Zurich University
of Applied Sciences

School of
Engineering
CAI Centre for
Artificial Intelligence

Project work (Computer Science)

Universal Songbook: Real time chord prediction from live audio

Authors	Kevin Kläger Guilherme Vicentini Urban Lutz
Main supervisor	Prof. Dr. Thilo Stadelmann Prof. Dr. Matthias Rosenthal

Date	24.12.2021

Zurich University
of Applied Sciences

School of Engineering

DECLARATION OF ORIGINALITY

Project Work at the School of Engineering

By submitting this project work, the undersigned student confirm that this work is his/her own work and was written without the help of a third party. (Group works: the performance of the other group members are not considered as third party).

The student declares that all sources in the text (including Internet pages) and appendices have been correctly disclosed. This means that there has been no plagiarism, i.e. no sections of the project work have been partially or wholly taken from other texts and represented as the student's own work or included without being correctly referenced.

Any misconduct will be dealt with according to paragraphs 39 and 40 of the General Academic Regulations for Bachelor's and Master's Degree courses at the Zurich University of Applied Sciences (Rahmenprüfungsordnung ZHAW (RPO)) and subject to the provisions for disciplinary action stipulated in the University regulations.

City, Date:

Winterthur, 29.12.2021

Signature:

The original signed and dated document (no copies) must be included after the title sheet in the ZHAW version of all project works submitted.

Zurich University
of Applied Sciences

Abstract

Every musician knows how hard it can be to play or accompany an unknown song for the first time. It is even more difficult to predict which chords should be played based only on a sung melody. Therefore, we propose a prototype that helps novice musicians to acquire the arduous skill of chord prediction during their learning process. In addition, we present an end-to-end solution, offering a real-time chord prediction. To create this solution, we assembled different architectures, with which we performed experiments that showed our model is still imperfect. Moreover, our model proved to be impractical for real-time chord predictions. Nevertheless, an alternative application to our model could be in the composition of new songs where ideas do not demand strictly correct predictions. Finally, we see potential for further development and overall improvement of our model to reach its main purpose.

Preface

We would like to express our gratitude to Prof. Dr. Thilo Stadelmann and Prof. Dr. Matthias Rosenthal for their excellent support during our thesis. We have gained plenty of insights from their valuable inputs. Likewise, with their assistance during our meetings, we were able to move forward step by step and finish our thesis. We also would like to thank all the music students and their music professor for participating in our interview with their sincere opinion.

Table of Contents

1 Introduction

Aspiring musicians often develop a form of musical intuition to guess which notes or chords follow next in the sequence based on what they have heard previously [1]. However, this skill must be learned, and it requires several years to develop [2]. The difficulty to play a song by guessing alone, without having sheet music available, is quite high, especially for novice players [2]. An example situation where the skill of guessing chords can be utilized is a musician at the campfire. The musician plays a song where he/she knows the melody and lyrics but might not know the correct chord sequence by heart. In these situations, musicians usually apply their musical intuition they had acquired over the years to guess which chord should be played next. This ability to play songs without knowing the chords is what we aim to ease for novice musicians by developing a *universal songbook*.

The chord prediction task can be solved by a composition of solutions to established problems in the domain of Music Information Retrieval (MIR) [3]. Although, there are multiple established solutions for the different tasks of MIR, there are no substantial solutions that target the learning of musical intuition of beginner musicians by predicting chord sequences in real-time. For this reason, we tried to build the *universal songbook*, by answering the question: *How can we create a general chord predictor with artificial intelligence that will ease the learning process of novice musicians?*

In this thesis, we propose two specific architectures, one of which with two variants, that combine several approaches from recent research to solve the problem of real-time chord prediction based on melody input. Our solution can process music and predict the future chord in a real-time and continuous manner, requiring no manual interaction of the musician while the song is performed.

In the following chapters we first present a summary of the state-of-the-art solutions for the MIR domain that are relevant for this thesis such as music transcription, music source separation, melody harmonization and others. Next, we compile our selection of existing solutions and libraries to build our proposed architectures based on various aspects, for instance the simplicity of the integration of those solutions. Afterwards we introduce each architecture and its benefits for our problem and explain, why some of the proposed architectures are currently not qualified. Subsequently, we present a series of results from experiments and our evaluation regarding the main goal. Finally, we take a step back and analyse the entire implementation with our conclusions and propose further improvements to achieve better results.

We would like to mention that the prototype was conceptualized and built over a single semester as part of a research project assignment at ZHAW School of Engineering. Therefore, to limit the amount of data and time needed, we focus on the genre of popular contemporary western music. Moreover, some of the music theory will be mentioned in this thesis but not explained, as we assume that the reader is familiarised with musical terms and conventions. Clendinning and Marvin [4] provide a comprehensive introduction to the music theory used in this project.

2 State-of-the-art Music Information Retrieval

MIR deals with the task of retrieving information from music. The goal is to enable several down-stream tasks in the field of musicology, like source separation, genre classification, melody harmonization or automatic music transcription [5].
For the task of this thesis, several MIR sub-tasks can be utilized to achieve the goal. The following chapters go into more detail on individual parts of MIR and what state-of-the-art solutions exist that are relevant for this thesis.

2.1 Automatic Music Transcription

The goal of automatic music transcription (AMT) is to produce a sheet music equivalent of a recording. It is considered one of the most challenging problems within MIR [6].
AMT can be further divided into two distinct sub-tasks, monophonic and polyphonic transcription. While monophonic transcription is considered resolved, there is no general solution to polyphonic transcription [6].

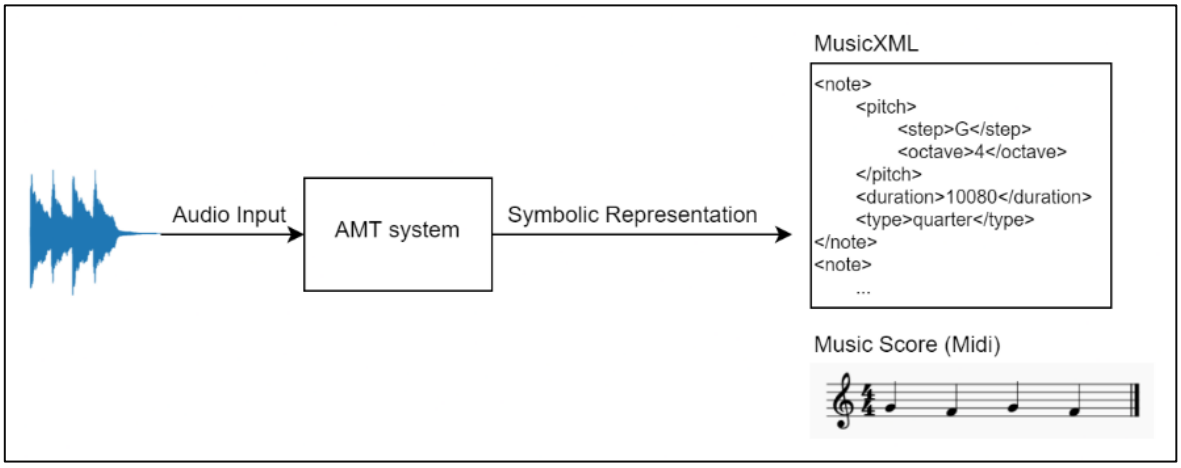

Figure 1: Visualisation of an AMT system with possible outputs of symbolic representation (not a complete list).

2.1.1 Monophonic Music Transcription

Extracting a monophonic note sequence from audio can be used to extract the melody of a song. This melody typically represents what a listener could hum or whistle when asked to do so, as it is representative of the song and often closely follows the singing voice or other important melodic motives.

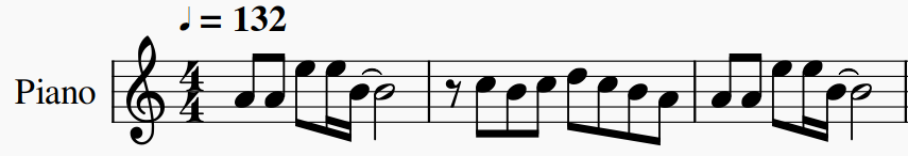

Figure 2: Visualisation of monophonic music in form of a music score. Example song: ATC - All Around The World from https://github.com/wayne391/lead-sheet-dataset [7]

Most instruments when playing a note do not produce a single perfect pitch but a collection of pitches at different frequencies [8]. The frequencies higher than the frequency of the actual note played are what is known as overtones. The differences in the overtones, along with the presence of

noise, are what makes instruments recognisable and is denoted as the timbre of an instrument [9]. Overtones, while being quieter than the foundational pitch, can become a problem when trying to retrieve the frequency of the note, especially when multiple notes are present and overtones overlap. Monophonic pitch trackers are often less affected by overtones present in the source audio, as they follow a single pitch, at the frequency highest in volume [10].

Convolutional Representation for Pitch Estimation (CREPE) is a state-of-the-art pitch tracker [11], based on a deep convolutional neural network (CNN) [12]. It outperforms other pitch tracking algorithms, which use heuristic approaches like PYIN [13] or SWIPE [14]. On their compiled dataset with a small error margin, CREPE has an accuracy of 90.9% and is 8 percentage points higher than PYIN with 82.6%. To measure its performance, Kim et al. [11] used the 'raw pitch accuracy' [15] [16].

Raw pitch accuracy is the proportion of correctly estimated pitches to the ground truth pitches within some error margin. The error margin is measured in cents, which represents musical intervals relative to a defined reference pitch (in Hz) [11]. A common value is 50 cents (quarter tone).

2.1.2 Polyphonic Music Transcription

Polyphonic music transcription is a more challenging problem than monophonic music transcription [17] [6] since a transcriber must be able to differentiate between multiple sources (instruments or vocals) that are combined in one audio signal. Even for a trained musician, this is no easy task [18]. In addition, algorithms working well for monophonic transcription, do not work as good for polyphonic transcriptions [6].

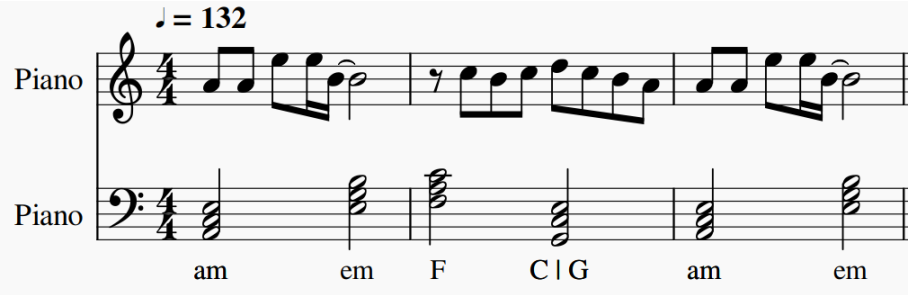

Figure 3: Visualisation of polyphonic music in form of a music score. Example song: ATC - All Around The World from https://github.com/wayne391/lead-sheet-dataset [7]

Research published a variety of different solutions for polyphonic music transcription. There are models that transcribe one type of instrument. Various models were developed for polyphonic piano transcription [19] [20] [21] [22], or for other instruments, for example guitars [23].
Other types of models transcribe multi-instrument audio [24] [25] [26] [27]. However, their output can be different. For example, ReconVAT [25] only outputs a single pianoroll, without assigning instrument labels for a note. MT3 [24] on the other hand, outputs a *Midi-like* format with separate tracks per instrument.

Besides the complexity and solution variety of polyphonic transcription, papers use different metrics, or use different names for the same metrics. This makes comparisons between papers difficult [24]. The following list, summarised from [24], gives a short introduction to commonly used metrics (we recommend Cheuk et al. [28] for a detailed analysis of these scores).

- **Frame-wise F1 score**: Audio data is split into frames. A matrix [nr of frames, nr of notes] defines, which notes are played at which frame. A column at frame x would be a true positive, if all the active notes correlate to the ground truth at frame x.

- **Onset F1 score (note-wise F1)**: The predicted pitch needs to be within some time limit. The value most used is 50ms.
- **Onset-offset F1 score (note-with-offset-wise F1)**: Notes must match the onset metric, as well as having matching offsets. The offset must also be within some time limit. A usual value for this is 20% of the note-duration or 50ms. Onset-offset F1 score can be considered the strictest metric of this list.

Multi-Task Multitrack Music Transcription (MT3) [24] uses a T5 model [29] and is capable of multi-instrument polyphonic transcription. The output of MT3 is a symbolic representation, which they call *MIDI-like*. For training and evaluation, Gardner et al. [24] combined multiple existing datasets and compiled a larger dataset from them. On all six datasets it was evaluated on, MT3 compares to, or exceeds previous state-of-the-art models (sometimes up to 260% relative gain) in frame-wise F1, onset F1 and onset-offset F1 metrics (for more info, see paper [24]). The paper for MT3 was submitted in November 2021, during our project. Therefore, we did not consider its implementation for this project.

According to Benetos et al. [17] non-negative matrix factorization (NMF) [30] [31] can also be considered state-of-the-art for polyphonic transcription. NMF remain relevant, where Neural Networks are facing challenges:
Lack of annotated datasets: Only relatively small, annotated datasets exist for polyphonic music transcription, consisting of several hours of audio. Because of this, Neural Networks tend to overfit the datasets. Since the paper released however, larger datasets have become available. Gardner et al. [24] compiled and proposed a dataset for polyphonic music transcription consisting of 6 existing datasets, resulting in over 1000 hours of audio. Compared to other audio tasks like automatic speech recognition with datasets of over 2500 hours [32], datasets for polyphonic music transcription remain small.
Adaptability to other acoustic conditions: NMF has the capability to improve performance for new instruments by iterating on only a few examples. Neural Networks have no equally effective mechanism to adapt as fast as NMF [17].

2.2 Music source separation

The goal of music source separation is to split musical information into separate parts, for example isolating vocals from a song or separating instruments. Separating music into its parts can greatly improve the performance of down-stream tasks, like automatic transcription, as information that is not needed for the task, for example drums, can be filtered out [33].

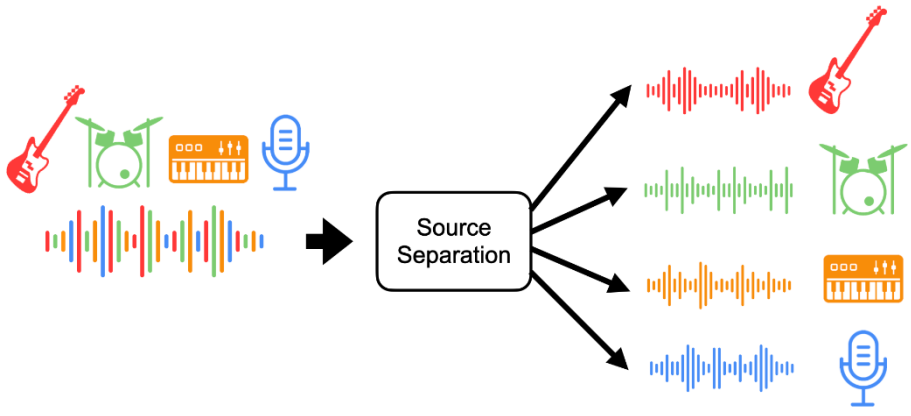

Figure 4: Visualisation of music source separation task. Image from https://source-separation.github.io/tutorial/landing.html [34]

Recent music source separation papers operate on different domains to solve the task (waveform, spectrogram, or both). Therefore, papers use different Neural Network architectures for their models [35]. Popular architectures for music source separation are CNN [35] [33], bidirectional long short-term memory (biLSTM) [36] or a combination of them [37].

The current state-of-the-art model for the MUSDB18 dataset [38] is Hybrid Demucs [37], which is an improved form of the Demucs architecture [36]. It outperforms other models and achieves an average signal to distortion ratio (SDR) of 7.68 decibels (dB). This is an improvement of around 1.4 dB.
Like MT3 in chapter 2.1.2, the paper for Hybrid Demucs was released during our thesis and was not considered for implementation.

Signal to distortion ratio (SDR): According to Défossez et al. [36], SDR is the log ratio between volume of the estimated source projection onto the ground truth, and the volume of what is left out of this projection (e.g., contamination by other sources or artifacts).

$$SDR := 10\,log_{10}\left(\frac{\|s_{target}\|^2}{\|e_{interf}+e_{noise}+e_{artif}\|^2}\right)$$, original paper from Vincent et al. [39]

2.3 Melody harmonization

The goal of melody harmonization is to generate harmonic context to a monophonic melody. In musicology, melody harmonization is highly subjective and there are often several fitting chord sequences [40] for a given melody.

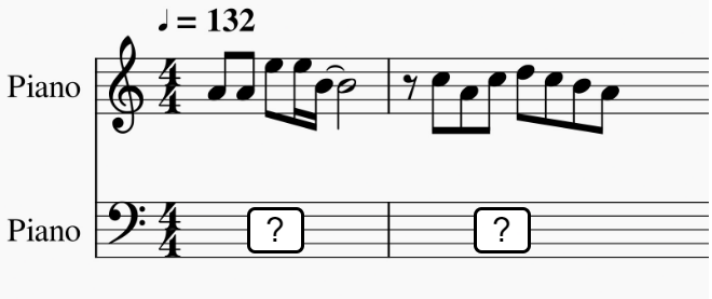

Figure 5: Visualisation of melody harmonization task in form of a music score. Example song: ATC - All Around The World from https://github.com/wayne391/lead-sheet-dataset [7]

When building a melody harmonizer, studies often use the original chords of a given song as the ground truth of what the chord progression should be, but also collect qualitative opinions of how well the chords are perceived to fit the melody by musicians and non-musicians [40].

When comparing different algorithms to harmonize melodies, for instance deep-learning based and non-deep-learning based algorithms, the deep-learning based models, such as biLSTM [41] and *deep Multitask models*, proved to be better than non-deep-learning based algorithms, in particular Hidden Markov Models (HMM) [42] and Genetic Algorithms (GA) [43], in a variety of features, especially in *harmonicity* and *interestingness* [7].

Sun et al. [40] proposed a melody harmonizer based on biLSTM and applies orderless NADE [44], chord balancing (i.e., class weighting) and blocked Gibbs sampling [40]. It outperforms the previous state-of-the-art model in a subjective study with more than 100 participants in the metrics of "*coherence*", "*chord progression*" and "*interestingness*". The model has close results to human

composers and even gains a higher score in *chord progression* compared to human composers (3.53 for the model, 3.47 for human composers).

2.4 Chord sequence prediction

Chord sequence prediction is the task of guessing the next chord based on a sequence of previous chords. [45]. This task is similar to predicting the next word of a sentence in the domain of Natural Language Processing (NLP). The concept of words and sentences can be adapted to music by interpreting a chord token as a word and a sequence of chord tokens as a sentence [46].

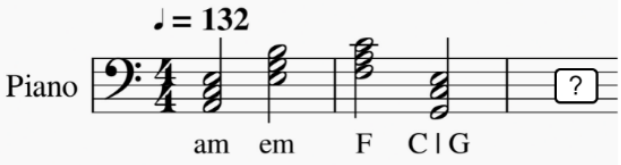

Figure 6: Visualisation of chord sequence prediction task in form of a music score. Example song: ATC - All Around The World from https://github.com/wayne391/lead-sheet-dataset [7]

To process chord tokens, the tokens must first be converted into a numerical representation. Based on the NLP concept of word embeddings, Madjiheurem et al. [46] show that meaningful vector representations of chords can be obtained for chords as well. In [47], the authors obtain chord embeddings from a corpus of classical music by using the skip-gram methodology. Using these chord embeddings, they implement a LSTM-based neural network, which is trained to predict the next chord token in the sequence.

2.5 Music Generation

Music generation has very different set of goals than most other tasks within MIR [48]. Instead of purely predicting melodies or chords by their statistical likelihood, music generation aims to replicate the creativity of a musician writing a completely new piece of music. Therefore, in the domain of music generation, the quality metrics are much different. For example, when generating a new piece of music, qualitative attributes like “interestingness” or “pleasantness” are important, alongside a measure of how "creative" it is, indicating to what level the piece was surprising or bland [49].

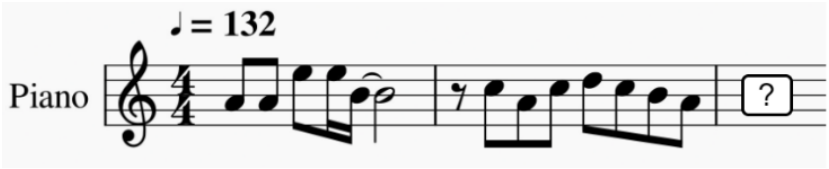

Figure 7: Visualisation of music generation task, specifically melody generation, in form of a music score. Example song: ATC - All Around The World from https://github.com/wayne391/lead-sheet-dataset [7]

Generative models can also be used to complete unfinished melodies or create complete songs from only melody. This is comparable with what the task of melody harmonization is trying to achieve, the ingredient that differentiates music generation from other subtasks is the element of creativity. The creative element makes music generation non-deterministic [49].

Musicautobot [50] is a multitask model based on Transformer-XL [51] that is also suited to do music generation. It uses a feature named *temperatures*, to determine how much randomness in the pitch

and rhythm should be added in the generation of melody or harmony. A further step would be to combine Transformers with Generative adversarial networks (GAN) [52] to improve the music creation performance [53]

2.6 Normalisation

While not technically being a task specific to MIR, music normalisation is an important step to western musical analysis [54]. Especially normalizing the keys of the piece is useful for further processing, as downstream systems only ever have to know a single key. This is achieved by detecting the tonal centre of the piece and transposing it to a default key of C Major. This helps to counteract the fact that in modern music, certain keys are much more common than others [55]. Normalizing to C Major makes the key of the song a non-factor, given the right key was detected.

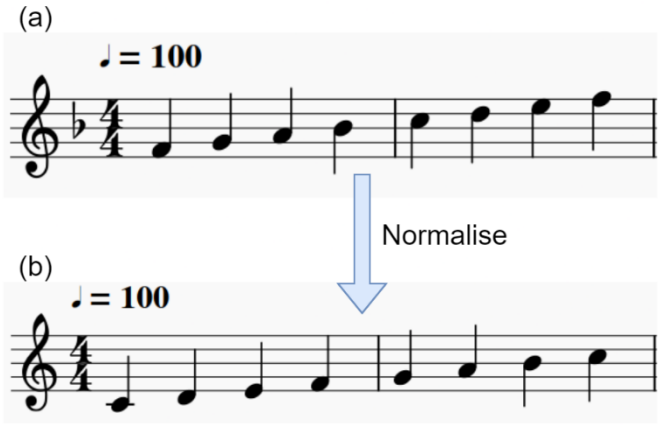

Figure 8: Visualisation of normalisation task in form of a music score. (a) melody in F major Key. (b) transposed melody in C major key.d

Key detection can be implemented using different strategies ranging from simple algorithms like matching the notes in the melody to scales, to highly complex systems able to take handle diatonic modes (where a scale is used with a tonal centre different to its root) and shifting keys in modulations [55].

3 Proposed System Architectures

We propose two system architectures, capable of solving the chord prediction problem as stated in chapter 1 by combining a selection of available solutions from the MIR domain. We emphasize that the architectures are based on our ideas but use publicly available components. We integrate and connect these components together to solve the task.
The system architectures are designed with modularity of components in mind. If research proposes a new solution to any subtask, we can substitute that component or we can observe how the performance of the system overall develops when using different implementations.

The following section first describes which tools and libraries are used for which subtask. Then, we explain the general idea behind the system architectures and what components they use. The system architectures are described to work on single audio files, so we could evaluate and test them in this non-real-time setting, because our dataset [7] consisted of MIDI files with split melody and chord

tracks. This ensured us repeatable and verifiable experiments. The last section in this chapter then focuses on transferring the system architectures to work in a real-time setting.

3.1 Tools and libraries used

This section provides an overview of the tools and libraries we chose and explains why we chose them. Each system architectures then uses and combines a selection of them.

Symbolic representation: We decided to use music21, because it simplified development, if most of the components work with music21 objects. In addition, music21 provides a variety of tools for analysing, searching, and transforming music in symbolic form [56].

Normalisation: Because the symbolic representation is encapsulated in music21 objects, we can also use the music21 transpose function to normalise the music score. A concrete implementation of key detection is included in music21 in the form of the Krumhansl-Schmuckler algorithm [55].

Polyphonic music transcription: For polyphonic transcription, we decided to use ReconVAT. ReconVAT is a semi-supervised deep learning model and aims to generalize better to musical genres, that are not present in the training data [25]. While ReconVAT is not state-of-the-art, we chose it as the source code is publicly available and because it can deal with a multitude of instruments, not just piano.

Monophonic music transcription: We decided to use music21 monophonic transcription for this, because it was easy to implement, as we already use music21 for other tasks and its return value can directly be used for further processing. Another option would have been a pitch tracker that achieves high scores in pitch tracking but requires post-processing steps. The frequency per timestep would need to be converted to symbolic representation (music21 stream) by creating notes from the frequencies and define their duration. With music21 monophonic transcription, we could skip these steps. However, music21 does not present performance measures for this feature.

Music source separation: Spleeter is a tool released by Deezer, an online music streaming service, containing pretrained models to split audio into different stems [33]. While newer models with better performance have been published [37], we found Spleeter to be easy to use and, as it can use GPUs, performs well.

The pretrained models are U-Nets [57] which is an encoder/decoder CNN architecture with skip-connections which estimate a filter mask for each source it separates. This is different from the approach previously taken by previous authors, where the model directly generates waveforms corresponding to the different sources [33].

Chord sequence prediction: To predict which chord follows in the sequence of previously observed chords, we use a model developed by Azuoni et al. [47] in the context their work measuring the prediction accuracy across different composers and eras. We decided to use this model as it represents recent research, and the code was publicly available.

Melody harmonization & Melody generation: We decided to use musicautobot, because it provides a multitask model capable of melody harmonization, melody generation and some more functions. Instead of using multiple models with different results, one model working with music21 objects as input and output made it simple to integrate in our system.

3.2 Polyphonic Listener (Architecture A)

The System Architecture A works under the assumption that both melody and harmony information are available, hence its name “Polyphonic Listener”. In case of the guitar player at the campfire, Architecture A would listen to both the guitar and the vocals, processing both parts separately. If the guitar player remembers some of the chords and plays them, the played chords will be considered. If the musician does not remember any chord at all, he/she can sing the melody and start to play the chords as they are predicted. In that sense, previous predictions are considered, allowing Architecture A to build connected chord sequences. Which chord the musician plays however is his/her own decision, allowing to correct wrongly predicted chords and improving overall accuracy. Combined with the processing of the melody, we hope to improve accuracy further as we can detect, when the song might be about to move to a different chord pattern than previously observed.

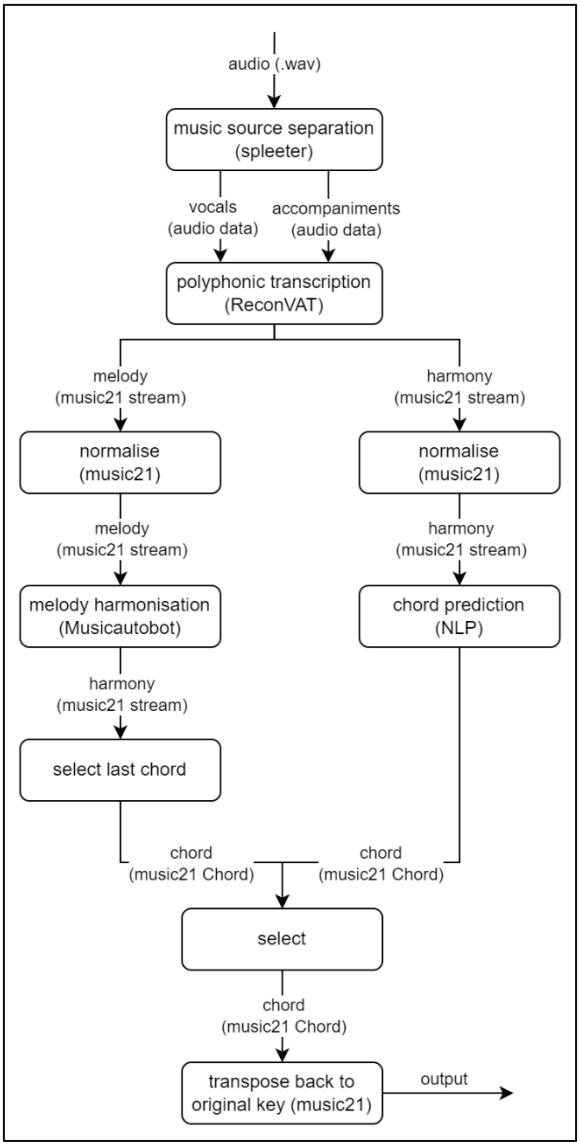

Figure 9: Structure of Architecture A with polyphonic transcription

A distinct feature of Architecture A is the parallel evaluation of two separate prediction systems, yielding separate results, on which a voting algorithm then selects a result based on the confidence of both predictions.

This architecture depends on being able to split audio into harmony and melody parts and the accurate transcription thereof. While splitting audio works [33], it still creates audible artefacts which, in combination with the shortcomings of state-of-the-art polyphonic audio transcription (see chapter 2.1.2) renders the resulting transcription unsuitable for further processing.

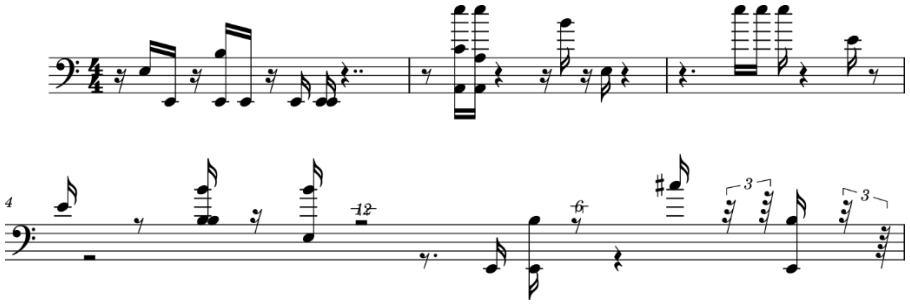

Figure 10: Polyphonic transcription with ReconVAT on an accompaniment separated by Spleeter

Figure 10 shows an excerpt of a transcription with ReconVAT on an accompaniment that was separated by Spleeter. Often, single Notes are detected instead of chords and the notes span a large frequency range.

Due to these shortcomings, Architecture A was not pursued further and was not considered for further testing and evaluation.

3.3 Monophonic Listener (Architecture B)

Architecture B extracts only melody information from audio data via monophonic transcription. Because monophonic transcription is less complex than polyphonic transcription (see chapter 2.1) this approach could yield better symbolic representations, when used on audio data.
With this solution, a guitar player can sing the melody and / or play the chords. In contrast to Architecture A, played chords from the musician are not considered, as only the melody serves as input.

We decided to implement and evaluate two variants of the monophonic listener architecture, using results from previous work with publicly available pretrained models. We discuss other possible solutions in chapter 5.

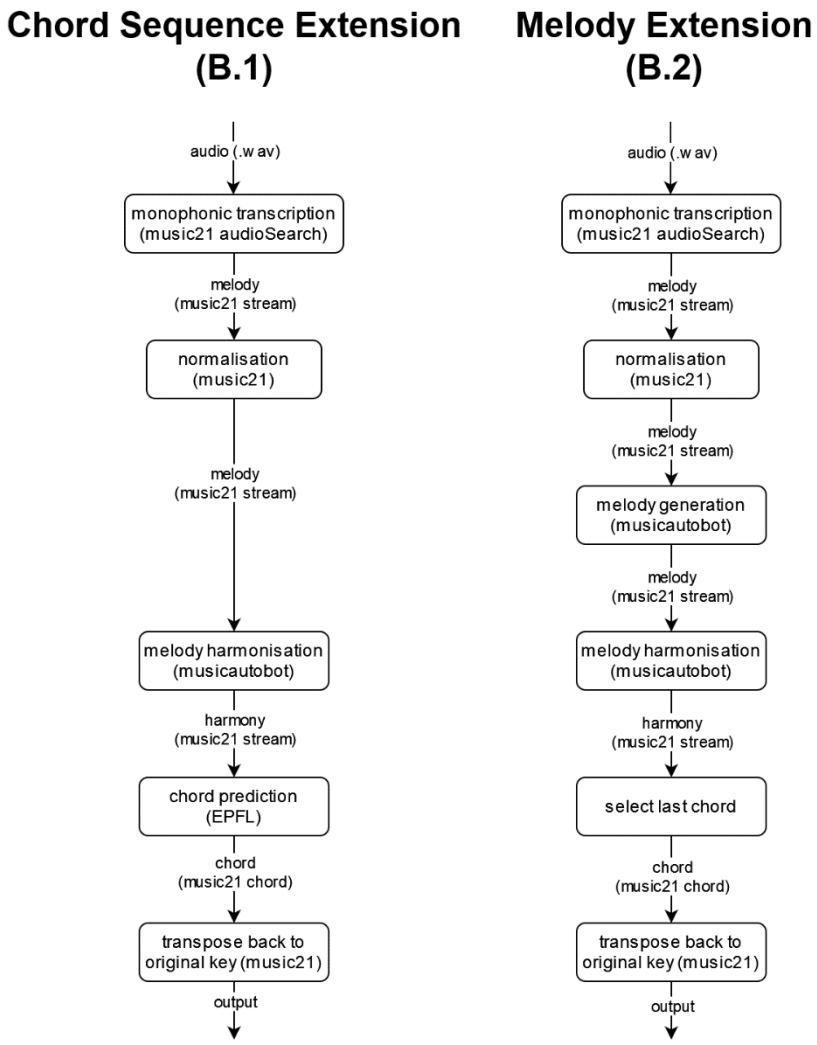

Figure 11: Chosen variants for Architecture B

The Chord Sequence Extension model (referred to as “B.1”) uses a melody harmonizer and chord prediction system sequentially. From the symbolic representation of the monophonic transcription, the melody harmonizer generates a chord sequence. The harmonized chord sequence is used to predict the next chord. It is implemented as a combination of both the multitask model from musicautobot for harmonization and a LSTM based prediction model.

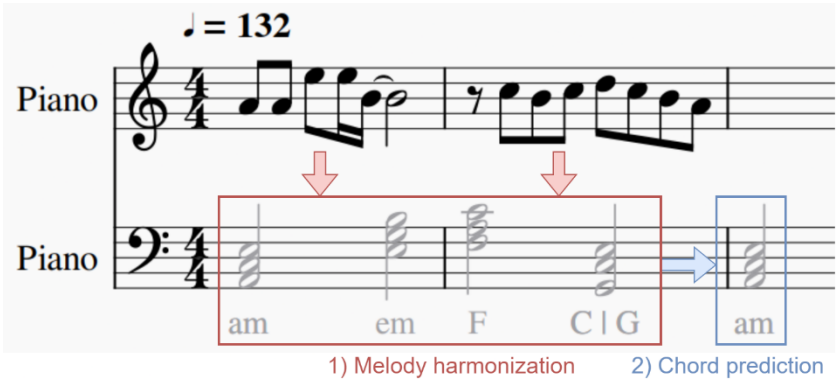

Figure 12: Visualisation of the core idea behind the Chord Sequence Extension Architecture in form of a music score.

The Melody Extension model (referred to as “B.2”) differs in approach by removing the explicit use of chord prediction. Instead, it works by leveraging different features from the multitask model of

musicautobot. Melody generation completes the yet unobserved melody. Then the melody harmonizer harmonizes both the real and the generated part to get a chord sequence. The first chord of the last bar of the harmonization is interpreted as the predicted next chord.

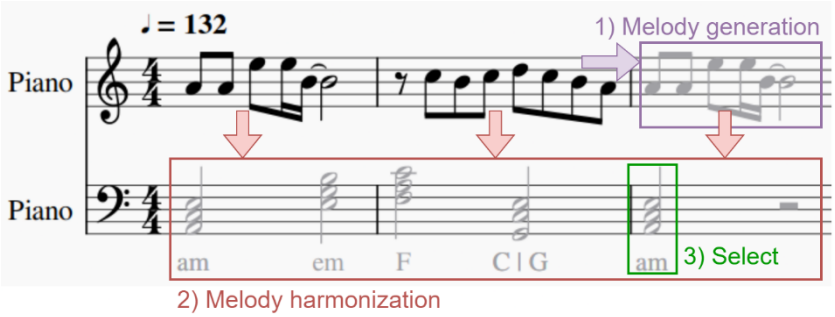

Figure 13: Visualisation of the core idea behind the Melody Extension Architecture in form of a music score.

3.4 Real-time implementation

To adapt our architectures to real-time prediction, we changed our input interface to continuously record audio data. The new interface transcribes the recorded audio every second to a music21 stream and appends this transcription to a list. Meanwhile, if the list is filled with at least eight notes the monophonic listener retrieves the last eight notes from it and outputs the predicted chord until the process is stopped.

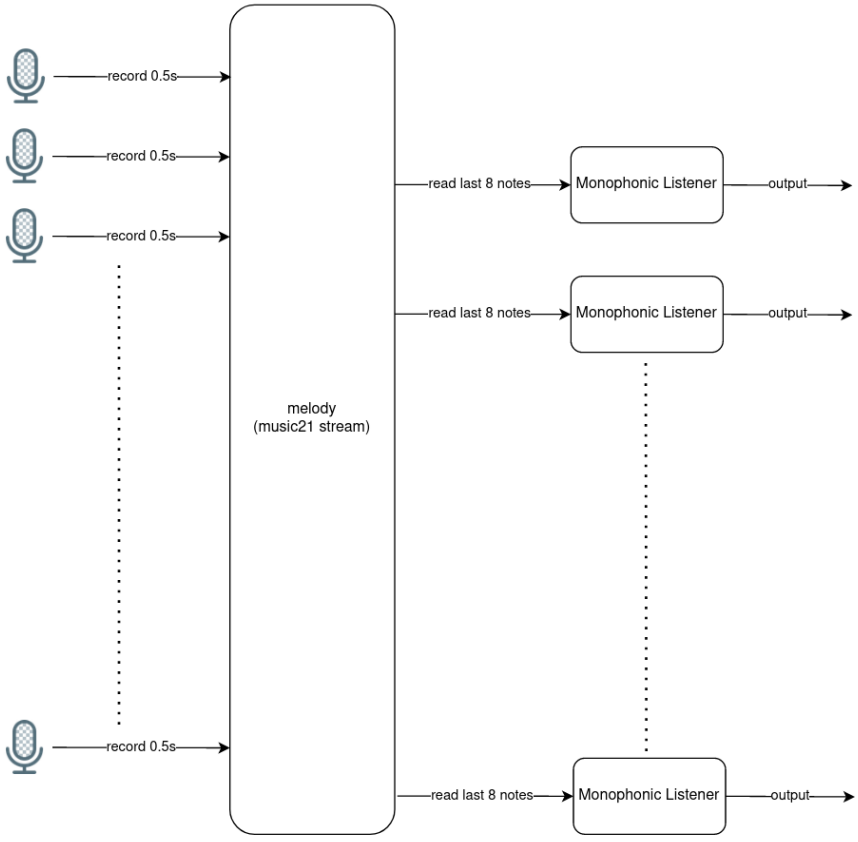

Figure 14: Chord prediction transferred in a real-time context

4 Experiments and Results

In this section, we present a series of experiments that were performed to measure different aspects of our solution. First, we interviewed seven musicians to assess the overall acceptance of our implementation. Second, we performed a ceiling analysis over our architectures to establish, which components could be enhanced to raise the overall performance. Finally, we measured the processing time of components in our proposed architectures to estimate the feasibility of real-time prediction.

4.1 Interview with musicians

A survey with six music students and one music professor was made to measure the acceptance, interest, and the best use case for our model. We asked them to rate the transcription from audio to symbolic representation and what they thought was the best use case for our tool, for instance, helping with the creation of new songs or helping during the process of learning a new song. We also presented them a sequence of chords and asked them to guess what the next chord should be. First, they were asked to guess without any help, and then they were presented with the chord predicted by our solution to help them in their own decision. With that, we could assess how musicians feel about being assisted by our tool.

To be able to make a qualitative statement about the monophonic transcription, we showed the participants the same melody played in four different ways, such as whistle, sung, played with an acoustic guitar and played on the piano, as these are possible instrument sources for our transcription. We choose to play a melody from Elton John - Can You Feel the Love Tonight. Below are 3 bars of the melody retrieved from our dataset followed by our model's transcription of the original melody played on the piano.

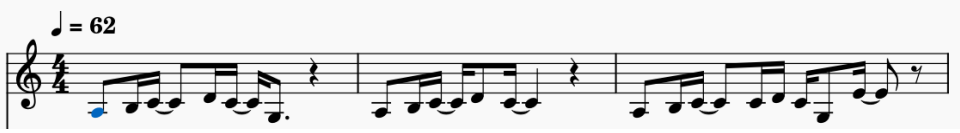

Figure 15: Original melody (ground truth) of Elton John - Can You Feel the Love Tonight from https://github.com/bearpelican/musicautobot [50]

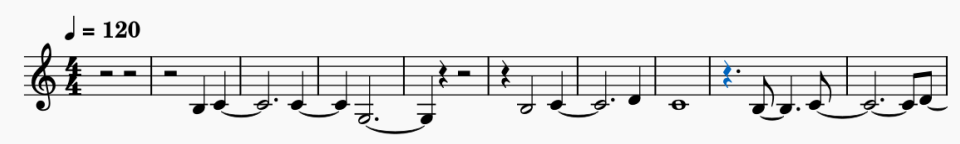

Figure 16: Monophonic transcription (music21) of Elton John - Can You Feel the Love Tonight, when played on a piano.

Just from comparing the pictures, we can see that both melodies are not the same. This was also the opinion of all interviewed participants. The majority found the transcribed melody not similar at all with the original melody, regardless of how the audio was recorded.

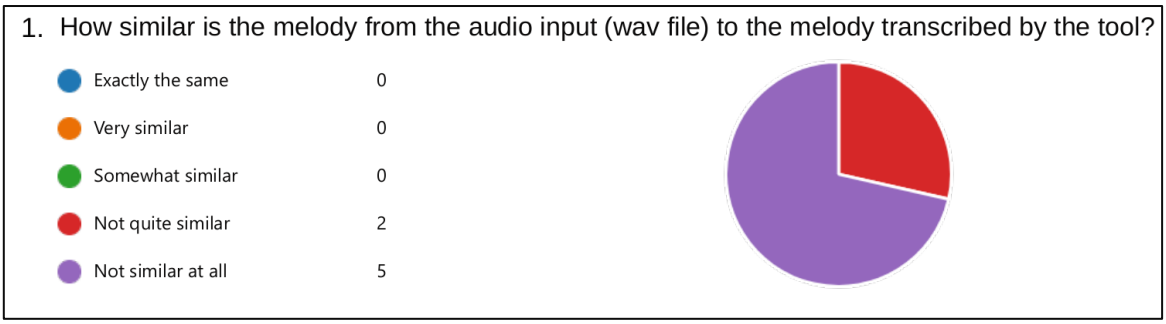

Figure 17: Interview results - evaluation of the similarity of the transcribed melody

When asked if the tool helped them to predict the next chord of a given chord sequence, the results were divided, three of them found it somewhat helpful, three neither helpful nor unhelpful. The professor found it to be somewhat unhelpful.

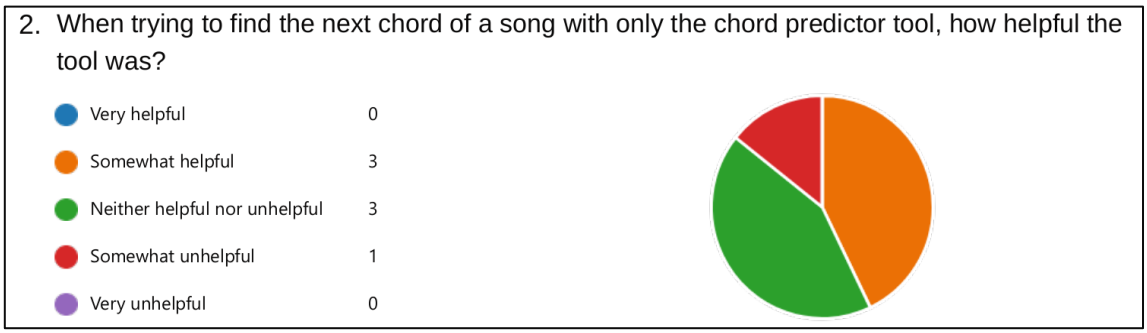

Figure 18: Interview results - participants opinion about chord predictor

Four out of seven participants would use the tool rather for the music creation process than for the learning process. The professor would not use the tool at all. The reason was that the tool was not able to correctly transcribe the played melody, implying that this would lead to wrong predictions.

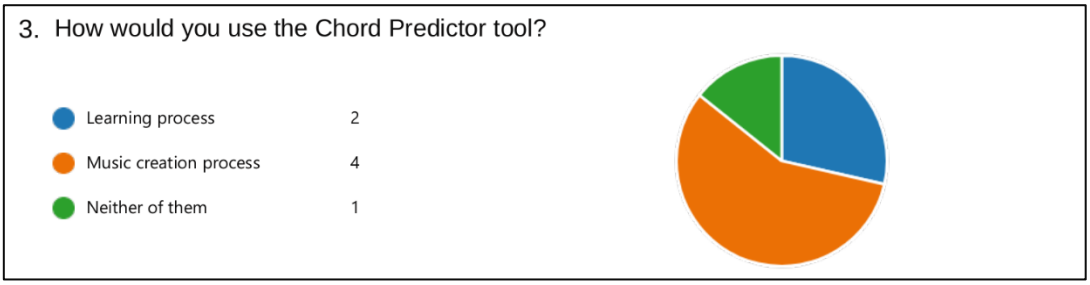

Figure 19: Interview results - chord predictor use case evaluation

Discussion interview with musicians

As a result of this interview, we see that the acceptance of our model as a chord predictor was low, and that the monophonic transcription has almost no similarity to the played melody.

This shows us, that the usability could be improved greatly by the implementation of a better monophonic transcriber. A state-of-the-art pitch tracker could be used, even though we would have needed to implement post-processing steps to get a symbolic representation from the frequencies the pitch-tracker detects.

During the creation process of a new song, creativity and different ideas, even “wrong” chords can be useful [49]. This reflects in our results, where participants showed interest in our model as a support tool for the creation of new songs.

4.2 Quantitative Evaluation

In this experiment, we evaluate the performance of both architectures on a dataset of MIDI song snippets. In order to test the pipeline starting from audio files, a dataset is required that contains audio as well as which chord is being played at what time. Such dataset, to the best of our knowledge, is not readily available. To simplify our requirements towards the dataset, we test both architectures without the audio transcription step, allowing us to only deal with symbolic information. Furthermore, we have already tested the quality of the pitch tracking in the preceding chapter and found it to be unreliable.

4.2.1 Dataset

We use the Hooktheory dataset compiled by Yeh et al. [7] by scraping Hooktheory.com, a website containing snippets of songs aiming to teach harmonic theory to songwriters. The snippets contain separate parts for melodies and chords, as well as the key signature and tempo information.

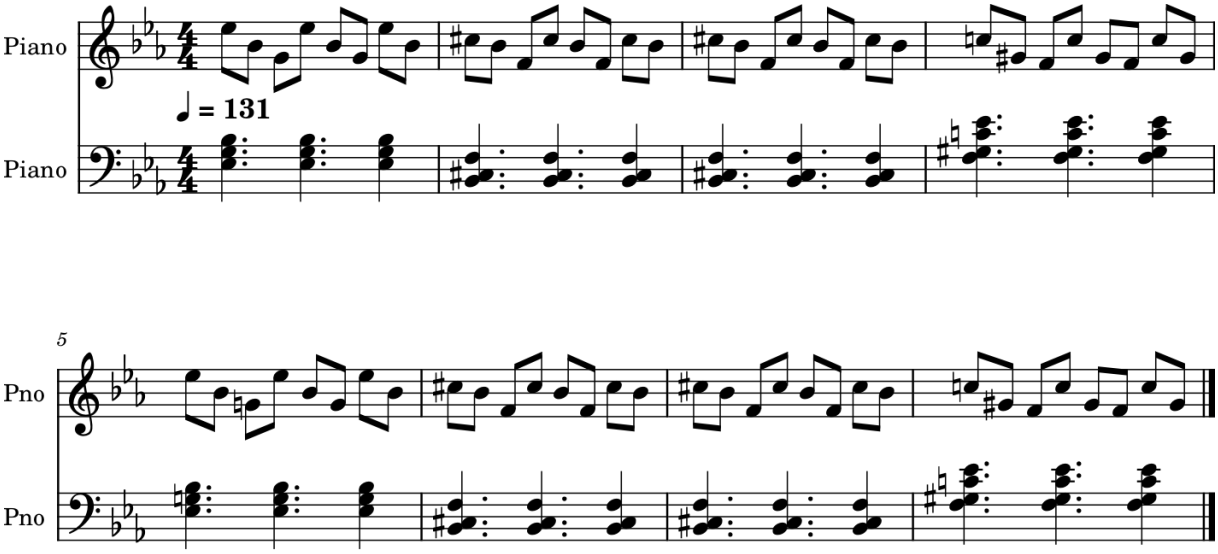

Figure 20: Example snipped of a music score. The piece is Clocks from Coldplay https://github.com/wayne391/lead-sheet-dataset [7]

From these song snippets, we consider the first four bars of the melody part to be our input. The cut-off at four bars was chosen arbitrarily. Choosing this length of input results in approximately 8 seconds of music, when calculating with an average tempo of 120 bpm and time signature of 4/4. Therefore, there should be a representative amount of musical information available to the predictor. The expected next chord of a snippet is then defined as the first chord in the fifth bar of the chord part. For this experiment, we evaluate a subset of 800 snippets from 537 distinct songs.

As can be seen on Figure 20: Example snipped of a music score. The piece is Clocks from Coldplay https://github.com/wayne391/lead-sheet-dataset , the dataset contains key signatures for each song snippet. However, we came to find that the key signature included in the data is not usable for our purpose, as it does not include information on the mode the piece is in. For example, the key signature of the example snippet shown in Figure 20 indicates a key signature of Eb. If we normalise to C using this information, the resulting notes would be part of the C mixolydian scale instead of the standard major scale, as the piece is not in Eb major but in Eb mixolydian.

To normalise to C major, we therefore cannot only read the key signature, but have to run a key detection algorithm to find the corresponding major key. For this purpose, we run the Krumhansl-Schmuckler algorithm included in music21 on the full snipped to get an accurate key we can use as a ground truth. The resulting mode of this analysis is always either major or minor. In case the returned key is minor, we treat the related major scale as the key resulting in all snippets to be normalised to the C major scale. Using this method, we can retrieve the correct key of Bb Major for the example in Figure 20.

The distribution of keys detected by this method resembles the distribution of keys found across the Spotify catalogue [58], with the top 3 keys being D-, G- and C major respectively.

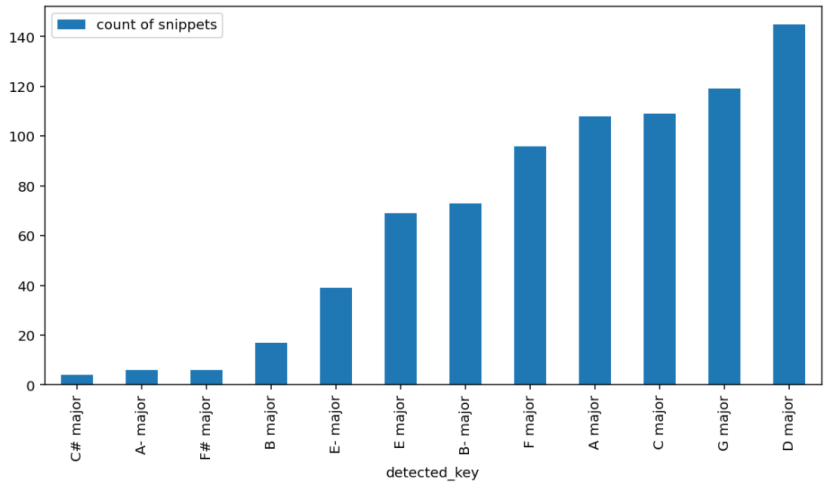

Figure 21: Distribution of keys across dataset

Overall, G major and D major are also the chords most often expected as a next chord. Not every chord in every key is part of the sample, but since the snippets will be normalised to the key of C major, all keys will be treated the same.

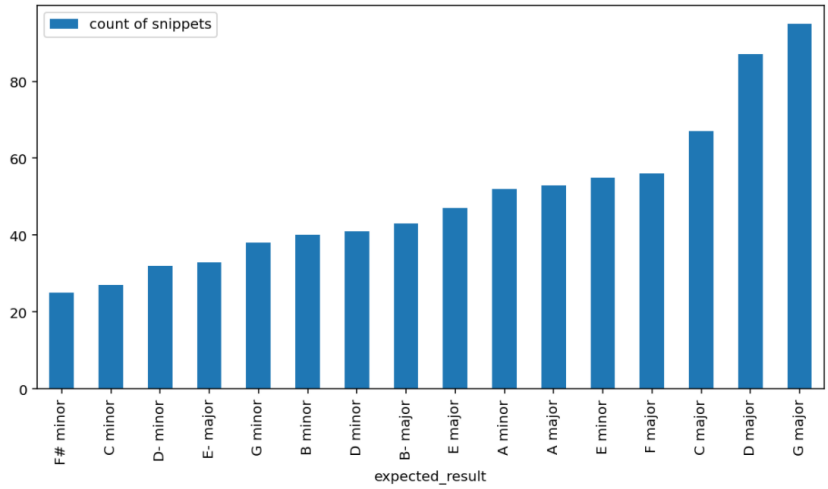

Figure 22: Distribution of expected next chords across dataset

Taking into account the key of the snippet, we can identify the role of the chord in relation to the key. This information is expressed as the number of steps the root of the chord is away from the key centre and notated in roman numeral form. Most chord progressions after four bars expect a I, IV, vi, ii or V as next chord. This is to be expected, as many songs revolve around chord changes around these chords [59].

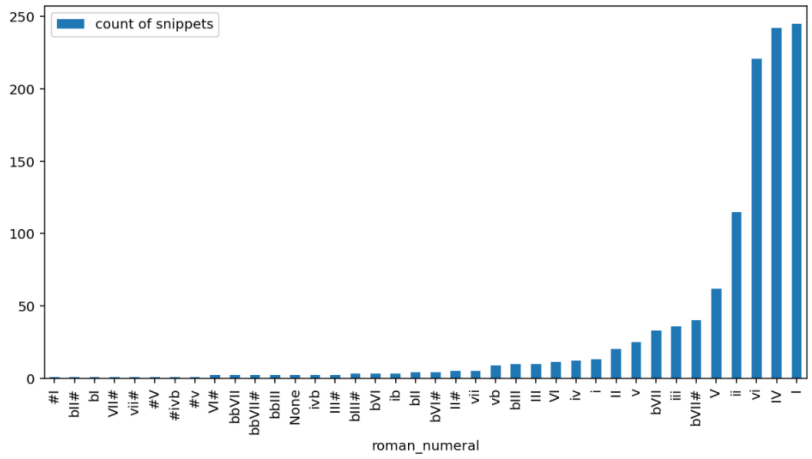

Figure 23: Distribution of chords across dataset, expressed in roman numeral form relative to the respective key

4.2.2 Chord distance functions

To measure the performance of both architectures, we use three heuristics measuring the distance between the predicted and the expected chord.

Note overlap

The first one is a simple measurement on how many notes both chords share. Since we simplify chords to triads, the expected and the actual result can at most have three notes in common. The expected range of this metric therefore is between 0 and 3, with higher scores implying higher similarity.

Using overlapping notes as a measurement of prediction precision has both benefits and drawbacks. The benefits are that it is simple to implement and understand, and that allows enough granularity to measure improvements. The drawbacks are, that this method has no notion of the importance of the individual notes.

Considering for example two pairs of chords, both having a note overlap of 2:

- A minor (A, C, E) and C major (C, E, G)
- C major (C, E, G) and C minor (C, Eb, G)

A minor and C major are closely related (C major being the major relative scale to A minor) while C major and C minor are less common to be featured in the same chord progression, as C minor features a note that is not present in the C major scale (Eb), yet both pairs score equally well on the Note overlap metric.

Circle of fifths distance (Co5 distance)

A more nuanced way to measure the distance between chords would be to measure distance on the circle, as described in [60] and [61].

Distance on the circle of fifths as defined by De Haas et al. in [60] is measured with the number of hops required to go from the predicted chord to the expected chord. Hops are valid in both directions. Going from the inner ring of minor parallels to the outer ring also counts as one hop. If both chords are minor, it is possible to stay on the inner ring of minor chords [60].

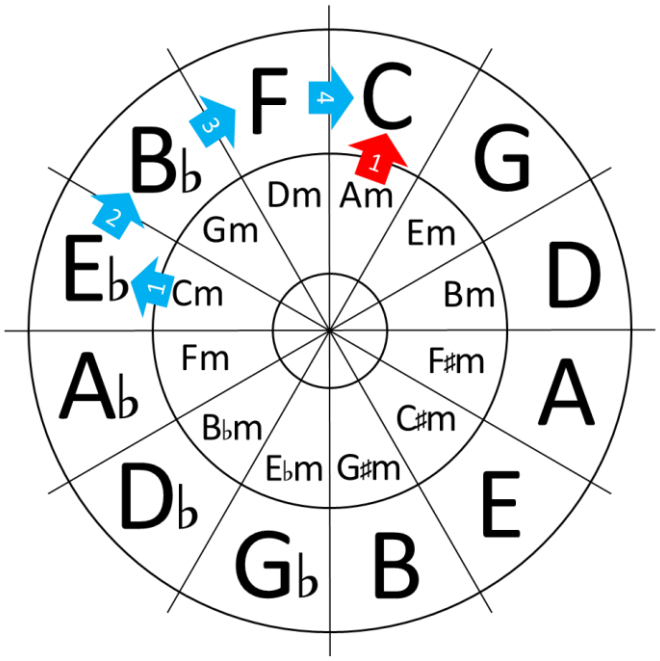

Figure 24: Distance between C minor and C major (in blue) and between A minor and C major (in red) on the circle of fifths

As shown on the visualisation, the two example chord pairs (A minor and C major and C major and C minor) result in distance scores of 1 and 4 when calculated by the Co5 distance function.
Compared to the simple note overlap heuristic, Co5 distance is harder to implement, because it must consider the different enharmonic spellings of chords. For example, a F# major chord in the context of G major might be called Gb major in the context of B major. These differences are carried over to the normalised key of C. For this purpose, we use the music21 library, which provides a function to normalise chord spellings into a specific key.
Two chords can at most be 7 steps apart (in the case of a major chord and its opposite minor chord), giving this measure an expected range of 0 to 7, with lower scores implying higher similarity.

Cosine similarity of chord embeddings

As a third way to measure the distance between the expected and the observed chord, we use the pretrained chord embeddings we already leverage in the context of chord prediction with the LSTM model as described in chapter 3.1. The embeddings have been trained on chord sequences represented by roman numerals and therefore depend on accurate key detection when converting chords to embeddings.

The two embedding vectors are then compared using the cosine similarity [62], as previous authors have done in the context of chord embeddings. [63]

Comparison

Using all three measurements allows us to cross-validate the results of each distance function.

	Note overlap	Co5 distance	Chord Embeddings
Note overlap	1.000000	-0.567374	-0.457582
Co5 distance	-0.567374	1.000000	0.806265
Chord Embeddings	-0.457582	0.806265	1.000000

Table 1: Correlation matrix between metrics

The correlation matrix between all distance functions shows a strong correlation between the cosine distance of the chord embeddings and the distance of the chords on the circle of fifths. Our interpretation is that the chord2vec model was therefore able to learn a meaningful representation of the musical concept we used in the Co5 distance function. The Note overlap function does not correlate as well with neither the Co5 distance nor the Chord Embedding distance. This is expected due to the shortcomings described above and therefore, we believe the other two metrics to be more relevant in our case.

To be able to compare the different metrics more easily, we normalise the results to a scale between 0 and 1, with 1 indicating strong similarities between two chords, and 0 implying no similarity.

4.2.3 Baseline performance

We measure the performance of both architectures on the dataset described in chapter 4.2.1 with the input data prepared in the described way. All three of the distance functions are applied to the result.

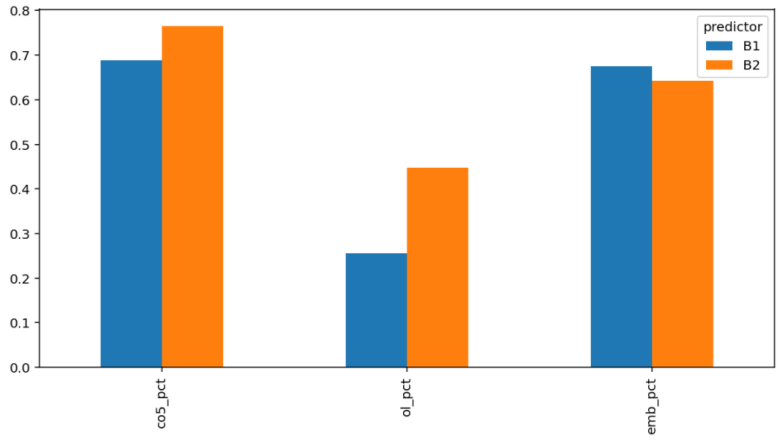

Figure 25: Baseline performance per model, comparing co5 (circle of fifths distance), ol (overlapping notes) and emb (cosine similarity of chord embeddings), all metrics normalised to scale of 0 to 1

As shown in Figure 25, architecture B.2 outperforms B.1 when looking at the circle of fifths distance only, whereas, when comparing the chord embeddings, B.1's score is higher. We hypothesize that this might be due to the LSTM leveraging the same chord embeddings to predict the next chord. The metric counting the number of overlapping chords shows an especially large difference between the two approaches.

4.2.4 Ceiling Analysis

To guide future improvements, we aim to find out which part of the Architecture has the biggest contribution to the overall error. For this purpose, we conduct a ceiling analysis looking at a selection of steps from each architecture.

B.1

We divide Architecture B.1 into three different segments.

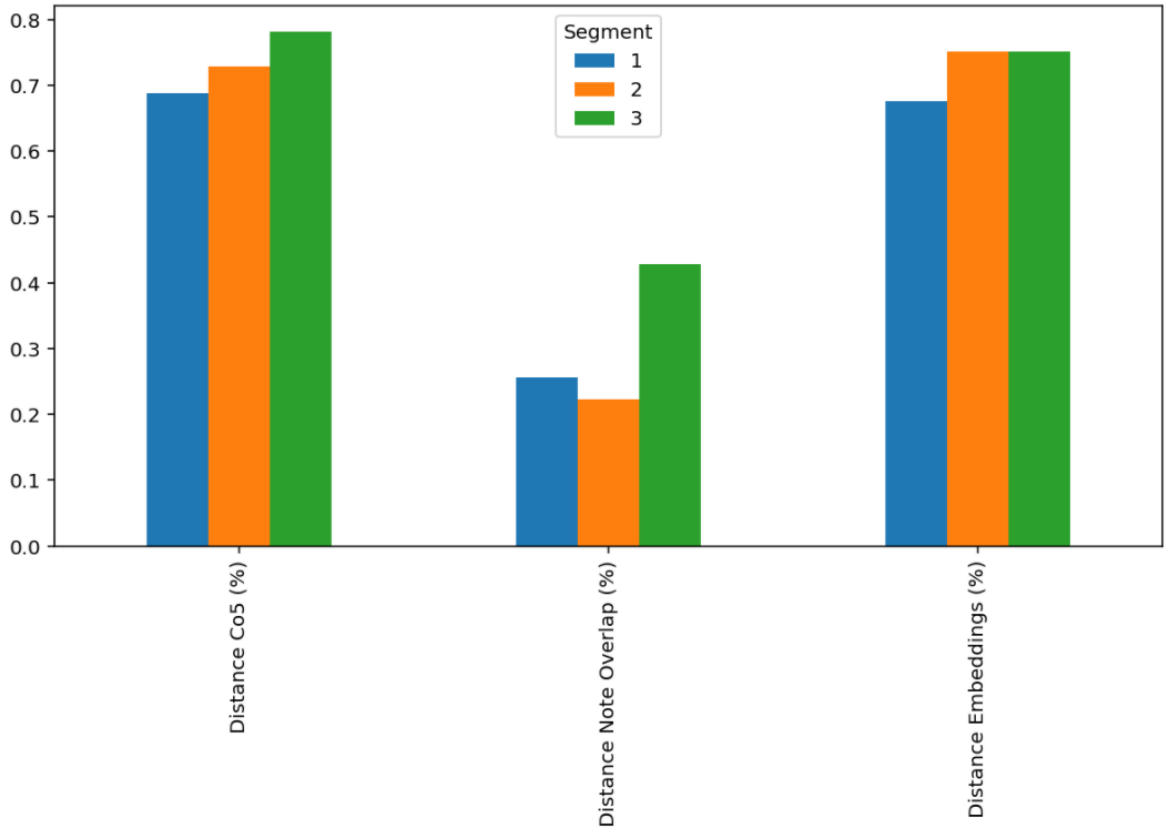

Figure 26: Results of the ceiling analysis of architecture B.1

Segment 1: The first segment is our baseline we established in the previous chapter. The melody will run through the normalization where it gets transposed, then it will be harmonized and, on the chord sequence produced, a prediction will be made.

Segment 2: The second segment removes the normalization step and starts directly with the harmonization. Therefore, we need to normalise the melody up front to the detected key of the complete song snippet.

Any improvement of Segment 2 over the baseline is a result of the errors in the key detection. As the architecture itself only has access to 4 bars of the melody, it is not always able to correctly detect the key and since later steps work with key – agnostic scale degrees instead of normal chords, errors will be produced. For the Co5 distance, this translates into an improvement from 68.8% to 72.8%. When looking at the chord embeddings, the difference is even greater, from 67.6% to 75.0%. When looking at note overlap, the result is surprisingly a decrease from 25.5% to 22.3, further indicating that note overlap might be unreliable as a metric.

Segment 3: Instead of harmonizing a melody, Segment 3 directly takes the real chord sequence as an input. The chord sequence, just like the melody before, has to be transposed to correct key upfront for the architecture to make a meaningful prediction. The increase in performance measured by the Co5 distance is 72.8% to 78.1%, indicating a potential performance improvement of 5.3% if a better harmonization tool were used. The difference measured by the embedding distance function is surprising in that there is almost none. Only 0.14% can be improved by using the real chords instead of the harmonized chord sequence. For the note overlap distance, performance almost doubled from 22.3% to 42.8%.

B.2

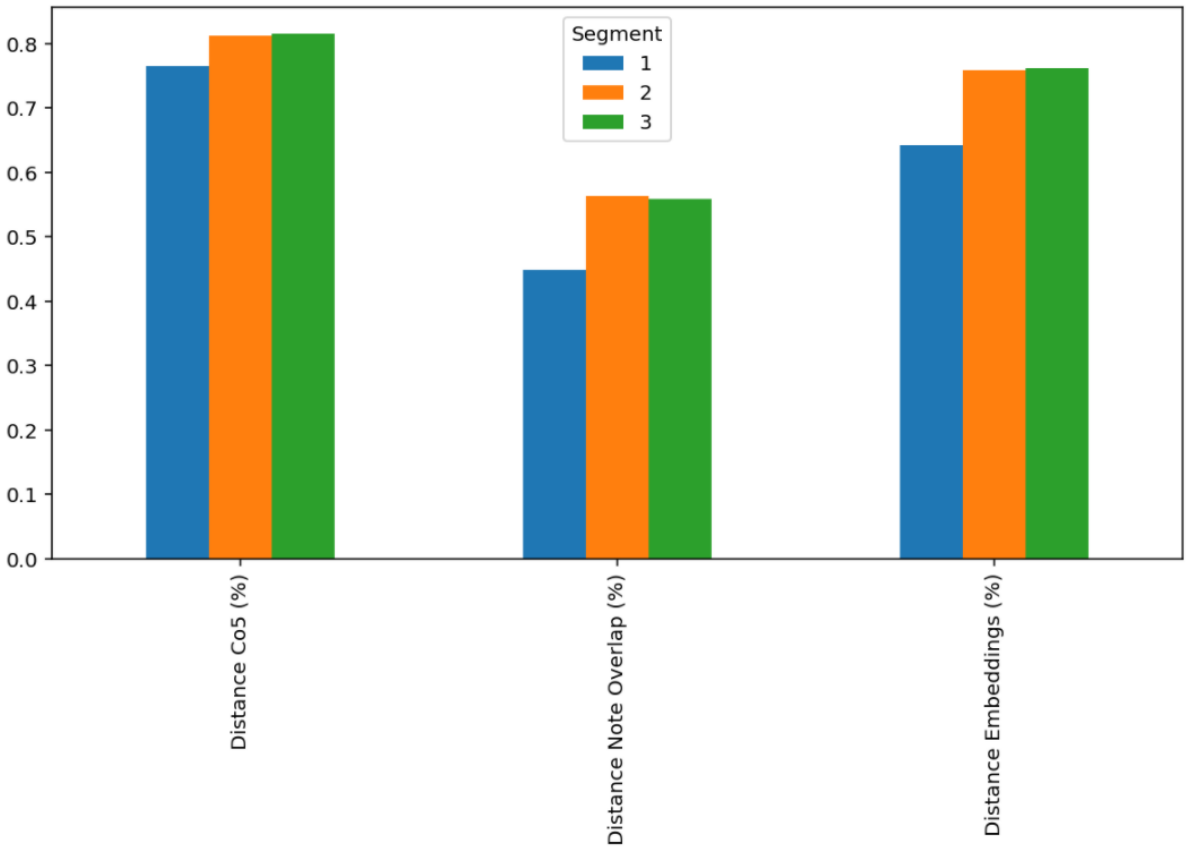

Figure 27: Results of the ceiling analysis for architecture B.2

Architecture B.2, just as B.1, was also split up into three different segments:

Segment 1 and Segment 2 work the same way as with Architecture B.1. The baseline is a unnormalized midi melody, while the second segment uses key detection over the full song snippet. For Segment 2, there is a clear improvement over the baseline across all metrics (4.7%, 11.5% and 11.7%). This is to be expected, as all subsequent steps can run on pretrained models using a single key and therefore rely on the input to be normalised correctly.

Segment 3 works by instead of having the musicautobot library complete the fifth bar of the melody, we pass the correct 5 bars of the melody to harmonize. Whichever chord is harmonized for the melody in the fifth bar is interpreted as the next chord. Whether the real 5 bars of melody or an autocompleted version of the melody are harmonized seems to play a minor role with less than 0.5% change in either metric.

Approach	Metric	Segment 1	Segment 2	Δ to previous	Segment 3	Δ to previous
B.1	Circle of Fifths Distance	68.79%	72.80%	4.01%	78.15%	5.35%
B.1	Overlapping Notes	25.54%	22.33%	-3.20%	42.82%	20.48%
B.1	Embedding Distance	67.58%	75.04%	7.46%	75.18%	0.14%
B.2	Circle of Fifths Distance	76.47%	81.20%	4.73%	81.60%	0.40%
B.2	Overlapping Notes	44.80%	56.30%	11.50%	55.88%	-0.42%
B.2	Embedding Distance	64.21%	75.95%	11.73%	76.20%	0.25%

Table 2: Full results of ceiling analysis across both architectures

The ceiling analysis has shown that for both architectures, the quality of the key detection is a large contributor to the overall performance of the chord prediction.

One strategy to use the results observed to increase real world performance of our application would be to have a longer listening window for key detection than what would be necessary for the actual chord prediction. That way, key detection can be done on a longer stream of information while keeping the stream that the actual prediction is based on short to increase performance.

Another workaround could be to have the user input the key of the piece manually.

4.3 Real-time Performance Evaluation

As our goal is a real time prediction, the process time of each component is an important factor for it to be useful. Therefore, we measured how long each step of our pipeline takes to be calculated.

For architecture B.1 the average time needed to harmonize a melody was around 4.3 seconds. Transposing a melody to another key and predicting the next chord were the fastest calculations, needing on average around 0.04 seconds. Transcribing the audio to symbolic representation, and predicting the next chord took around 0.09 seconds on average. The sum of the time spent of each component of B.1 was almost 4.6 seconds for one prediction. Architecture B.2 uses the same transcription, harmonization, and transpose modules, but it needs to extend the melody, which took on average 3.2 seconds. In total B.2 needs around 7.6 seconds for one prediction. We would like to mention that these time calculations are influenced by the machine that runs the algorithms. The time performance was calculated from a normal computer using a graphics process unit (GPU) for the harmonization and melody extension calculations. Therefore, we would like to remind the reader, that this evaluation might change depending on the machine used for testing.

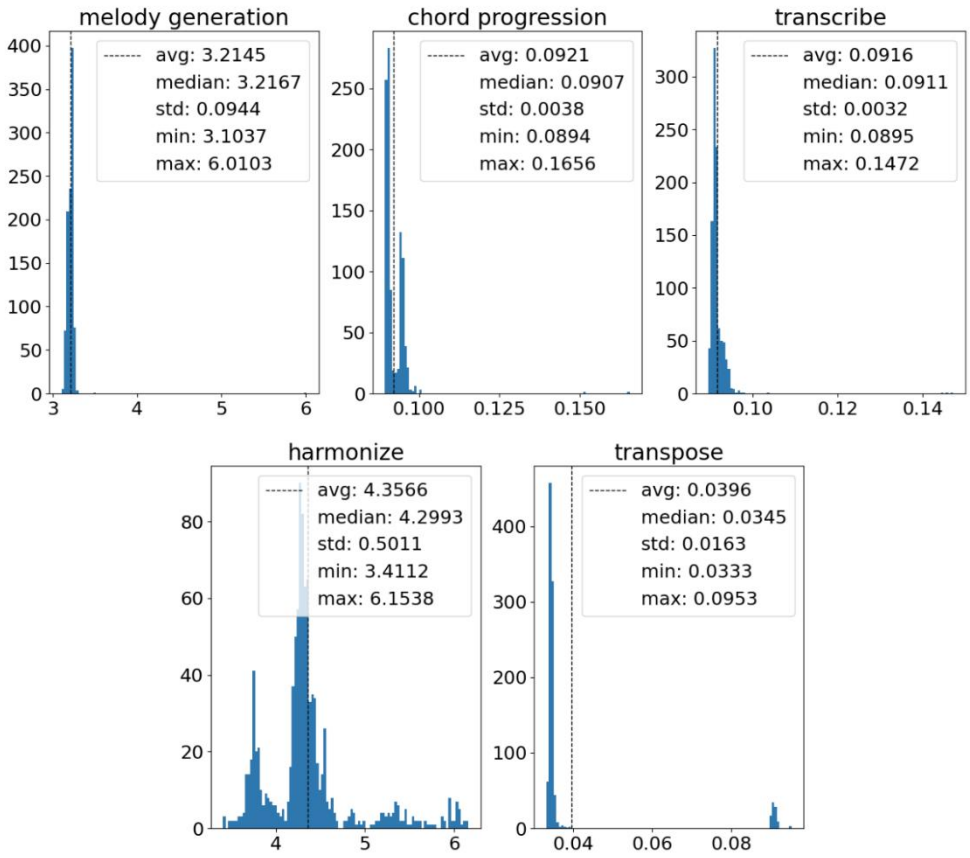

Figure 28: Performance of individual tasks in seconds

Discussion Real-time Performance evaluation

According to our results, the performance of some components is not suitable to use in a real-time context. Especially melody extension and harmonization are the most time consuming, both are from musicautobot.

The prediction will always have a time deficit, that the predicted chord should already have been played by the time our program shows it to the musician. If we consider a pop song, that typically has a tempo of 100 beats per minute (BPM) [64] and a 4/4 time signature, the length of a beat is 0.6 seconds. Therefore, our prediction would be at least six beats behind the song. Even if we consider that each chord has a length of four beats, we would still be one chord prediction behind of the correct chord.

4.4 Summary of results

In summary, our evaluations showed us that people are more interested in using our model to help them to create music instead of real-time predictions during the learning process. Moreover, our model validated that real-time prediction demands a fair number of computational resources and that each second is an essential aspect of a real-time prediction. As stated in chapter 3, the used tools and libraries could be replaced with better performing implementations.

5 Conclusion and Areas of improvement

5.1 Conclusion

To summarize, we have described and listed the state-of-the-art of MIR tasks relevant for this thesis in chapter 2. This could be used as a basis for creating models with state-of-the-art components. To solve the task of a Universal Songbook, we presented two approaches for a real-time chord prediction system that listens to audio input and outputs the predicted chords. In addition, we also implemented both approaches by using publicly available libraries and evaluated them with experiments, to measure their performance. A qualitative study with musicians suggests that our model has potential as a supporting tool during the music creation process. The results of the quantitative analysis overall were inconclusive in regard to whether the architectures are usable for real use cases, as we did not have a pre-existing benchmark to compare our scores to.

In conclusion, both models provide unsatisfactory results. Therefore, we cannot consider the task of a Universal Songbook to be solved.

5.2 Limitations

In its current state, our model is incapable of recognizing music structures (e.g., verse, refrain etc.). This could increase the overall chord prediction performance, as some of these structures have a fixed number of chords that repeat themselves for a certain amount of time. This might be suitable specially for popular western music, as these songs follow almost strictly some of these structures. Consequently, the real-time prediction is not capable of recognizing patterns in the played melody to acknowledge repeated sections that would result in equal predictions.

5.3 Areas of Improvement

Various improvements and changes can be made to our implementation. In this section we want to present and discuss a selection of them, based on our experiences during the project work.

Improve monophonic music transcription
As we see in chapter 4.1, monophonic transcription did not reliably work with the chosen music21 library. Another approach could be to use a pitch tracker, for example CREPE [11]. This could yield better transcription, and subsequently better predictions.
In addition, a better way of handling real-time microphone data should be used. Our approach was simplistic and does not produce desirable results, because directly transcribing snippets of audio data is not reliable, as notes could be played over multiple snippets for example.

Improve key detection
As discussed in chapter 4.2.4, wrong key detection is a large contributor to the overall performance. We could improve the performance by changing the way the key is detected, either by running key detection over a larger window than the actual prediction or by using better strategies to detect keys.

Use polyphonic music transcription
Even though we had to abandon our idea of using polyphonic transcription for our model, this approach can be successful if the polyphonic transcription would work better. We believe that using the real (and correctly transcribed) harmony information would lead to better predictions.

Use a self-trained multitask model

Using available libraries and pretrained models accelerated our development. However, we cannot provide a training analysis for musicautobot (melody harmonization and melody generation), as no information could be found for the pretrained model.

Training a multitask model would create more insights about accuracy, loss etc. With that information, improvements to the model can be made. Additionally, a self-trained model could potentially have a better performance than musicautobot.

Predict chord from melody directly

Instead of using multiple NN for prediction, this approach would use one NN with melody information as input and then directly outputs the predicted chord. This is related to the task of melody harmonization, but instead of predicting the chords for the current melody, it predicts the following chord to be played. For training, a dataset with split melody and chords is needed, an example could be used from Yeh et al. [7]

6 Annex

6.1 Bibliography

[1] J. S. Bamberger und A. Hernandez, Developing musical intuitions: A project-based introduction to making and understanding music, Oxford University Press, 2000.

[2] M. DeBellis, «What is musical intuition? Tonal theory as cognitive science,» *Philosophical Psychology, 12:4,* pp. 471-501, doi: 10.1080/095150899105693, 1999.

[3] T. Carsault, J. Nika, P. Esling und G. Assayag, «Combining Real-Time Extraction and Prediction of Musical Chord Progressions for Creative Applications,» *Electronics 10, no. 21: 2634, doi: 10.3390/electronics10212634,* 2021.

[4] J. P. Clendinning und E. W. Marvin, The Musician's Guide to Theory and Analysis: Third Edition, W W NORTON & CO, 2016.

[5] R. Typke, F. Wiering und R. C. Veltkamp, «A survey of music information retrieval systems,» in *Proc. 6th international conference on music information retrieval*, 2005.

[6] F. Argenti, P. Nesi und G. Pantaleo, «Automatic Music Transcription: From Monophonic to Polyphonic,» in *Musical Robots and Interactive Multimodal Systems*, 2011, pp. 27-46, doi: 10.1007/978-3-642-22291-7_3.

[7] Y.-C. Yeh, W.-Y. Hsiao, S. Fukayama, T. Kitahara, B. Genchel, H. M. Liu, H.-W. Dong, Y. Chen, T. Leong und Y.-H. Yang, «Automatic melody harmonization with triad chords: A comparative study,» *Journal of New Music Research. 50:1,* pp. 37-51, doi: 10.1080/09298215.2021.1873392, 2021.

[8] C.-W. Weng, C.-Y. Lin und J.-S. R. Jang, «Music instrument identification using mfcc: Erhu as an example,» in *Proc. 9th Int. Conf. of the Asia Pacific Society for Ethnomusicology*, Phnom Penh, Cambodia, 2004.

[9] D. Talkin und B. Kleijn, «A robust algorithm for pitch tracking (RAPT),» *Speech coding and synthesis,* 1995.

[10] J. Böhler und U. Zölzer, «Monophonic pitch detection by evaluation of individually parameterized phase locked loops,» in *19th International Conference on Digital Audio Effects (DAFX16)*, 2016.

[11] J. W. Kim, J. Salamon, P. Li und J. B. Pablo, «Crepe: A Convolutional Representation for Pitch Estimation,» *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* pp. 161-165, doi: 10.1109/ICASSP.2018.8461329, 2018.

[12] Y. LeCun und Y. Bengio, «Convolutional Networks for Images, Speech, and Time-Series,» 1997. [Online]. Available: http://yann.lecun.com/exdb/publis/pdf/lecun-bengio-95a.pdf.

[13] M. Mauch und S. Dixon, «PYIN: A fundamental frequency estimator using probabilistic threshold distributions,» *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* pp. 659-663, doi: 10.1109/ICASSP.2014.6853678, 2014.

[14] A. Camacho und J. G. Harris, «A sawtooth waveform inspired pitch estimator for speech and music,» *The Journal of the Acoustical Society of America 124,* pp. 1638-1652, 2008.

[15] J. Salamon, E. Gómez, D. P. Ellis und G. Richard, «Melody Extraction from Polyphonic Music Signals: Approaches, applications, and challenges,» *IEEE Signal Processing Magazine, vol. 31, no. 2,* pp. 118-134, doi: 10.1109/MSP.2013.2271648, March 2014.

[16] G. E. Poliner, . D. P. W. Ellis, A. F. Ehmann, E. Gómez, S. Streich und B. Ong, «Melody Transcription From Music Audio: Approaches and Evaluation,» *IEEE Transactions on Audio, Speech, and Language Processing, vol. 15, no. 4,* pp. 1247-1256, doi: 10.1109/TASL.2006.889797, May 2007.

[17] E. Benetos, S. Dixon, Z. Duan und S. Ewert, «Automatic Music Transcription: An Overview,» *IEEE Signal Processing Magazine,* Nr. 36, pp. 20-30, doi: 10.1109/MSP.2018.2869928, 2019.

[18] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff und A. Klapuri, «Automatic music transcription: Challenges and future directions,» *Journal of Intelligent Information Systems,* Nr. 41, doi: 10.1007/s10844-013-0258-3, 2013.

[19] R. Kelz, S. Böck und G. Widmer, «Deep Polyphonic ADSR Piano Note Transcription,» *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* pp. 246-250, doi: 10.1109/ICASSP.2019.8683582, 2019.

[20] Q. Kong, B. Li, X. Song, Y. Wan und Y. Wang, «High-resolution Piano Transcription with Pedals by Regressing Onset and Offset Times,» *IEEE/ACM Transactions on Audio, Speech, and Language Processing, doi: 10.1109/TASLP.2021.3121991,* 2021.

[21] S. Sigtia, E. Benetos und S. Dixon, «An End-to-End Neural Network for Polyphonic Piano Music Transcription,» *IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 24, no. 5,* pp. 927-939, doi: 10.1109/TASLP.2016.2533858, May 2016.

[22] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore und D. Eck, «Onsets and Frames: Dual-Objective Piano Transcription,» 2017. [Online]. Available: https://arxiv.org/abs/1710.11153.

[23] X. Fiss und A. Kwasinski, «Automatic real-time electric guitar audio transcription,» *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* pp. 373-376, doi: 10.1109/ICASSP.2011.5946418, 2011.

[24] J. Gardner, I. Simon, E. Manilow, C. Hawthorne und J. Engel, «MT3: Multi-Task Multitrack Music Transcription,» November 2021. [Online]. Available: https://arxiv.org/abs/2111.03017.

[25] K. W. Cheuk, D. Herremans und L. Su, «ReconVAT: A Semi-Supervised Automatic Music Transcription Framework for Low-Resource Real-World Data,» *Proceedings of the 29th ACM International Conference on Multimedia, doi: 10.1145/3474085.3475405,* 2021.

[26] L. Lin, Q. Kong, J. Jiang und G. Xia, «A Unified Model for Zero-shot Music Source Separation, Transcription and Synthesis,» *Proceedings of 21st International Conference on Music Information Retrieval, (ISMIR),* 2021.

[27] Y.-T. Wu, B. Chen und L. Su, «Multi-Instrument Automatic Music Transcription With Self-Attention-Based Instance Segmentation,» *IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 28,* pp. 2796-2809, doi: 10.1109/TASLP.2020.3030482, 2020.

[28] K. W. Cheuk, Y.-J. Luo, E. Benetos und D. Herremans, «Revisiting the Onsets and Frames Model with Additive Attention,» *2021 International Joint Conference on Neural Networks (IJCNN),* pp. 1-8, doi: 10.1109/IJCNN52387.2021.9533407, 2021.

[29] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li und P. J. Liu, «Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, doi,» 2020. [Online]. Available: https://arxiv.org/abs/1910.10683.

[30] D. Lee und H. Seung, «Learning the Parts of Objects by Non-Negative Matrix Factorization,» *Nature. 401,* pp. 788-91, doi: 10.1038/44565, 1999.

[31] P. Smaragdis und J. C. Brown, «Non-negative matrix factorization for polyphonic music transcription,» *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (IEEE Cat. No.03TH8684),* pp. 177-180, doi: 10.1109/ASPAA.2003.1285860, 2003.

[32] R. Ardila, M. Branson, K. Davis, M. Henretty, M. Kohler, J. Meyer, R. Morais, L. Saunders, F. M. Tyers und G. Weber, «Common Voice: A Massively-Multilingual Speech Corpus,» 2020. [Online]. Available: https://arxiv.org/abs/1912.06670.

[33] R. Hennequin, A. Khlif, F. Voituret und M. Moussallam, «Spleeter: a fast and efficient music source separation tool with pre-trained models,» *Journal of Open Source Software, 5,* pp. 2154, doi: 10.21105/joss.02154, 2020.

[34] E. Manilow, P. Seetharman und J. Salamon, «Open Source Tools & Data for Music Source Separation,» 2020. [Online]. Available: https://source-separation.github.io/tutorial.

[35] N. Takahashi und Y. Mitsufuji, «D3Net: Densely connected multidilated DenseNet for music source separation,» 2020. [Online]. Available: https://arxiv.org/abs/2010.01733.

[36] A. Défossez, N. Usunier, L. Bottou und F. Bach, «Music Source Separation in the Waveform Domain,» 2021. [Online]. Available: https://arxiv.org/abs/1911.13254.

[37] A. Défossez, «Hybrid Spectrogram and Waveform Source Separation,» November 2021. [Online]. Available: https://arxiv.org/abs/2111.03600.

[38] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis und R. Bittner, «MUSDB18 - a corpus for music separation,» 2017. [Online]. Available: https://doi.org/10.5281/zenodo.1117372.

[39] E. Vincent, R. Gribonval und C. Févotte, «Performance measurement in blind audio source,» *IEEE Transactions on Audio, Speech and Language Processing, Institute of Electrical and Electronics Engineers, 14 (4),* p. 1462–1469, 2006.

[40] C.-E. Sun, Y.-W. Chen, H.-S. Lee, Y.-H. Chen und H.-m. Wang, «Melody Harmonization Using Orderless NADE, Chord Balancing, and Blocked Gibbs Sampling,» 1 February 2021. [Online]. Available: https://arxiv.org/abs/2010.13468.

[41] S. Hochreiter und J. Schmidhuber, «Long short-term memory,» *Neural computation, vol. 9, no. 8,* p. 1735–1780, 1997.

[42] R. Lawrence und B. Juang, «An introduction to hidden Markov models,» *ieee assp magazine,* pp. 4--16, 1986.

[43] J. H. Holland, «Genetic algorithms,» *Scientific american,* pp. 66--73, 1992.

[44] H. Larochelle und I. Murray, «Neural autoregressive distribution estimation,» *Journal of Machine Learning Research - Proceedings Track. 15,* pp. 29-37, 2011.

[45] F. Korzeniowski, D. R. Sears und G. Widmer, «A large-scale study of language models for chord prediction,» in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[46] S. Madjiheurem, L. Qu und C. Walder, «Chord2vec: Learning musical chord embeddings,» in *Proceedings of the constructive machine learning workshop at 30th conference on neural information processing systems (NIPS2016)*, Barcelona, Spain, 2016.

[47] E. Anzuoni, S. Ayhan, F. Dutto, A. McLeod, F. C. Moss und M. Rohrmeier, «A historical analysis of harmonic progressions using chord embeddings,» *Proceedings of the 18th Sound and Music Computing Conference,* pp. 284--291, 2021.

[48] P. Doornbusch, «Algorithmic Composition: Paradigms of Automated Music Generation,» *Computer Music Journal,* pp. 70--74, 2010.

[49] F. Carnovalini und A. Rodà, «Computational Creativity and Music Generation Systems: An Introduction to the State of the Art,» *Frontiers in Artificial Intelligence, vol. 3, doi: 10.3389/frai.2020.00014,* 2020.

[50] A. Shaw, «Creating a Pop Music Generator with the Transformer,» 13 August 2019. [Online]. Available: https://towardsdatascience.com/creating-a-pop-music-generator-with-the-transformer-5867511b382a.

[51] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le und R. Salakhutdinov, «Transformer-XL: Attentive Language Models beyond a Fixed-Length Context,» *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics,* pp. 2978–2988, doi: 10.18653/v1/P19-1285, July 2019.

[52] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville und Y. Bengio, «Generative Adversarial Networks,» 2014. [Online]. Available: https://arxiv.org/abs/1406.2661.

[53] A. Muhamed, L. Li, X. Shi, S. Yaddanapudi, W. Chi, D. Jackson, R. Suresh, Z. C. Lipton und A. J. Smola, «Symbolic Music Generation with Transformer-GANs,» *35th AAAI Conference on Artificial Intelligence, (AAAI) 2021,* 2021.

[54] D. Byrd und T. Crawford, «Problems of music information retrieval in the real world,» *Information processing & management,* pp. 249--272, 2002.

[55] D. Temperley, «What's Key for Key? The Krumhansl-Schmuckler Key-Finding Algorithm Reconsidered,» *Music Perception, vol. 17, issue 1,* pp. 65–100, doi: 10.2307/40285812, 1 October 1997.

[56] M. S. Cuthbert und C. Ariza, «music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data,» *11th International Society for Music Information Retrieval Conference (ISMIR 2010),* pp. 637-642, 2010.

[57] O. Ronneberger, P. Fischer und T. Brox, «U-Net: Convolutional Networks for Biomedical Image Segmentation,» *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. MICCAI 2015. Lecture Notes in Computer Science, vol 9351, doi: 10.1007/978-3-319-24574-4_28,* pp. 234-241, 2015.

[58] Buskirk, «The Most Popular Keys of All Music on Spotify,» 2017. [Online]. Available: https://web.archive.org/web/20171016100014/https://insights.spotify.com/us/2015/05/06/most-popular-keys-on-spotify/. [Zugriff am 22 12 2021].

[59] J.-F. Paiement, D. Eck und S. Bengio, «A probabilistic model for chord progressions,» *Proceedings of the Sixth International Conference on Music Information Retrieval (ISMIR),* 2005.

[60] W. B. De Haas, F. Wiering und R. C. Veltkamp, «A geometrical distance measure for determining the similarity of musical harmony,» *International Journal of Multimedia Information Retrieval, Vol 2,* pp. 189--202, 2013.

[61] E. C. A. Childs und C.-H. Chuan, Mathematics and Computation in Music, Springer, 2009.

[62] D. Jatnika, M. A. Bijaksana und A. A. Suryani, «Word2vec model analysis for semantic similarities in english words,» *Procedia Computer Science,* pp. 160--167, 2019.

[63] D. Herremans und C.-H. Chuan, «Modeling musical context with word2vec,» *arXiv preprint arXiv:1706.09088,* 2017.

[64] M. Biss, «Rhythm Tips for Identifying Music Genres by Ear,» [Online]. Available: https://www.musical-u.com/learn/rhythm-tips-for-identifying-music-genres-by-ear/.

6.2 List of abbreviations

(bi)LSTM	(Bi-directional) Long Short Term Memory
AMT	Automatic Music Transcription
BPM	Beats Per Minute
CNN	Convolutional Neural Network
Co5	Circle of fifths
CREPE	Convolutional Representation for Pitch Estimation
dB	Decibels
GA	Genetic Algorithm
GAN	Generative Adversarial Networks
HMM	Hidden Markov Model
MIDI	Musical Instrument Digital Interface
MIR	Music Information Retrieval
MT3	Multi-Task Multitrack Music Transcription
NADE	Neural Autoregressive Distribution Estimation
NLP	Natural Language Processing
NMF	Non-Negative Matrix factorization
NN	Neural Network
PYIN	Probabilistic YIN
SDR	Source-to-Distortion Ratio or Signal-to-Distortion Ratio
SWIPE	Sawtooth Waveform Inspired Pitch Estimator
T5 Transformer	Text-To-Text Transfer Transformer
ZHAW	Zürcher Hochschule für Angewandte Wissenschaften

6.3 List of figures

6.4 List of tables

6.5 Project assignment

Zürcher Hochschule
für Angewandte Wissenschaften

School of Engineering

Learning to Predict Chord Progression in Contemporary Music
PA21_stdm_01

BetreuerInnen:	Thilo Stadelmann, stdm Matthias Rosenthal, rosn
Fachgebiete:	Artificial Intelligence (AI) Bildverarbeitung (BV) Datenanalyse (DA) Digitale Signalverarbeitung (DSV) Machine Learning (ML) Software (SOW)
Studiengang:	ET / IT / ST
Zuordnung:	Centre for Artificial Intelligence (CAI)
Interne Partner:	Institute of Embedded Systems (InES)
Gruppengrösse:	3

Kurzbeschreibung:
Guitarist, Keyboarders and other musicians playing accompanyment for a band (or to a recording) know the difficulty and fun of playing along to the music without sheet music or a chord sheet. While it is hard for a beginner to predict the next chord, with some experience people usually get very good in "guessing" where the melody is probably going and what chord to play next on the instrument (yes, knowledge of the key of the song and of the circle of fifths helps as well).

In this thesis, we want to mimic this human learning process with machine learning to create the prototype of a software that listens to music and predicts in real-time the current and next chord. This could later be turned into an app that, as a "universal chord sheet", can substantially easy the learning curve for young musicians to just play along to any song they might have never heard before.

Work packages:

- Review of related work on chord recognition and prediction (scientific literature, public github repos)
- Define prototype scope and training data (e.g., first first midi-based, only then audio-based)
- Set up the development environment (locally and potentially on our GPU cluster)
- Build initial prototype, iterate (focus on algorithms, machine learning, functionality; not on UI)
- Write a scientific report with a focus on motivation, argumentation, methods, evaluation & results (the PA thesis)

Voraussetzungen:
This thesis can be conducted in German or English. Some knowledge of music theory and machine learning is a plus, but not required. Strong interest in AI and scientific research as well as scholarly work is mandatory, as well as the ability to independent conceptual work, good general programming skills (the project is probably conducted in Python and PyTorch, but both can be learned during the project) and a pragmatic approach to experimentation and thorough evaluation of results.

The Computer Vision, Perception and Cognition Group at the newly-founded ZHAW Centre for Artificial Intelligence conducts research and teaching in pattern recognition using deep learning methodology. In our thesis proposals, we want to address high-achieving students with a curiosity for gaining (first) experience with scientific research applied to the problem at hand. The supervisors are very enthusiastic about the topics and have plenty of experience as well as detailed ideas (and offering of support) for the project start-up. We are looking for equally enthusiastic and high-achieving students, with the hope (given good results) of a joint scientific publication.

Die Arbeit ist vereinbart mit:
Kevin Kläger (klaegkev)
Urban Lutz (lutzurb1)

Friday 21. May 2021 17:07

Guiherme Vicentini Briner (vicengui)

Weiterführende Informationen:
https://medium.com/@huanlui/chordsuggester-i-3a1261d4ea9e

6.6 Meeting protocols

September 20, 2021

- Define research question
- 2-3 weeks literature-research
- Possible formats
    - MIDI
    - Chord Sequence
    - Audio
- Scientific approach: Hypothesis -> Experiment, repeat
- What are possible metrics
- Create a timetable
- Organise meetings
    - Monday, 13:30
    - Send agenda before meeting

September 27, 2021

- Research for chord prediction / harmonization exists
    - PA agreement is a guideline, can be flexible
- Possibilities
    - Chord prediction (original idea)
    - Listen to music and generate chord
    - Generate music
- Variations
    - Application of current technology
        - Predictability for a chart-hit
        - Defining trends for prediction
    - Improving current approaches
        - What would we like to add here?
            - More complex models (e.g., not omitting chords)
            - Copyrighting, plagiarism detection
            - New technique and compare to existing literature (GAN)
    - Present possible directions next meeting
    - From input, list suitable chords that can be played (audio, midi, etc. possible)
    - New architecture for chord prediction
        - More complex models @vicengui
        - GAN @Kevin Kläger
        - NLP improvements @Urban Lutz
    - Request access for clusters etc. now @vicengui
    - Allow access to git repository for Thilo & Matthias
    - Send protocol as mail to Thilo @Urban Lutz

October 4, 2021

- Defined melody harmonization / chord prediction as our task (melody in, which chords should be played)
- Real-time functionality
- Universal Songbook: Accompany a song

- Replicate state-of-the-art for these tasks
    - Define criteria to assess replicated papers to see where improvement is possible/necessary.
- Priority on libraries / tools that are publicly available and easy to implement

October 18, 2021

- We will follow our implementation idea from power point
- Input audio -> symbolic representation (music transcription) is not a focus. We will use existing libraries
- If possible and enough time, train models with dataset (to be defined)
- Inputs from Thilo
    - Possible extension: multitask learning model that can solve multiple tasks (Chord Sequence Completion and Melody Harmonisation) instead of having 2 models.
    - Possible extension: Instead of solving everything with ML, one could also use Information Retrieval. Example: Has the melody already been harmonised? => If yes - take the chord, if no - let the model predict.
    - Both models produce an output. Here we still have to find a solution, which of the two is selected, if they are not of the same opinion (keyword 'confidence prediction').
- The goal is to develop a prototype that can be shown live. even if it doesn't work perfectly.
- We change the meeting to a 2-week rhythm. The next meeting will take place on 01.11.

November 1, 2021

- Quantitative evaluation: minimum expectation of steps, and then evaluate → where to evaluate
- Ensure comparability to state-of-the-art
- Start with documentation: create table of contents with bullet points
- Scientific papers are usually structured in a similar way
    - The most difficult thing is how do I summarise thoughts / results etc.?
    - How much do I write per chapter?

November 15, 2021

- Our status: polyphonic approach does not work due to missing dataset and bad polyphonic transcription.
    - Mail to authors of https://arxiv.org/pdf/2001.02360.pdf for their dataset
- Audio2Midi could be omitted → test with midi keyboard
- Thilo: Focus on something that works as a prototype.
    - Qualitative analysis: are chords right? Example with 5 test persons
    - Comparability to other studies is not relevant
- Future Work idea: Harmonizer could be skipped, own model → melody goes in, calculate next chord
- Write documentation with concrete text
    - Differentiation between "what already exists" and "what we have done" is important. It must be explicitly clear which part is ours
    - Motivation: what motivates me personally, but also: why is it a relevant problem. References that make clear why it is important (e.g., chord prediction is hard → reference that confirms this statement).

    - Method: How do we describe it?
        - Our approach is not clear. We had the 2 approaches. Why and why did we choose this method? Example title: selected approaches (first followed approach 1, didn't work because of xxxx, therefore approach 2)
    - Describe the setup in an understandable way (libraries, content and how it works do not need to be described, only why we used it).
    - Future work (choose better names according to Matthias: Possible next steps / Areas of improvement): Conclusion → we have achieved this and that.
    - Outlook: Next, we would go in this direction and bring in these improvements
    - No code in the documentation (except in the appendix if necessary)

November 29, 2021

- Demo: Full real-time is difficult because of the performance of our tools
    - Performance is too slow for real-time (musicautobot)
    - Describe: What are the steps for real-time, where are pros/cons, which parts should be exchanged
- Try real-time with minimal effort or try midi real-time
- Train models with own data → if we have time, otherwise add to Future Work
- Input for structure
    - Method
        - No logging of iterative process. This is only relevant if it really changed drastically by changing a few parameters
        - Describe what worked and describe only the final result
        - In the method part, describe Architecture A, but let it be thrown out (also give reasons directly)
    - Our Contribution can be read in 2-3 places (abstract, introduction, conclusion)
        - 2-4 sentences very precisely our own contribution to the state of the art
- Submit draft of thesis on 13.12.2021 (first 5 pages maybe)

December 13, 2021

- Input for Documentation
    - Chapter 3 naming: Proposed System Architectures
        - Architectures can also have individual names
    - Chapter 4
        - Structure: what are the results
        - Only interpret after showing results
    - Introduction
        - Own contribution at the end

December 20, 2021

- Github repo: make Thilo and Matthias to admins
- Defined submission of PA for 24.12.2021, 12:00
- Insert logo of CAI manually from intranet
- Feedback meeting of PA: 14.02.2022

6.7 Timetables

Planned Timetable

Week	Tasks
38	Research
39	Research
40	Research
41	Research
42	Methodology
43	Implementation
44	Implementation
45	Training
46	Training
47	Results
48	Documentation
49	Documentation
50	Documentation
51	Deadline

Effective Timetable

Week	Tasks
38	Research
39	Research
40	Research
41	Research
42	Research Methodology
43	Implementation
44	Implementation
45	Implementation
46	Implementation Methodology
47	Implementation Results
48	Implementation Documentation Results
49	Documentation
50	Documentation Results
51	Documentation Deadline

7 Further Information

Github repository: https://github.zhaw.ch/PAIT/chord-predictor