivad kasutajatele pakkuda lihtsasti arusaadavat viisi muusika kohta infot saada.

Me anname ülevaate mitmest probleemist ja nende lahendustest muusika infootsingu valdkonnas. Meie uurimustöö näitab, et närvivõrgud on kõige kaasaegsemad lahendused peaaegu kõigile uuritud probleemidele. Selle töö tulemusena valmis veebirakendus, mis teostab järgmisi ülesandeid: tempo hindamine, akordide tuvastus, helistiku tuvastus, muusikateose vormi leidmine, žanri tuvastamine, instrumentide tuvastamine. Leitud info põhjal koostati kasutajale kirjeldus loomulikus eesti keeles. Ainus teostatav lahendus kirjelduste loomiseks oli mallide kasutamine. Täisautomaatseks kirjelduste loomiseks

oleks vaja rohkem lähteandmeid, mis kirjeldavad tekstiliselt muusikat.

Võtmesõnad:
Muusika infootsing, Veebirakendus, Python, Muusika infootsingu moodulid

CERCS: P175

Contents

1 Introduction **6**

2 Music information retrieval **8**
2.1 What is music information retrieval? 8
2.2 Definitions and explanations 8
2.3 Performed tasks 10
2.3.1 Tempo estimation 10
2.3.2 Key detection 11
2.3.3 Form discovery 12
2.3.4 Genre classification 14
2.3.5 Chord recognition 17
2.3.6 Instrument recognition 17

3 Natural Estonian descriptions **21**
3.1 Music description 21
3.2 Description generation 22

4 Used modules and frameworks **24**
4.1 Selection of programming language 24
4.2 Module selection for MIR 24
4.2.1 Aubio 24
4.2.2 Essentia 2.0 25
4.2.3 Librosa 25
4.2.4 Madmom 25
4.2.5 Marsyas 26
4.2.6 YAAFE 26
4.2.7 Final module choice 26
4.3 Module Selection for Web Application 28
4.3.1 Django 28
4.3.2 Flask 28
4.3.3 Final module choice 28
4.4 Docker 29

5 Implementation specifics **30**
5.1 Docker 30
5.2 Web application architecture 30
5.3 Music information retrieval 31

6 Testing **33**

7 Conclusion **36**

References **43**

Appendix **44**
I. Glossary 44
II. Licence 45

1 Introduction

With increasing quality and quantity of audio recording hardware and the spread of digital audio workstations, it is easier than ever to create and save music. This music is being uploaded to streaming services such as SoundCloud or Spotify. Due to the difficulty and time consumption of labelling and metadata gathering for all songs manually, automatic systems are necessary for companies such as Spotify, which have a library of millions of unique songs that is increasing in size. Methods are necessary for retrieving information from audio. There are also applications like Capo [Sup21], which help the user detect the beats and chords of songs to simplify learning it.

The automatic description of music in natural language includes two tasks: music information retrieval and description generation. Computational music information retrieval approaches typically use features and create models to describe music [SGGU14]. These features are rarely usable directly by humans. It would be good for listeners to have easily readable descriptions for songs that could be created automatically.

In [SGGU14] the researchers state that music information retrieval as a research field is driven by a set of core applications:

- Music retrieval applications are intended to help users find music in large collections by a particular similarity criterion.
- Music recommendation systems typically propose a list of music pieces based on modeling the user's musical preferences.
- Automatic music playlist generation which has the aim of creating an ordered list of results, such as music tracks or artists, to provide meaningful playlists enjoyable by the listener.
- Intelligent user interfaces that support the user in experiencing serendipitous listening encounters.

There are also applications which fall under multiple categories from this list. Automatic descriptions of music in natural language could be used for music retrieval applications. These descriptions could provide the users with more easily digestible information than simply using a list of tags.

In this thesis we give an overview of approaches for some tasks in the field of music information retrieval. In addition to that we also give an overview of music information retrieval modules usable in Python. We created a web application which can, given as input a music file, automatically produce a description for it in natural Estonian. Automatic description of music in natural language has not been done previously to this extent, according to our research. We provide the source code and pre-trained models

used in the final version of the web application. The code is modular and one could easily use parts of the program in their own work.

First, in Section 2 we give a quick overview of the field of music information retrieval. We also offer a thorough overview of some tasks in the field of music information retrieval. This section also includes our approaches to these tasks. In Section 3 we look at how music has been described historically and how it is described in the field of music information retrieval. We also detail how the descriptions for music were created in this thesis. In Section 4 we provide an overview of music information retrieval modules usable in Python and explain our module choice. We also provide reasoning for our selection of framework for web application creation. Finally, we provide implementation specifics about the web application and music information retrieval in Section 5. We try out our approaches and evaluate them in Section 6.

2 Music information retrieval

This section gives an overview of the field of music information retrieval and how it is used within this thesis.

2.1 What is music information retrieval?

Music is a pervasive topic in our society as almost everyone enjoys listening to it and many also create. The research field of Music Information Retrieval (MIR) is foremost concerned with the extraction and inference of meaningful features from music (from the audio signal, symbolic representation or external sources such as web pages), indexing of music using these features, and the development of different search and retrieval schemes (for instance, content-based search, music recommendation systems, or user interfaces for browsing large music collections) [SGGU14].

The field consists of tasks of varying difficulty. For example, we could easily find the loudest part of a song looking directly at the waveform of the song. It is much more difficult to classify a song into a genre given only its waveform. This task requires extensive feature extraction. There are objective measures such as the tempo or the key of songs. There are also more subjective tasks such as recognizing the genre or the mood of songs. All of this information could be used to group together similar songs. Streaming services could use this information to recommend new songs to listeners. Some of these tasks have been attempted in this thesis and are outlined in section 2.3.

2.2 Definitions and explanations

In this subsection we provide definitions and explanations for some more common approaches or terms in the field of music information retrieval.

How is audio saved? An analog audio signal is an electrical waveform which is a representation of the velocity of a microphone diaphragm. It is a two-dimensional signal which carries a voltage change with respect to time. Digital audio is an alternative means of carrying an audio waveform. This is generally achieved by using pulse code modulation (PCM). Instead of being continuous, the time axis is represented in a discrete or stepwise manner. This process is called sampling and the frequency with which samples are taken is called sampling rate. Each sample is represented by a discrete number [Wat02]. A waveform of Highway to Hell by AC/DC can be seen in Figure 1.

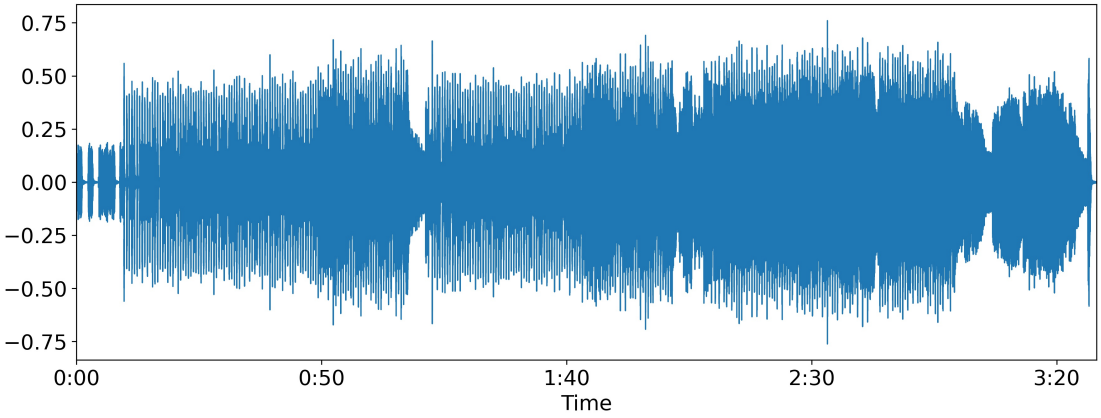

Figure 1. The waveform of Highway to Hell by AC/DC

Discrete Fourier transform (DFT) Audio signals are recorded and listened to in the time domain; they contain the intensity of the sound as a function of time. However, pitches and therefore chords are represented by frequencies. The Discrete Fourier transform is used to convert the time-domain input signal into a frequency-domain representation which can be analyzed for intensities of specific pitches. [LN14]

The most commonly used method for this is the fast Fourier transform (FFT). FFT is so common, that DFT and FFT are sometimes used interchangeably. There are also other ways to transform the waveform into the frequency-domain such as constant-Q transform.

Mel frequency scale The Mel scale is based on a mapping between actual frequency and perceived pitch as the human auditory system does not perceive pitch in a linear matter. The mapping is approximately linear below 1kHZ and logarithmic above. [L$^+$00] Mel frequency is commonly used in research as it is more similar to how people perceive pitch.

It was first proposed in [SVN37] after conducting experiments where subjects had were given two tones one of which they could change. They were then asked to change the lower tone until it was perceived as half of the frequency of the other tone.

Spectrogram A spectrogram is a representation of the frequency spectrum with respect to time. It is created by choosing a window size in which the waveform is converted to the frequency spectrum. The x-axis represents the time and the y-axis represents the frequency. The increments on the x-axis are the windows in which the transformation

was completed. The y-axis is typically divided into some amount of frequency ranges. A mel-spectrogram is a spectrogram where the y-axis is represented in mels compared to the usual hertz.

Figure 2 contains a power spectrogram of Under Pressure by Queen. This shows the volume at which each frequency was playing during each window. This is an example of a log-frequency spectrogram. The analysis was completed with a window size of 2048 and a sampling rate of 22050 Hz. The first few bass notes are clearly visible from this spectrogram.

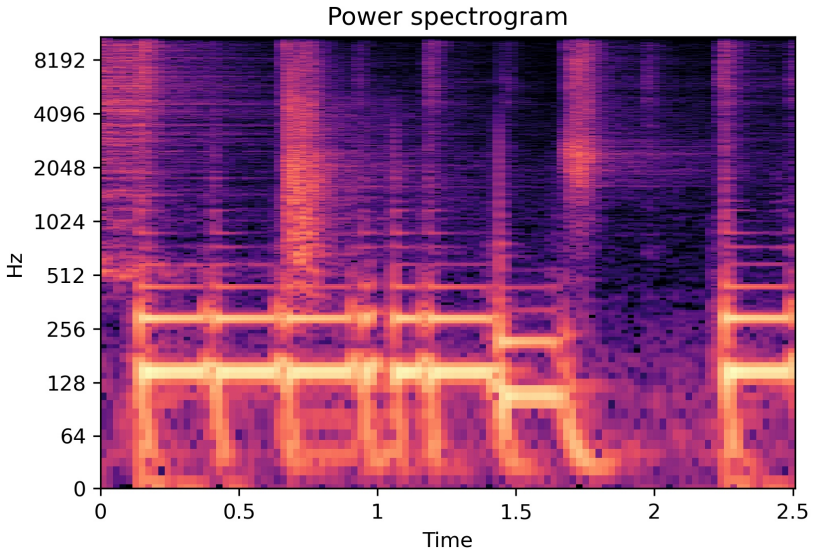

Figure 2. A power spectrogram from the first few seconds of Under Pressure by Queen.

2.3 Performed tasks

In this section we outline the performed music information retrieval tasks that our final program was able to perform. We also take a look at how these problems have been solved previously in the literature.

2.3.1 Tempo estimation

Tempo is the speed or pace of a music piece. It is typically measured in beats per minute(BPM). While the tempo of a song can change during the song, we assume that it doesn't. We tested out multiple approaches for finding the BPM of songs. There are 3

beat trackers in the madmom module which can be used to find the beats in the music and based on those beats we could estimate the tempo. The beat trackers use a recurrent neural network, a conditional random field and recurrent neural networks with a dynamic bayesian network approximated by a Hidden Markov Model [BKS+21a]. These methods are computationally quite expensive and can take a long time. We also tried using the tempo estimator from the librosa module.

According to our preliminary testing, the tempo estimator from the librosa library gave better results. Only a few pop/rock songs were used for these tests. The accuracy could be different when using different genres of music. Detecting the tempo of these genres is generally quite easy due to having a repeating drum pattern. However, the robustness of the beat tracking systems is often much less guaranteed when dealing with classical music because of the weakness of the techniques employed in attack detection and tempo variations inherent to that kind of music. [MAR04]

The tempo estimator from the librosa library uses cyclic tempograms which are a robust mid-level representation that encodes local tempo information [GMK10]. They are similar to spectrograms where instead of pitch, the tempo is on the y-axis.

2.3.2 Key detection

In music, a key is the main group of pitches, or notes, that form the harmonic foundation of a piece of music [Cha21]. The key of a song can change in the middle of a song in a process known as modulation, however in this thesis we assume that a song only has a single key. In this work, we used the key detection method provided by the Madmom module. It is a convolutional neural network (CNN), that is fed a log-magnitude log-frequency spectrogram of an audio signal [KW18]. Up until this paper, typical approaches extracted a feature vector from music and compared it to hand-made template vectors to detect the key. The main drawback of such methods was that they only worked well for one style of music.

For preprocessing, they limited the frequency range of the input to the harmonically most relevant 65 Hz to 2 100 Hz. In contrast to previous works, they also used shorter(around 20 seconds) snippets of music for training. This allowed training to be faster and provided the network with a greater ability to generalize as it could not parts of music that made the key clearer. The proposed network did not use any dense layers. Instead they used more convolutional and pooling layers. Dropout and batch normalization were used to combat overfitting. The network outputs a 24-element vector of probabilites where each number corresponds to a key. This includes the major and minor key for each note in the chromatic scale. Different modes such as the lydian are not detected. The resulting network was evaluated on different styles of music: electronic dance music, pop/rock and classical music. This approach outperformed state-of-the-art

solutions, which were tuned to specific genres. This paper also mentions the difficulty of detecting modulations in music due to the difficulty of classifying shorter excerpts of music. Key detection generally requires a large amount of context [KW18].

2.3.3 Form discovery

Form discovery in this context refers to discovering the structure of a music piece and dividing it into segments. It is an important task in music information retrieval, since it allows us to describe different parts of songs on their own. Some songs might include key modulations between segments and such segments would require their own analysis. Song structure is typically depicted with capital letters from the start of the alphabet. For example, a song that contains a verse, a chorus and another verse has a structure of ABA. A popular structure of Pop/Rock songs is ABABCB, where A depicts the verse, B is the chorus and C is the bridge or the solo. Songs also typically contain an introduction and outro segment which are mostly instrumental, even in songs that contain vocals. The form of music pieces can also be more complicated. There may be variations in between segments that are considered to belong to the section of a song. Form discovery can be considered somewhat subjective due to that.

For performing segmentation we used a convolutional neural network. We use code [OL21], which is based partially on a paper focused on boundary detection in music structure [USG14]. Firstly the beat tracker from the Madmom module is used for extracting all beat times from the music. The beat tracker uses recurrent neural networks and a dynamic Bayesian network [BKW16]. The beat times are used in the pooling stage of the neural network. For each audio file, a log-scaled Mel spectrogram is extracted from the audio signal with a window size of 46 ms (2048 samples at 44.1 kHz) and 50% overlap. Context windows of 16 bars are then classified to determine whether the central beat is a segment boundary. In other words, the input to the CNN is a spectrogram excerpt of N frames, and its output is a single value giving the probability of a boundary in the center of the input.

As suggested in [OL21], we use the Internet archive portion of the SALAMI dataset [SBF+11] for training and validation. It is a publically available dataset which contains audio files and also has manually annotated segmentation for those songs. Some songs were unavailable for download and we ended up with 422 songs for training and 40 for validation. The songs are live recordings from various different artists. There was no genre metadata available for these recordings. The researchers achieved a boundary detection f-Measure of 59% at a tolerance of 2 beats for a random 0.9/0.1 split [OL21]. According to our own testing, the network seems too sensitive as it detects too many segments. In some songs, the border where a chord sequence repeats is detected. The network also seems sensitive to changes in vocals. No singing in the middle of verses can

be detected as a different section. This leads to cases where sections of songs are split into many segments. Some of those segments can also be very short in duration(under 3 seconds).

In order to discover the song structure we would need to cluster the detected segments. This serves two purposes: finding repeating parts in songs such as choruses and combining segments that actually belong to the same section of a song. An option for clustering is by creating a self-similarity matrix of the song and then using non-negative matrix factorization to create a new feature space [KS10]. We followed this paper to create the self-similarity matrix. The song was divided into frames with a length of 400ms and a hop size of 200ms. We then extracted the first 13 MFCC coefficients, spectral centroid, spectral slope and spectral spread for each frame. Those were combined with the 12-dimensional chroma features, each dimension corresponding to a pitch. The values of all features were normalized before calculating the self-similarity matrix. The self-similarity matrix was then created by calculating the distance between every frame using an exponential variant of cosine similarity. This leaves us with an $N \times N$ matrix where N is the amount of frames in a song. Every number in this matrix shows the similarity between two frames as a number between 0 and 1. The created self-similarity matrix can be seen in Figure 3. The label axes show the frame number.

We attempted to use the method outlined in [KS10] for clustering. They used non-negative matrix factorization. Given an $n \times m$ non-negative matrix S, NMF aims at estimating the non-negative factors $W(n \times r)$ and $H(r \times m)$, that best approximate the original matrix : $S \approx WH$. r is the rank of decomposition and a higher rank of decomposition will generally yield a better approximation of the original matrix. After the factorization, we can create decompositions A_k of the original matrix S using the following formula : $A_k = W(:, k)H(k, :)$. This allows us to create r decomposed matrices. The new feature space is created by taking the diagonal of each of these decomposed matrices. This leaves us with a new feature space of size $r \times N$. The researchers suggested using the Bayesian information criterion or Mahalanobis distance to cluster the segments in the new feature space, but their explanation lacks detail and we were unable to recreate their approach. Figure 4 shows an example of one of these decomposition layers. It shows the similarity between the intro and outro of the song.

Since the previously discussed method for clustering did not work for us, we devised a method for clustering based on the self-similarity matrix. Firstly we calculated the similarity between every segment by taking a mean of the area in the self-similarity matrix that shows the similarity of the frames in both segments. This can be seen in Figure 5. The highest similarity value was then selected. If the two segments were next to each other, they were combined into one segment. Otherwise they were simply placed into one cluster. The final number of clusters was set to 4 as in [KS10]. This could be changed based on the genre of music or there could be a lower bound to the similarity

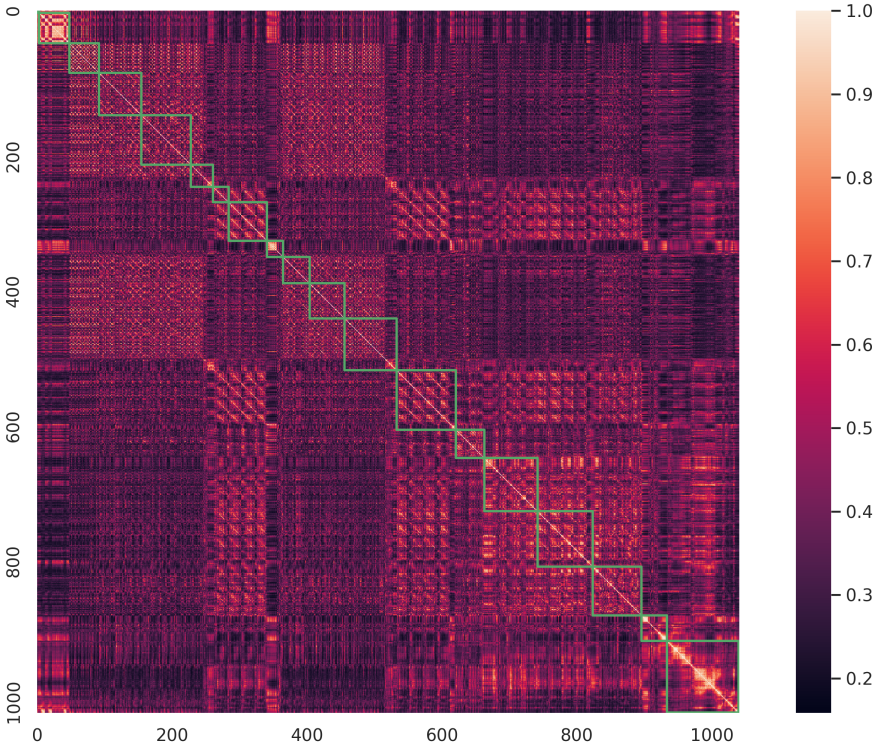

Figure 3. Self-similarity matrix of Highway to Hell by AC/DC with overlapping segmentation created with the convolutional neural network

score after which no more segments are combined. The result of this method can be seen in Figure 6.

2.3.4 Genre classification

Genres in music are a way to categorize music. Songs belonging to the same genre generally have similar instrumentation, style and rhythm. Determining a genre of a song can be quite subjective. Classifying the genre of a song can be useful in music recommendation systems as people often like songs belonging to specific genres.

There are many datasets that include genre information but the biggest of them is the FMA dataset [DBVB17]. It contains a total of 106574 tracks of varying length with 879 GiB of data. They provide genres for each of these songs. There is also a genre hierarchy, with 161 total genres. The researchers have provided subsets of this dataset that have less tracks and less genres. For our purposes we selected the smallest available dataset called FMA small. It contains 8000 tracks of 30s, with 8 balanced genres represented.

Figure 4. One layer of decomposition of the self-similarity matrix of Highway to Hell by AC/DC with overlapping segmentation created with the convolutional neural network

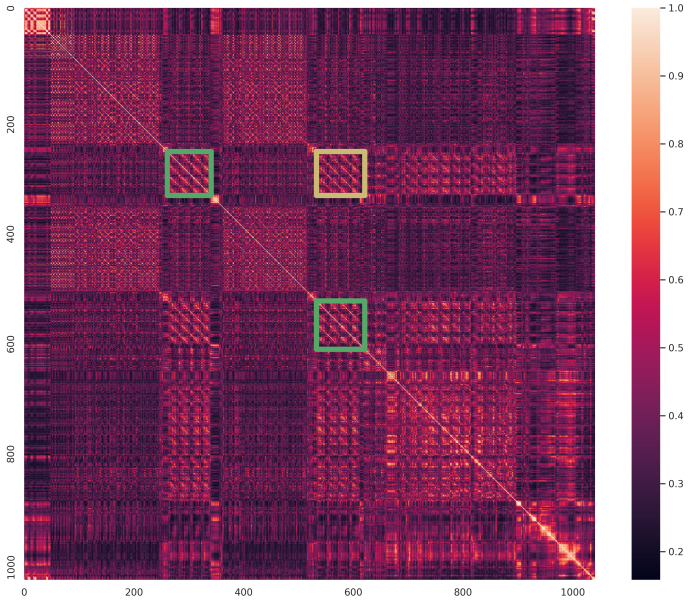

Figure 5. Self-similarity matrix of Highway to Hell by AC/DC showing the area where the similarity of two segments(green) was measured(yellow)

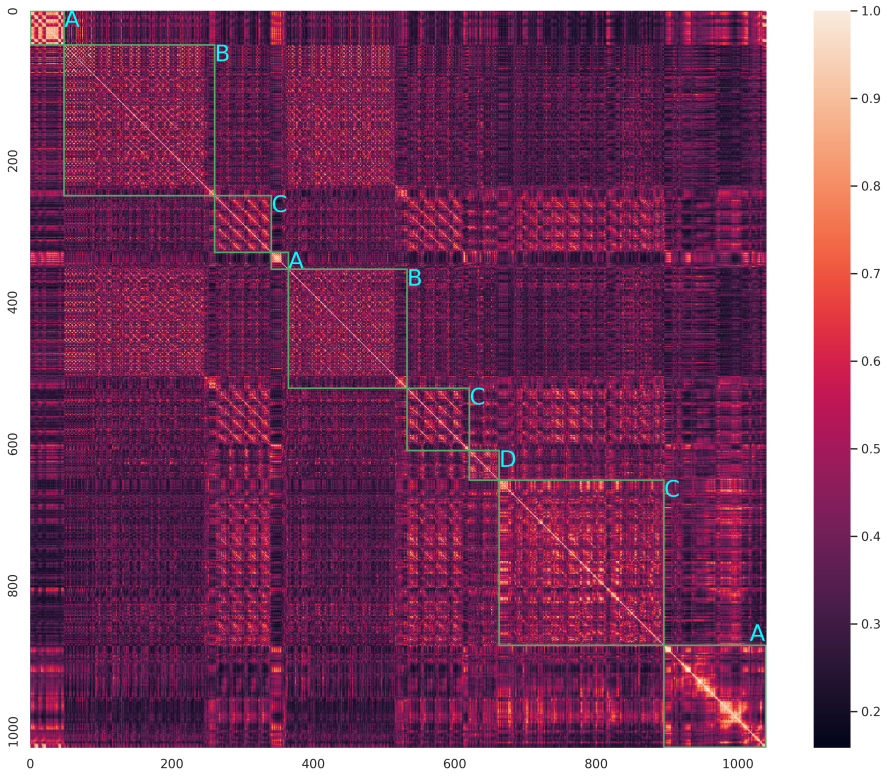

Figure 6. Self-similarity matrix of Highway to Hell by AC/DC with overlapping segments after clustering

There were 1000 tracks for each genre. The genres present in the dataset were hip-hop, pop, folk, experimental, rock, international, electronic and instrumental.

For creating the model for genre classification we followed the approach outlined in [Don18]. It is a convolutional neural network. In the original paper they used the GTZAN dataset which contains 1000 clips of 30s that represent 10 genres. We decided to use the smallest subset of the FMA dataset. There were 6 corrupted files which were deleted. The model takes as input a mel-spectrogram that represents 3 seconds of audio. The input is a matrix of shape 64×256, where 64 is the number of bands in the mel-spectrogram and 256 is the number of windows with 50% overlap where the spectrogram was calculated. Preprocessing included converting the spectrogram to log-scale.

Training was conducted by selecting 3 second segments from the entire training set and predicting their genre. For testing and actual prediction purposes, the song is divided into 3 second clips. The mel-spectrograms are extracted for each of these clips and the

result of the prediction is the majority vote among all clip-level predictions. 80% of the data was used for training, testing and validation sets each had 10% of the entire dataset.

The authors of [Don18] also provided the code that they used for training and testing the model. They reported achieving a 70% accuracy in genre classification. We were only able to achieve an accuracy of around 45% on the testing data using the FMA subset. Similar results were replicated when using the exact dataset used in the paper. It is unclear what caused such disparity.

2.3.5 Chord recognition

For chord recognition we used the built-in processors from the Madmom module. This approach combines a fully convolutional neural network(CNN) for feature extraction with a conditional random field(CRF) for chord sequence decoding [KW16]. Since the CNN has fixed-size kernels, the audio must first be preprocessed. This is done by computing the magnitude spectrogram of the audio and applying a filterbank with logarithmically spaced triangular filters. The filtered magnitudes are logarithmised again to compress their value range. Pooling and dropout layers are used to combat ovefitting. This method performed comparably to other state-of-the-art methods for chord recognition at the time of writing. The network is capable of detecting 24 different chords. This includes the major and minor triads for each note in the chromatic scale. More complicated chords, for example ones that include more notes, are not detected.

2.3.6 Instrument recognition

Instrument recognition is the task of retrieving which musical instruments were used for a piece of music. It might make it easier to predict the genre of a song based on the instruments. It is expected that rock music would include guitars and drums, while classical music might include violins and cellos for example. This is also necessary for music platforms that want to create better recommendations, as they could more easily recommend similar pieces of music if they know the instrumentation.

Instrument recognition can mean recognizing a single instrument from a single played note. This task was performed in [TRW17]. They compared using the attack of the sample to using the entire sample. The attack is the significant rise of the sound, ending in a longer decay period. This usually means the very first moments of a note being played with an instrument, but it differs from instrument to instrument. The dataset used in this paper was the London Philharmonic Orchestra dataset [Lon21]. It contains recordings from 20 different instruments and also includes several playing techniques. The experiments were performed on 8 instruments. The authors conducted several experiments where they focused on the attack or the entire signal. They also

tried limiting the frequency range. Before training, the audio signal was converted to the frequency domain by using Fast Fourier transform. The resulting frequency distribution was partitioned into ranges of frequencies. A multilayer perceptron was used to predict the instrument based on the partitioned frequency distribution. The best result was acheived when using the entire signal length and frequency range. They achieved an accuracy of 93.5%. Sadly, this approach is not applicable to our work, since we have entire songs, where multiple instruments overlap.

Traditional music information retrieval tasks are performed by extracting features such as a spectrogram from audio, but in [LQW15], the researchers performed the task of instrument recognition on raw audio data. Given an audio clip, their model should output a boolean vector of length l, where l is the amount of instruments. Each value in this vector would correspond to the existence of an instrument in a clip. The dataset used in this paper was MedleyDB [BST+14]. It contains 122 annotated musical recordings that have confidence scores of an instrument being present in a time frame. The researchers used a fixed threshold to label time frames as either containing an instrument or not. The original dataset contains annotation about 82 instruments but they were grouped and 11 classes remained. The predictions were performed on clips that lasted 1 second each. A convolutional neural network was used in order to generate predictions from the raw audio. Compared to the baseline of using mel-frequency cepstral coefficients and a random forest or logistic regression, the neural network performed considerably better.

In [LC16] the researchers proposed using a convolutional neural network on the spectrogram of a music clip of music for predicting the instrument. They used 1-d and 2-d convolutions in addition to rolling up the log-frequency axis into a Shepard pitch spiral. High and low frequencies were subject to different convolution strategies. This allowed them to outperform regular convolutional neural networks. Their approach was also trained on MedleyDB [BST+14], similarly to [LQW15] to but as the testing data was different, the results of these papers can not be compared. The previous paper also performed predictions on clips of 1 second while the latter used 3 second clips.

Convolutional neural networks were also used for instrument detection in [HKL16]. They calculated the mel-spectrogram of the music with different frame sizes and also compared activation functions. Their work was performed on the IRMAS(Instrument Recognition in Musical Audio Signals) dataset [BJFH12]. It contains 3 second .wav files of polyphonic music for 11 different instruments. The training dataset contains 6700 annotated excerpts of 3 seconds in which only one instrument is predominant. The testing set consists of around 3000 excerpts of variable length annotated with one to five instruments. The researchers in [HKL16] also conducted experiments with different window sizes for analysis and concluded that 1 second windows performed the best. Smaller windows were not reliable for instrument detection and larger windows left some instruments undetected. The achieved results outperformed previous approaches on this

dataset in when released in 2016.

In 2020 another paper was released [RKJ+20] that was focused on instrument recognition on the IRMAS dataset. Only six instruments were chosen from the original eleven for training and testing. Their approach used the first 13 mel-frequency cepstral coefficients(MFCCs) and spectral features such as zero crossing rate, spectral centroid, spectral bandwidth and spectral roll-off. Common machine learning methods such as logistic regression, support vector machines and decision tree based models were tested. The best performing model was the support vector machine. The presented results were better than in the previously discussed paper [HKL16] on the same dataset but as they did not use all the instruments, the results can not be compared directly.

Convolutional neural networks have also been used for frame-level instrument recognition [HCY19]. Previously discussed papers focused on whether an instrument was present in a piece of music or not. Frame-level instrument recognition takes this a step further by also predicting the times at which the present instruments were active. Since there was no dataset that included pitch labels and a variety of genres, the researchers created their own dataset called MuseScore, which contains a total of 500 tracks. For each of these tracks there is an available audio file and also a piano roll for each instrument. The piano roll shows exactly at what times and which notes the instruments played. It is worth noting that the audio files were synthetic, which means they were not performed but rather created by a program. They used constant-Q transform to create spectograms from the songs and used an encoder/decoder network to create piano rolls for all present instruments. The results were compared to methods from three other papers. The only model that outperformed the one detailed in this paper was the one that used a training set that was similar to the testing set.

The quality of machine learning models depends on the dataset on which they were trained. More data is generally better for creating models that work well. The researchers in [HDM18] set out to create a larger, diverse and polyphonic dataset with multi-label instrument annotation. The dataset was created from freely available musical content. For this they used the Free Music Archive. The annotation was semi-automatic, as the candidates for the dataset were created automatically using a multi-instrument estimator. This estimator was created based on AudioSet [GEF+17]. The final annotation was crowd-sourced as over 230K judgements from more than 2,500 unique contributors across the 20 instrument classes were collected. The dataset includes 20000 excerpts of 10s from songs of various genres that have been labelled for the presence of 20 instruments. The developers of AudioSet have also released a pre-trained feature embedding model based on the VGG architecture for object detection in images. This model produces a 128-dimensional feature vector every 0.96 seconds with an non-overlapping windows. The output of this model is available for each audio file in the OpenMIC dataset. These embeddings were used in conjunction with random forests to create baseline results

based on the OpenMIC dataset.

The embedding model was also utilized in [GSL19]. They compared several approaches for the task of multi-label instrument detection. The best-performing model was a neural network that used attention. In addition to being the best-performing, it was also quite lightweight compared to fully connected neural networks. Due to the code used for this paper being open-source [Gur21], it was easy to replicate. The results were also promising. Due to these reasons this is the approach we follow in our program as well.

3 Natural Estonian descriptions

This section gives a quick overview of music description and analysis. We also discuss details about how we created descriptions from music.

3.1 Music description

Analysing and describing music is not at all a new endeavour. According to [BD87] the start of musical analysis can be traced back to the 1750s. There have been essays released as early as 1935 on the analysis of classical music [Tov44]. These essays were meant for the general public who would go to concerts. Musical analyses generally contain information which could not be retrieved directly from the music and require previous knowledge about the author or era in which it was written. There have also been attempts at describing music using music by Hans Keller using his methods known as functional analysis [O'H20].

When analysing and describing music, there are some objective measures such as the tempo or the duration of a song. There are also far more subjective features like the genre or mood of a piece. There is also a varying amount of detail in musical analysis. One could analyse every note present in the melody or not describe the melody at all in their work. Because of these reasons, it is hard to create a standard that could be followed.

Computational music information retrieval approaches typically use features and create models to describe music by one or more of the following categories of music perception: music content, music context, user properties, and user context [SGGU14]. Those features are mostly numerical. Textual features are generally only used in categorical variables. There is research looking into creating novel features [MDP08]. Out of the previously mentioned categories of music perception, this thesis only focuses on musical content, since as the input we are only getting a sound file and not any context with it.

In our research we could not find any papers looking into generating music descriptions in natural language. Natural language processing has been used in the field of music information retrieval but as a way of gathering data about music [OEAGS18]. There have also been attempts at generating music from natural language [Ran15]. Natural language processing has also been used to analyse the lyrical content of songs for discovering structure and finding similar songs [MMC+05].

3.2 Description generation

Our original idea for generating descriptions for music was to use machine learning to create a model that would generate descriptions about music given the audio. Automatic generation of descriptions has been done before in videos [RQT+13]. Unfortunately we could not find data suitable for this task. The textual information in large music datasets usually consists of the song title, author and album name. There may be some other features such as the genre or present instruments as well. We could not find a dataset that would include text-based music descriptions in a consistent format that would be usable for this thesis. There were some collections of essays in musical analysis, but they were not focused purely on the music and included information about the authors and other topics, which could not be retrieved from the audio. We could not find any collections of musical analyses in Estonian.

Because of these reasons, we decided to use templates for the generation of descriptions. Template-based systems are natural language generating systems that map their non-linguistic input directly (i.e., without intermediate representations) to the linguistic surface structure [ER00]. Simply put, we wrote sentences that we could insert information into, to create sentences in natural language. For example, we used the sentence "This piece is in ...". The ellipses would then be replaced by the key signature retrieved from the music. This allowed us to create a description for each task completed in the music information retrieval section. The descriptions for each task are added together to create the description for the entire piece. In the following paragraphs we show how descriptions were generated using information retrieved from music.

Tempo description From the music information retrieval we only have one metric related to the tempo of the music. That is a number which shows the amount of beats that occur in a minute in the song. This metric is also known as Beats Per Minute (BPM). To describe tempo, we use the Estonian translations of Italian markings for tempo retrieved from Estonian Wikipedia [Wik21]. We removed overlapping areas of markings and created a mapping between tempo and tempo description. Each tempo range has its own description.

Key description As mentioned in Section 2.3.2, the system used for key detection can detect 24 total keys: minor and major for each note in the chromatic scale. We translate the key label into Estonian and insert it into a template to create the description.

Harmony description From the music information retrieval we receive a list of timestamps and chord labels. This covers the entire song. First, we create a list of chords that

are in the same key as the song based on the retrieved key. We then create a list of all present chords with their fragment of presence in the song. This way every chord has a number assigned to it that shows the percentage of time that it was active. We then list the chords and the in-key chords for the user in the description using templates. There are multiple templates for cases where only one chord was detected or there were no chords detected. The chords are listed in order of percentage that they were active. This system only handles major and minor triads, more complicated chords are not detected.

Form description The input to the form description generator is a list of segments with start times, end times and cluster labels. The times are not used for our purposes. We add together all cluster labels in the order in which they appear in the song and insert that into a template to create the description for the form of the music.

Instrument description The instrument recognition system detects 20 different instruments and returns a list of instruments that are present in the music. We create a description which lists all of the instruments. There are multiple templates to deal with the case of not finding any instruments. One of the detected instruments was the human voice. There is a separate template for the human voice. If it is present, it is put into a separate sentence after the sentence listing the instruments.

Genre description As input, the genre description generator receives a string which shows the detected genre of the song. This input is then mapped to a description which contains the genre text in Estonian. Instrumental was one of the detectable genres, but if that is the most likely genre, the next most likely genre is selected instead. This is because instrumental is not really a genre, it is rather a type of music.

4 Used modules and frameworks

This section contains an overview of the frameworks and modules that were used to create the application for this thesis.

4.1 Selection of programming language

Python was chosen as the programming language for this thesis due to various reasons. Firstly, Python has many freely available modules for music information retrieval [MRR15]. Using modules for this task saves time which can be spent on other aspects of the thesis. Secondly, there are also many modules for creating web services in Python [Pyt21]. These modules are necessary for creating a web version of the program. Finally, Python is the programming language that the author is most familiar with and learning other programming languages would be an additional time-cost. C++ and MatLab are also viable options for music information retrieval(MIR), but they are not as commonly used in creating web applications.

4.2 Module selection for MIR

This subsection contains overviews of some MIR modules available for Python and includes the decision process for choosing one that would be used in the final program. We looked at what tasks these modules were built for. Another important factor was the ease of learning since the author has no previous experience with any of the discussed modules. Modules with thorough documentation and tutorials were better for this. In addition to that, we looked at the existence of high level feature extraction. It would be possible to create high level features ourselves, but if there are state-of-the-art solutions already present in modules, we could use those. In addition to that we looked at the time that these modules took for calculating a similar metric. We don't want users to wait too long to get their answers.

4.2.1 Aubio

Aubio is a open-source tool designed for the extraction of annotations for audio signals [BTM+03]. The first version was released in 2003 by Paul M. Brossier and others. The latest version was released in February of 2019 [BTM+19] and it is still being updated. Aubio was written in C, but also has a Python interface. Aubio arrays can also be viewed simply as NumPy arrays, making work with them easier in Python. It can be used for tasks such as onset detection, beat tracking, tempo extraction and melody

extraction [Bro06]. Aubio has quite a thorough documentation which also includes documentation about the Python interface [Bro18]

4.2.2 Essentia 2.0

Essentia [BWGG+21] is an open-source C++ library for audio analysis and audio-based music information retrieval [BWGG+13]. It contains a large collection of algorithms used for digital audio processing and a large amount of music descriptors. Essentia was designed with the purpose of providing robustness and computational efficiency making it usable in large-scale applications. It was first released in 2013 and is still being updated. It was written by researchers in the Music Technology Group in Pompeu Fabra University in Spain. The library is written in C++ but also has a Python wrapper. They provide extensive documentation about the library which also contains a tutorial about the Python wrapper. Essentia is used in many companies and projects around the world [BWGG+21].

4.2.3 Librosa

Librosa is an open-source [McF21] Python package for audio and music signal processing [MRL+15]. Librosa provides implementations for common functions in the field of music information retrieval. It was designed to help transition MIR researchers into Python and to make MIR techniques available to more people. Unlike the previous two modules, all functions are implemented in pure Python. This package also contains interfaces to visually represent audio data. This might be useful in further understanding the music and also for development purposes.

Librosa was first released in January of 2014. The latest version released was in July of 2020. Similarily to the previously discussed modules, they also have a through documentation and a guide for new users [MRL+21]. There is also a large collection of community-made freely available Jupyter Notebooks showing code examples [TSC+21].

4.2.4 Madmom

Madmom is an open-source [BKS+21b] audio processing and music information retrieval (MIR) library [BKS+16] written in pure Python. It was written by people in the Johannes Kepler University and the Austrian Research Institute for Artificial Intelligence. In addition providing low-level features, madmom provides high level features for describing music. Many of those features are generated by using machine learning models such as Hidden Markov Models or neural networks. It contains state-of-the-art-algorithms

for tasks such as tempo estimation and chord recognition. Madmom was first released in October of 2015 and the latest version was released in November 2018. There is documentation that contains tutorials for newer users [BKS+21a].

4.2.5 Marsyas

Marsyas (Music Analysis, Retrieval and Synthesis for Audio Signals) is an open-source [TLN21] software framework for audio processing with specific emphasis on Music Information Retrieval applications [Leb15]. It was first released in the year 2000 [TC00], but has received updates since then. The researchers responsible for this framework also took part in the Music Information Retrieval Evaluation Exchange (MIREX) competition [Tza07]. They completed tasks in audio tag classification, audio onset detection and audio beat tracking. Marsyas was originally written in C++, but has Python bindings. Documentation exists for C++ part, but there were limited instructions for using the Python bindings. The framework lacks high level features present in other modules.

4.2.6 YAAFE

YAAFE is an open-source [MEF+21b] audio feature extraction program [MEF+10]. It was developed with a focus on efficiency. This efficiency comes from a lower complexity of calculations due to reducing redundancy. They also focus on the ability to process very long audio files while not taking up too much memory. YAAFE is primarily a command line program. It was written using C++ for calculations and Python for input parsing and creating the feature library. It was first released in 2010 and the latest version came out in 2017. YAAFE lacks high-level features and is instead focused in efficiency in calculating lower level features. The documentation contains a user manual and an overview of all usable functionalities [MEF+21a].

4.2.7 Final module choice

Most of the discussed modules had thorough documentations and code examples. One exception was the Marsyas module which was lacking a Python guide. All of the discussed modules were open-source and free to download and use. Most modules also had some way of generating high-level features such as predicting the key or tempo of a piece.

Figure 7 shows computation time taken by several modules for calculating Mel-frequency cepstral coefficients(MFCCs). Mel-frequency cepstral coefficients are com-

mon metrics used in the field of music information retrieval and require extensive calculations to achieve. They are used for creating many higher-level features. MFCCs were calculated for 16.5 hours of audio data and the figure shows time taken in seconds. The modules written in C++ were significantly faster in this task. Extrapolating from this figure we can see that calculating MFCCs for 8.5 seconds of audio would take around 1 second with the librosa library. A 3 minute piece would take around 38 seconds in that case. Obviously this all depends on the hardware used, but at least we have some estimate. In our own testing we found, that librosa is fast enough, since our program is not meant to handle large amounts of data. Neural networks used for many tasks in this thesis take much more time than low-level feature extraction.

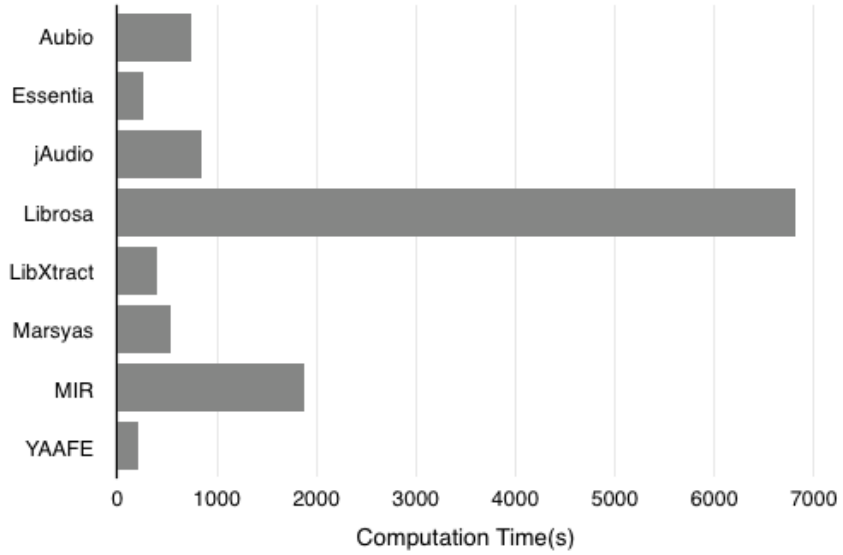

Figure 7. Graph of Computational Time of Feature Extraction Tools [MRR15]

We decided to use madmom and librosa for this thesis. Madmom, because it has the largest amount of high-level features out of all the discussed modules. It has also implemented some state-of-the-art solutions for generating these features. Most other modules have simpler solutions to the same problems and might not be as accurate. Librosa is used mainly for extracting low-level features, some of which are not available in madmom.

4.3 Module Selection for Web Application

An online survey conducted 42279 among software developers in February of 2020 [Sta20] found the most popular Python web frameworks to be Django [Dja21] and Flask [Fla21]. In this subsection we will be looking at those two frameworks in detail and picking one that best fits this project.

4.3.1 Django

Django is the most popular web framework for Python [Sta20]. Its popularity is also shown by the fact the there are over 260000 questions on Stack Overflow with the 'django' tag. Django is a full-stack framework. Full-stack frameworks are frameworks that help with the full development stack from the user interface till the data store. It gives full support to developers including basic components like form generators, form validation, and template layouts [SFV19]. The key features of Django are speed, security, scalability and versatility [SFV19]. Django has been used to develop some of the largest sites in the world such as Pinterest and Instagram. It was first introduced in 2003 and later released as an open-source project in 2005 [Ghi20]. It has remained open-source ever since.

4.3.2 Flask

Flask is a lightweight web application framework [Fla21]. It is the second most popular web framework for Python [Sta20]. There are over 43000 questions on Stack Overflow with the 'flask' tag. Flask is a micro framework. Micro frameworks are lightweight frameworks that don't offer additional functionalities and features, such as database abstraction layer, form validation, and specific tools and libraries [SFV19]. The key features of Flask are its lightweight framework, its compatibility with Google App Engine and its built-in development server and debugger [SFV19]. Flask has been used to develop some of the largest sites in the world such as Reddit, Airbnb and LinkedIn. It was first released on April 1, 2010 and is also an open-source project.

4.3.3 Final module choice

A bachelor's thesis released in 2020 compared Flask and Django from the point of view of a novice web developer [Ghi20]. To perform the comparison the authors built 2 applications with both frameworks: a social network similar to Facebook and an eCommerce like application. They found that Flask provided simplicity, flexibility and it was quick and easy to learn. The main advantages of Django were its extensive features and library support. The authors noted that Django was too cumbersome for smaller sized

applications. Because the program created for this thesis is not too large, we decided to go with Flask as it should be easier to learn.

4.4 Docker

We decided to use Docker for the development and deployment of this project. This would simplify development as there are many modules and programs necessary for this project and Docker makes it easier to develop and deploy on different machines. It allowed us to create containers that were for specific tasks, such as training neural networks.

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and deploy it as one package [Ope21]. Using containers is similar to using virtual machines, but requires less overhead, making them faster and easier to work with [RBA17].

Modern applications are often assembled from existing components and rely on other services and applications. Each of these components comes with its own set of dependencies that may conflict with those of other components. By packaging each component and its dependencies, Docker solves the problems such as conflicting dependencies and platform dependencies [Mer14]. If a system runs Docker, the same container will execute similarly on all other systems running Docker.

Docker containers can be created by creating a dockerfile and building it. Dockerfiles contain all the necessary information for creating containers. They include information about the operating system and all the commands that need to be executed before use. This allows the user to specify the software versions that are installed. Developers can make sure that their software runs smoothly inside the container without having to worry about the environment in which it is run.

A selling point of Docker is the availability of ready-to-go docker images. There are over 5 million images available for download in Docker Hub(`https://hub.docker.com/`). Thanks to Docker Hub, it is easy to find an image with most of the necessary installed software. Docker is available on Windows, MacOS and Linux, making cross-platform development easy. It is actively used by millions of developers around the world. [Doc21]. It has also been suggested for reproducible research [Boe15].

5 Implementation specifics

This section contains information about the implementation of the program created as part of this thesis. The source code for this program is available at `https://gitlab.cs.ut.ee/villemtn/muusikakirjeldus`. Commands for starting the application are available in the README in GitLab. A link to the application will also be in GitLab.

5.1 Docker

As mentioned previously we used Docker for creating the environment in which to run the application. We decided to use the image *python:3.7-buster*. It includes Python version 3.7 and it runs on Debian 10. We decided to use Python 3.7 because the music information retrieval modules have not been updated lately and we were afraid that using they would not be compatible with newer versions of Python. In addition to Python we also needed FFmpeg for converting audio between formats. All necessary modules were also installed using the Dockerfile. The versions of the modules can be found in the Dockerfile or in *requirements.txt*. After installing all required modules, the application files are copied to the work directory of the container. Following that, the application is started.

5.2 Web application architecture

The web application part of the project was created using Flask. The design of the application is very simplistic and it only contains a single page. This page contains the title of the project, a field for uploading the music file, a text area for giving information to the user and checkboxes with which the user can select which tasks they want to be completed. Figure 8 shows the layout of the web application. The web application functionality is implemented in *app.py*.

If a file has not been uploaded yet, the text area if filled with the text that tells the user to upload a file. When the user uploads a music file, the file is saved in the *uploads* folder. The path to this file and a list of the tasks to be completed is then given to the music information retrieval function. The music information retrieval function returns a dictionary of task and result pairs. The exact return values are discussed in more detail in the following section. This dictionary is the given to the description generation function. The description of the music in natural Estonian is created in *description_generation.py*. Based on the completed tasks, a description is created. There is function for each task. Each of the functions contain the templates and mappings necessary for the creation of descriptions. The outputs of these functions are then added together to create the entire

description. The uploaded file is deleted once the request has been completed.

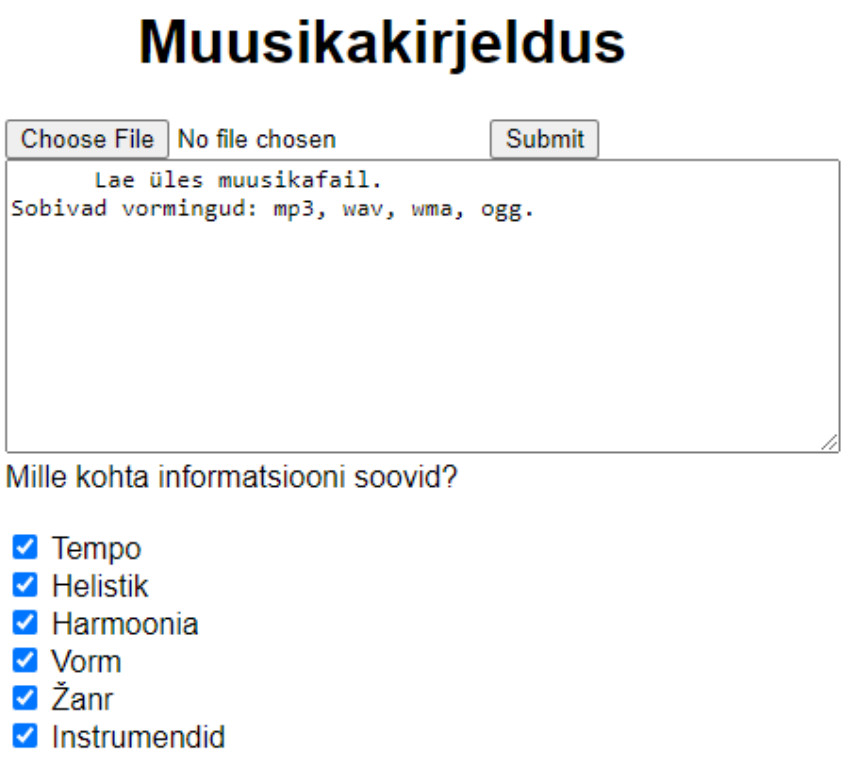

Figure 8. Layout of web application

5.3 Music information retrieval

The music information retrieval part of this program is implemented in *mir.py*, *segmentation.py*, *instrument_recognition.py* and *genre_recognition.py*. Some functions and trained networks are also in the *models* folder. *mir.py* receives the tasks and the path to the song and returns a dictionary where each task has a response. To do this, the waveform is read from the song file. This waveform is then given to necessary functions to produce the output.

Tempo, key, chords These features are extracted in *mir.py* using built-in functions in librosa and madmom modules. The tempo is returned as a float that shows the estimated beats per minute(BPM) of the song. The key is returned as a string. The chords are

returned as collection with start time, end time and chord label covering the entire song. Silent parts of songs are labelled as having no chord.

Form discovery Segmentation and clustering of the extracted segments is performed in *segmentation.py*. First, the downbeat tracking necessary for the segmentation is performed using *RNNDownBeatProcessor* and *DBNDownBeatTrackingProcessor* from the madmom module. Following that, the log-scaled mel spectrograms are extracted and max-pooled across beat times. We then normalize the features according to training data for each mel-band. The segment borders are extracted using the convolutional neural network. The self-similarity matrix is created for the song and that is used to cluster the segments. The final result is a list of segments with the start time, the end time and a cluster label for all segments.

Instrument recognition Instrument recognition is performed in *instrument_recognition.py*. First, the features are extracted using the VGGish model with a window size of around 0.96 s. The model can be found in *models/instrument_recognition/vggish*. The features are then divided into bins of 10 since that is the input shape of the instrument recognition network. The model outputs a vector of length 20 containing numbers between 0 and 1, where each number shows the likelihood of an instrument being present. We take an instrument-wise max from the output vectors. A fixed threshold is then applied to decide which instruments are present. The system seemed rather conservative, so the final threshold was selected to be 0.1. This can produce more false positives, but will more often retrieve correct instruments as well. The final result is a list of present instruments.

Genre recognition Genre recognition is performed in *genre_recognition.py*. First the mel-spectrogram for the entire song is calculated. Next up, the spectrogram is divided into segments of 3 seconds with a 50% overlap. The network is used to predict the genre of every segment. The class with the highest mean probability is selected and returned as a string. If the most likely genre is instrumental, we predict the second most likely genre.

6 Testing

To test the music retrieval capabilities of the application, we decided to retrieve information from a few songs and assess the results manually. All songs for testing were retrieved from the Free Music Archive (FMA)[DBVB17]. They provide freely usable and downloadable songs for various genres. We selected 5 total pieces from different genres to see how our system performed. FMA provided genre information for these piecess in addition to the author and the title, but no additional information was provided about the music. The chord recognition performance was not evaluated because there was no available data about the chords.

The first song we selected was Algorithms by Chad Crouch. It is an electronic piece that features a synthesizer. The beats-per-minute (BPM) is around 90. The predicted key is G major, which is correct. The BPM was estimated to be 89.1, which is a good result. The genre was detected to be international music. The system found no instruments present in the song.

The second song we selected was Sorry by Comfort Fit. It is a mostly instrumental hip-hop piece, that features a piano, drums and a synthesizer. There are some vocals used throughout the song as well. The BPM is around 95. The predicted key is F# minor, which appears to be correct. The BPM was estimated to be 95.7, which is once again a good result. The genre was detected to be electronic. The piece features synthesizers and drums which are commonly used in electronic music as well. This may explain the error. The only detected instrument was drums. This means that the piano and synthesizer were undetected.

The third song was Ghost Dance by Kevin MacLeod. It is an instrumental classical piece featuring a piano, some strings and cymbals. We estimated the BPM to be around 66, although it shifts a little during the song. The predicted key is D minor which appers to be correct. The BPM was detected to be 103, which is quite bad. As mentioned previously in the paper, estimation of tempo in classical music can be difficult. The genre was predicted to be experimental and no instruments were detected.

The fourth song was Shelley Winters Overdrive by Kinski. It is a rock song that features a heavily distorted guitar, a bass guitar and drums. The BPM is around 161. The predicted key was C# minor which is correct. The detected BPM was 107, which is wrong. The genre was predicted to be electronic. Drums and guitar were detected in the music.

The fifth song was These Days by Robin Grey. It is a folk song with vocals that features a guitar, a banjo and a bass. The BPM is around 103. The predicted key was F# minor, which is correct. The detected BPM was 103.3, which is correct. The genre was predicted to be hip-hop, which is incorrect. The song does not even feature drums which

makes this mistake very strange. The only detected instrument was guitar. The bass and banjo were undetected.

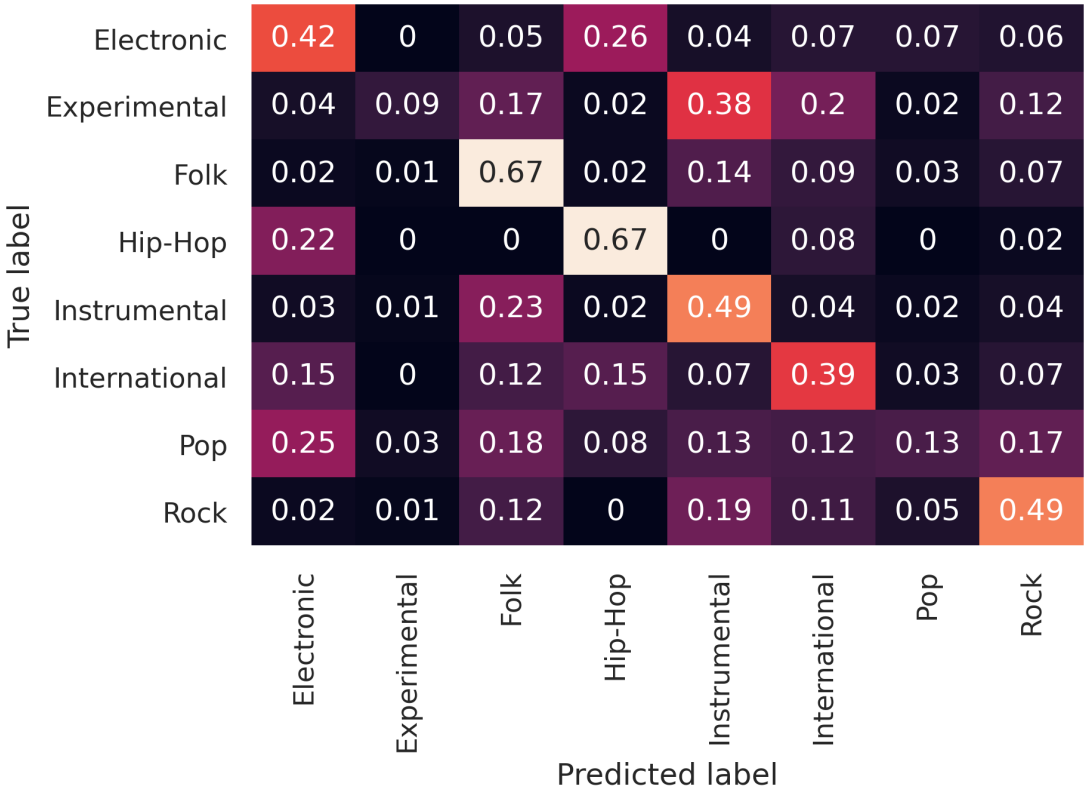

Figure 9. Confusion matrix for genre classification

We also performed testing after training the model for genre classification. The testing was performed on the testing subset extracted from the entire dataset. It contained 640 total songs. As discussed in Section 2.3.4, the used approach did not achieve good accuracy. In Figure 9, we show the confusion matrix for genre detection on test data. We can see that some genres like folk and hip-hop are classified correctly most of the time. This may be due to having defining characteristics that make classification easier. For hip-hop it is the consistent beat and the uniqueness of the vocals. For folk, it might be the acoustic instruments and some specific instruments that are not used in other genres. On the other hand, we have genres like experimental and pop which are rarely classified correctly. This might be because these genres are vague and don't have many defining characteristics. Pop is most often misclassified as electronic. This makes sense because pop music nowadays contains many elements from electronic music. Instrumental music is often misclassified as experimental music. This might be because experimental music

is also typically instrumental. This is also why having instrumental music as its own genre does not make much sense.

We tested the instrument recognition system after training the model. The testing was performed on the testing subset of the OpenMIC dataset and it contained 5085 10 second segments from songs. The results can be seen in Figure 10. We reported the F1-score for each instrument and also reported the macro-averaged F1-score. We can see quite a large difference between instruments. The clarinet is close to 0.65 and guitar is as high as 0.96. This shows us that the difficulty of detecting an instrument in music can depend on the characteristics of the instrument and how well it stands out. It is also possible that instruments such as organ and clarinet are more often used in conjunction with other instruments, making the detection harder.

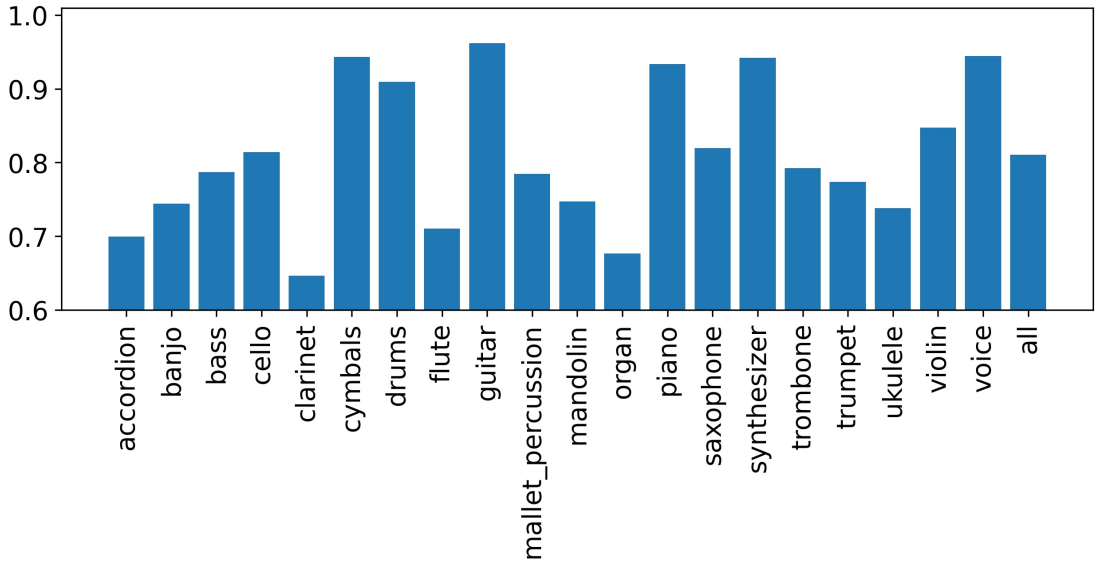

Figure 10. Class-wise F1 scores for instrument recognition

7 Conclusion

In this thesis we gave an overview of the field of music information retrieval and the approaches for some tasks in this field. In addition to that we also give an overview of music information retrieval modules usable in Python. As a result of the thesis a web application was created that can produce a description for music in natural Estonian. The source code for the application has been provided and can be used to further develop this project. There are also implementations of approaches for many tasks in the field of information retrieval.

In future work we would like to retrieve even more information from music. For example, we could assess the mood of a piece. In the current version of the program there are no metrics assessing the performance. This could include locating the culmination of a piece. There could also be a method for extracting the most important parts of a piece and describing them in more detail. In addition to that, some tasks should be redone when more data becomes available. The segmentation part, for example, did not perform too well in our opinion and could use some different methods. In this paper we performed all tasks separately and did not use data from one task to help with others. In the future we could look into using the detected instruments to predict the genre.

The automatic description generation should also be looked at in future work. Since natural language descriptions of music have not really been generated automatically in any languages, it makes sense to start with a language that has more source material. It would be interesting to see if these descriptions could be generated using neural networks. This requires large amounts of data and we were unable to test this approach in this thesis. Another possible improvement would be to use smaller neural networks for music information retrieval tasks. In the current state, the retrieval of all information from a music file can take multiple minutes.

References

[BD87] Ian Bent and William Drabkin. Analysis london, 1987.

[BJFH12] Juan J Bosch, Jordi Janer, Ferdinand Fuhrmann, and Perfecto Herrera. A comparison of sound segregation techniques for predominant instrument recognition in musical audio signals. In *ISMIR*, pages 559–564. Citeseer, 2012.

[BKS+16] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. madmom: a new Python Audio and Music Signal Processing Library. In *Proceedings of the 24th ACM International Conference on Multimedia*, pages 1174–1178, Amsterdam, The Netherlands, 10 2016.

[BKS+21a] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. Madmom documentation. `https://madmom.readthedocs.io/en/latest/`, 2021. [Online; accessed 25-February-2021].

[BKS+21b] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. Madmom GitHub. `https://github.com/CPJKU/madmom`, 2021. [Online; accessed 25-February-2021].

[BKW16] Sebastian Böck, Florian Krebs, and Gerhard Widmer. Joint beat and downbeat tracking with recurrent neural networks. In *ISMIR*, pages 255–261. New York City, 2016.

[Boe15] Carl Boettiger. An introduction to docker for reproducible research. *ACM SIGOPS Operating Systems Review*, 49(1):71–79, 2015.

[Bro06] Paul M Brossier. The aubio library at mirex 2006. *Synthesis*, 2006.

[Bro18] Paul Brossier. aubio Python documentation. `https://aubio.org/manual/latest/python.html`, 2018. [Online; accessed 23-February-2021].

[BST+14] Rachel M Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Pablo Bello. Medleydb: A multitrack dataset for annotation-intensive mir research. In *ISMIR*, volume 14, pages 155–160, 2014.

[BTM+03] Paul Brossier, Tintamar, Eduard Müller, Nils Philippsen, Tres Seaver, Hannes Fritz, cyclopsian, Sam Alexander, Jon Williams, James Cowgill, and Ancor Cruz. aubio website. `https://aubio.org/`, 2003. [Online; accessed 23-February-2021].

[BTM+19] Paul Brossier, Tintamar, Eduard Müller, Nils Philippsen, Tres Seaver, Hannes Fritz, cyclopsian, Sam Alexander, Jon Williams, James Cowgill, and Ancor Cruz. aubio/aubio: 0.4.9, February 2019.

[BWGG+13] Dmitry Bogdanov, Nicolas Wack, Emilia Gómez Gutiérrez, Sankalp Gulati, Herrera Boyer, Oscar Mayor, Gerard Roma Trepat, Justin Salamon, José Ricardo Zapata González, Xavier Serra, et al. Essentia: An audio analysis library for music information retrieval. In *Britto A, Gouyon F, Dixon S, editors. 14th Conference of the International Society for Music Information Retrieval (ISMIR); 2013 Nov 4-8; Curitiba, Brazil.[place unknown]: ISMIR; 2013. p. 493-8.* International Society for Music Information Retrieval (ISMIR), 2013.

[BWGG+21] Dmitry Bogdanov, Nicolas Wack, Emilia Gómez Gutiérrez, Sankalp Gulati, Herrera Boyer, Oscar Mayor, Gerard Roma Trepat, Justin Salamon, José Ricardo Zapata González, Xavier Serra, et al. Essenia website. `https://essentia.upf.edu/`, 2021. [Online; accessed 23-February-2021].

[Cha21] Samuel Chase. What is a Key in Music? A Complete Guide. `https://hellomusictheory.com/learn/keys/`, 2021. [Online; accessed 06-April-2021].

[DBVB17] Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. Fma: A dataset for music analysis, 2017.

[Dja21] Django. Django website. `https://www.djangoproject.com/`, 2021. [Online; accessed 01-March-2021].

[Doc21] Docker. Docker website. `https://www.docker.com/`, 2021. [Online; accessed 07-April-2021].

[Don18] Mingwen Dong. Convolutional neural network achieves human-level accuracy in music genre classification. *arXiv preprint arXiv:1802.09697*, 2018.

[ER00] Reiter Ehud and Dale Robert. Building natural language generation systems, 2000.

[Fla21] Flask. Flask website. `https://palletsprojects.com/p/flask/`, 2021. [Online; accessed 01-March-2021].

[GEF+17] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set:

An ontology and human-labeled dataset for audio events. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 776–780. IEEE, 2017.

[Ghi20] Devndra Ghimire. Comparative study on python web frameworks: Flask and django. 2020.

[GMK10] Peter Grosche, Meinard Müller, and Frank Kurth. Cyclic tempogram—a mid-level tempo representation for musicsignals. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5522–5525. IEEE, 2010.

[GSL19] Siddharth Gururani, Mohit Sharma, and Alexander Lerch. An attention mechanism for musical instrument recognition. *arXiv preprint arXiv:1907.04294*, 2019.

[Gur21] Siddhart Gururani. AttentionMIC GitHub. `https://github.com/SiddGururani/AttentionMIC`, 2021. [Online; accessed 03-May-2021].

[HCY19] Yun-Ning Hung, Yi-An Chen, and Yi-Hsuan Yang. Multitask learning for frame-level instrument recognition. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 381–385. IEEE, 2019.

[HDM18] Eric Humphrey, Simon Durand, and Brian McFee. Openmic-2018: An open data-set for multiple instrument recognition. In *ISMIR*, pages 438–444, 2018.

[HKL16] 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*, 25(1):208–221, 2016.

[KS10] Florian Kaiser and Thomas Sikora. Music structure discovery in popular music using non-negative matrix factorization. In *ISMIR*, pages 429–434, 2010.

[KW16] F. Korzeniowski and G. Widmer. A fully convolutional deep auditory model for musical chord recognition. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2016.

[KW18] Filip Korzeniowski and Gerhard Widmer. Genre-agnostic key classification with convolutional neural networks. *arXiv preprint arXiv:1808.05340*, 2018.

[L+00] Beth Logan et al. Mel frequency cepstral coefficients for music modeling. In *Ismir*, volume 270, pages 1–11. Citeseer, 2000.

[LC16] Vincent Lostanlen and Carmine-Emanuele Cella. Deep convolutional networks on the pitch spiral for musical instrument recognition. *arXiv preprint arXiv:1605.06644*, 2016.

[Leb15] Jakob Leben. Marsyas website. `http://marsyas.info/#`, 2015. [Online; accessed 26-February-2021].

[LN14] Nathan Lenssen and Deanna Needell. An introduction to fourier analysis with applications to music. *Journal of Humanistic Mathematics*, 4(1):72–91, 2014.

[Lon21] London Philharmonic Orchestra. London philharmonic orchestra sound samples, 2021. [Online; accessed 30-April-2021].

[LQW15] Peter Li, Jiyuan Qian, and Tian Wang. Automatic instrument recognition in polyphonic music using convolutional neural networks. *arXiv preprint arXiv:1511.05520*, 2015.

[MAR04] Bertrand David Miguel Alonso and Gaël Richard. Tempo and beat estimation of musical signals. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR), Barcelona, Spain*, 2004.

[McF21] Brian McFee. Librosa GitHub. `https://github.com/librosa/librosa`, 2021. [Online; accessed 25-February-2021].

[MDP08] Luca Mion and Giovanni De Poli. Score-independent audio features for description of music expression. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):458–466, 2008.

[MEF+10] Benoit Mathieu, Slim Essid, Thomas Fillon, Jacques Prado, and Gaël Richard. Yaafe, an easy to use and efficient audio feature extraction software. In *Proceedings of the 11th International Society for Music Information Retrieval Conference*, pages 441–446, Utrecht, The Netherlands, August 9-13 2010. `http://ismir2010.ismir.net/proceedings/ismir2010-75.pdf`.

[MEF+21a] Benoit Mathieu, Slim Essid, Thomas Fillon, Jacques Prado, and Gaël Richard. Yaafe - audio features extraction. `http://yaafe.github.io/Yaafe/`, 2021. [Online; accessed 26-February-2021].

[MEF+21b] Benoit Mathieu, Slim Essid, Thomas Fillon, Jacques Prado, and Gaël Richard. YAAFE github. https://github.com/Yaafe/Yaafe, 2021. [Online; accessed 26-February-2021].

[Mer14] Dirk Merkel. Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, 2014(239):2, 2014.

[MMC+05] Jose PG Mahedero, Alvaro MartÍnez, Pedro Cano, Markus Koppenberger, and Fabien Gouyon. Natural language processing of lyrics. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pages 475–478, 2005.

[MRL+15] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, volume 8, pages 18–25. Citeseer, 2015.

[MRL+21] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. Librosa documentation. https://librosa.org/doc/latest/index.html, 2021. [Online; accessed 25-February-2021].

[MRR15] David Moffat, David Ronan, and Joshua Reiss. An evaluation of audio feature extraction toolboxes. 11 2015.

[OEAGS18] Sergio Oramas, Luis Espinosa-Anke, Francisco Gómez, and Xavier Serra. Natural language processing for music knowledge discovery. *Journal of New Music Research*, 47(4):365–382, 2018.

[O'H20] WILLIAM O'HARA. Music theory on the radio: Theme and temporality in hans keller's first functional analysis. *Music Analysis*, 39(1):3–49, 2020.

[OL21] Ben Osheroff and Matthias Leimeister. Segmentation CNN GitHub. https://github.com/mleimeister/SegmentationCNN, 2021. [Online; accessed 22-April-2021].

[Ope21] Opensource.com. What is docker? https://opensource.com/resources/what-docker, 2021. [Online; accessed 07-April-2021].

[Pyt21] Python Wiki contributors. Python web frameworks — Python Wiki, 2021. [Online; accessed 22-February-2021].

[Ran15] Rohit Rangarajan. Generating music from natural language text. In *2015 Tenth International Conference on Digital Information Management (ICDIM)*, pages 85–88. IEEE, 2015.

[RBA17] Babak Bashari Rad, Harrison John Bhatti, and Mohammad Ahmadi. An introduction to docker and analysis of its performance. *International Journal of Computer Science and Network Security (IJCSNS)*, 17(3):228, 2017.

[RKJ+20] 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. IEEE, 2020.

[RQT+13] Marcus Rohrbach, Wei Qiu, Ivan Titov, Stefan Thater, Manfred Pinkal, and Bernt Schiele. Translating video content to natural language descriptions. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 433–440, 2013.

[SBF+11] Jordan Bennett Louis Smith, John Ashley Burgoyne, Ichiro Fujinaga, David De Roure, and J Stephen Downie. Design and creation of a large-scale database of structural annotations. In *ISMIR*, volume 11, pages 555–560. Miami, FL, 2011.

[SFV19] AL Sayeth Saabith, MMM Fareez, and T Vinothraj. Python current trend applications-an overview. *International Journal of Advance Engineering and Research Development*, 6(10), 2019.

[SGGU14] Markus Schedl, Emilia Gómez Gutiérrez, and Julián Urbano. Music information retrieval: Recent developments and applications. *Foundations and Trends in Information Retrieval. 2014 Sept 12; 8 (2-3): 127-261.*, 2014.

[Sta20] Statista. Most used web frameworks among developers worldwide, as of early 2020. `https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/`, 2020. [Online; accessed 01-March-2021].

[Sup21] Supermegaultragroovy. Capo website. `https://supermegaultragroovy.com/products/capo/`, 2021. [Online; accessed 13-May-2021].

[SVN37] Stanley Smith Stevens, John Volkmann, and Edwin Broomell Newman. A scale for the measurement of the psychological magnitude pitch. *The journal of the acoustical society of america*, 8(3):185–190, 1937.

[TC00] George Tzanetakis and Perry Cook. Marsyas: A framework for audio analysis. *Organised sound*, 4(3):169–175, 2000.

[TLN21] George Tzanetakis, Jakob Leben, and Steven Ness. Marsyas github. `https://github.com/marsyas/marsyas`, 2021. [Online; accessed 26-February-2021].

[Tov44] Donald Francis Tovey. Essays in musical analysis. 1944.

[TRW17] Babak Toghiani-Rizi and Marcus Windmark. Musical instrument recognition using their distinctive characteristics in artificial neural networks. *arXiv preprint arXiv:1705.04971*, 2017.

[TSC+21] Steve Tjoa, Leigh Smith, Owen Campbell, Mark Goldstein, and Sudara. Notes on Music Information Retrieval. `https://musicinformationretrieval.com/index.html`, 2021. [Online; accessed 25-February-2021].

[Tza07] George Tzanetakis. Marsyas submissions to mirex 2007. *MIREX 2007*, 2007.

[USG14] Karen Ullrich, Jan Schlüter, and Thomas Grill. Boundary detection in music structure analysis using convolutional neural networks. In *ISMIR*, pages 417–422, 2014.

[Wat02] John Watkinson. *An introduction to digital audio*. Taylor & Francis, 2002.

[Wik21] Wikipedia contributors. Tempo, 2021. [Online; accessed 12-May-2021].

Appendix

I. Glossary

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Villem Tõnisson**,
*(*author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Automatic Description of Music in Natural Estonian,
   *(*title of thesis)

   supervised by Sven Aller.
   *(*supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Villem Tõnisson
dd/mm/yyyy