MIREX 2018:
MULTI OBJECTIVE CHORD ESTIMATION

Stefan Gasser and Franz Strasser
Machine Learning and Audio: A Challenge
Johannes Kepler University, Linz, Austria
k1455519@students.jku.at

ABSTRACT

This paper summarizes our chord estimation submission for the MIREX 2018 chord estimation challenge. It contains an overview over the attempted baselines, the used neural network architectures, a definition of the exact sub-objectives we attempt to solve, and all methods used while training like data preprocessing and data augmentation.

1. INTRODUCTION

The detection of a chord that sounds at a concrete time in a song is a very important step for transcribing music. Even musicians have a hard time doing this, for a machine this is a very hard task as we can see for example in the results of the last MIREX challenges. But nevertheless, automatic chord detection with high precision is an extremely important step towards automatic music transcription.

The best approaches for chord estimation mostly rely on machine learning. But even considerably large neural networks trained on large amounts of data admin room for improvement, as we have seen from previous years. In the following, we present a slightly different approach to get better results.

2. RELATED WORK

A good overview about the current state-of-the-art in chord estimation is given by the submissions of the four participating teams in the MIREX 2017 chord estimation challenge. Relatively good results, especially for that tasks where only the root note, major and minor chords as well as the bass note has to be estimated, were achieved by Korzeniowski et al. [2]. They presented two approaches, one was a fully convolutional neural network (CNN) which was introduced in [4], the other was based on a Deep Chroma Extractor introduced in [3]. For further information on the topic we encourage you to go to the currently most recent MIREX results page and have a look at the submissions [1].

[1] See `http://www.music-ir.org/mirex/wiki/2017:Audio_Chord_Estimation_Results`.

3. OBJECTIVES FOR CHORD ESTIMATION

In music, a chord is a harmonic set of pitches consisting of two or more (usually three or more) notes that sound simultaneously. To uniquely describe such a chord (without regard for enharmonic equality), three parts are needed as described in the following.

3.1 Root Note

The root note describes the tonic note of that scale where the chord notes originate in. It is the only property that gives the absolute position of a chord and therefore essential for completely describing any chord. The class distribution is displayed in Table 1

Note	**N**	**C**	**C#**	**D**	**D#**	**E**
[%]	4.79	10.66	4.03	11.61	5.32	10.49

F	**F#**	**G**	**G#**	**A**	**A#**	**B**
8.43	4.24	11.65	4.79	12.12	5.94	5.9

Table 1. The distribution of time spent on the root notes in our data sets. N stands for None.

3.2 Chord Description

The chord description defines the spaces in semitones between the notes, which are responsible for the chords sound, and also define the number of different tones the chord has. There are for example the two mainly used chord types in western music, major and minor chords, but there are also more rarely used ones, like a dominant sevenths chord that consists of four notes. By analyzing our training data we have seen, that 68% of the time where a chord is sounding ("no chord" times were omitted), it is a major chord. The class distribution is displayed in Table 2

3.3 Inversion

The notes of a chord are not required to have the same order with regard to their frequency. This results in different, so called "inversions", which can be described by a number that is the index of the note with the lowest frequency, also called bass note. Our training data chords are approximately 94% of the time without any inversion. The class distribution is displayed in Table 3

Chord Description	maj	min	dim	aug
[%]	67.49	13.34	0.29	0.15
	doms	**majs**	**mins**	
	8.38	2.89	7.44	

Table 2. The percent of time spent with certain chord descriptions in our data sets. The full name of the classes is (in order): major, minor, diminished, augmented, dominant seventh, major seventh, minor seventh

Inversion	N	First	Second	Third
[%]	93.98	2.22	2.59	1.21

Table 3. The percent of time spent on certain inversions in our data sets. N stands for no inversion.

3.4 Notation

To uniquely notate (without regard for enharmonic equalities) such chords, there exist multiple different approaches. MIREX decided to use the notation introduced by Harte et al. in [1], which is popular in western countries. This notation is built on the previously described properties. We use it to structure the chord transcription task into multiple subtasks, as described in the following section.

3.5 Stages of the MIREX chord estimation challenge

The MIREX chord estimation challenge is divided into several stages with increasing complexity:

1. **root note only:** The most basic task is to only estimate the root note of a chord. If the root note is wrong, everything else in the chord estimation will be wrong too.
2. **major minor:** The root note including a distinction between major and minor chords is taken into account. All other chord descriptions are ignored.
3. **major minor bass:** All previous things including the bass note (or inversion) are considered. This means this stage is the first where full chord definitions are necessary.
4. **sevenths:** In this stage, also sevenths chords are taken into account, without the inversions.
5. **sevenths bass:** The final and most complex stage includes the root note, major, minor, and sevenths chords including their bass note (inversion).

4. ROBUST BASELINE APPROACHES FOR ROOT NOTE ESTIMATION

The first task tackled was the root note estimation. It is the essential part of the whole chord estimation process, because chords with wrong root note are counted as wrong in every stage.

For data preprocessing, we re-sampled the audio data to 44100Hz and calculated the STFT from it with a window-size of 2048 and a hop-size of 1024. This spectrogram is then logarithmically divided into 96 chroma bins. Then, always two bins are combined together, to get 48 bins. This summing up of the bins is done in a way, so that the result are not usual 48 chroma bins, but shifted by half of a bin. This procedure can be seen in Figure 1.

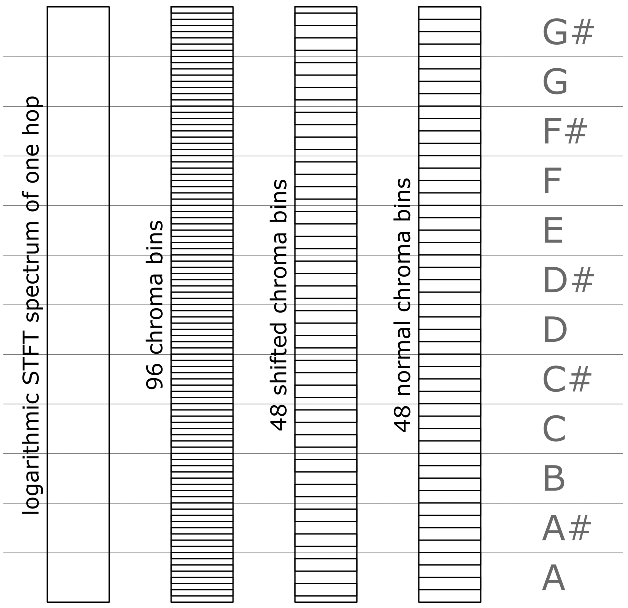

Figure 1. The graphical explanation of shifted chroma bins. The mid between two semitones is the center of a shifted chroma bin, while usual chroma bins have an edge there.

To get a robust baseline for the root note estimation, we started by fitting a random forest, that labels short snippets of audio with one of the twelve possible semitones or the “no chord” label. For training this small random forest (consisting of 100 decision trees), we used 10% of the songs in our training set, chosen uniformly at random.

We also tried a different approach, by creating 13 different random forest regressors (one for each semitone and one for “no chord”) that estimate the probability, that a short snippet of audio has one of the semitones as root note. The final output is the representing label of the forest that gives the highest probability. These forests, with 30 decision trees each, were trained on 5% of randomly chosen songs from our training set.

The results of both approaches can be found in Table 9. As can be seen in the results, these two baselines were very low. This could have a wide range of different reasons, but the two most probable are, we only used a very small part of the training set and no data augmentation which leads to a very poor generalization. The second reason is the lack of complexity of the approaches, the complexity of the problem is much higher than the complexity of the model class. Therefore, we were forced to use higher level machine learning techniques as described in the following.

5. CNN CHORD RECOGNITION

This section covers our adaptations to the fully convolutional neural network introduced in [4]. We decided on using the CNN approach instead of the DeepChormaEstimator because its deeper design allows for more complex representations and it is therefore more likely to adapt to the changes we introduce.

5.1 Preprocessing

The first step of the feature processing pipeline transforms the audio into a time-frequency representation that serves as the input for the CNN. All audio data is re-sampled to 44100Hz. We use an STFT with a hop-size of 4410 and a window-size of 8192, so that we get exactly ten frames per second. This STFT will then be restricted to a frequency range of 60Hz - 2600Hz and converted into a logarithmically scaled magnitude spectrogram L with 113 bins, using the following equation:

$$L = log(1 + STFT \times B_{\Delta})$$

The matrix B_{Δ} is our filter bank which maps the previously specified frequency range onto the 113, exponentially spaced, bins. This filter bank can theoretically be constructed using any number as the base of the exponent, but to make the process of data augmentation more simple we will use 2 as the base. The input for the CNN was then composed using a small fragment of this time-frequency signal. The input matrix X_i represents the audio file at time $i \times \frac{hop\_size}{sample\_rate}$ and is composed using:

$$X_i = [L_{i-C}, \dots, L_i, \dots, L_{i+C}]$$

For our purposes we chose $C = 11$. If the indices were out of the valid range we used zero padding.

5.2 CNN architecture

We used the same basic approach as Korzeniowski et al. in their submission from 2017 [2]. The main disadvantage of this approach is that it is restricted to the subset of the 25 most likely chords. Therefore, we decided to introduce a split into the three objectives that are described in Section 3. We think that the split into these objectives makes sense from a music theoretical point of view, and we hope that it will lead to better generalization, as there is no restriction of possible classes necessary. The general architecture of our Neural Network is described in Figure 2.

5.3 Data augmentation

Korzeniowski et al. state in [4] that this network tends to overfit, therefore good data augmentation methods are essential. Korzeniowski et. al. used two data augmentation methods, although they didn't specify how they did it exactly, we tried our best to recreate them based on their description.

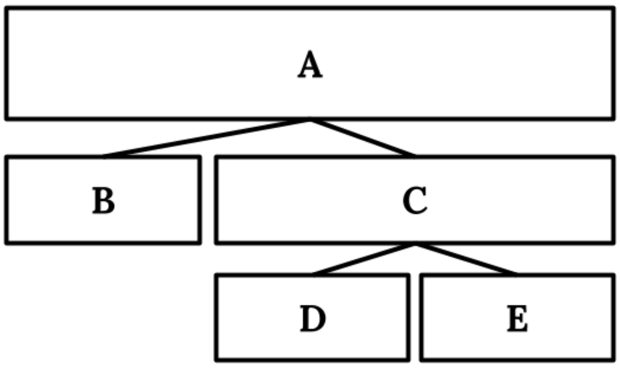

Figure 2. The general overview over out proposed CNN architecture. It is separated in 5 parts (A-E) for more details regarding the implementation look at the corresponding layouts in Table 4-Table 8.

Layer Type	Parameters	Output Size
Input		113×23
Conv-ReLU	$32 \times 3 \times 3$	$32 \times 111 \times 21$
Conv-ReLU	$32 \times 3 \times 3$	$32 \times 109 \times 19$
Conv-ReLU	$32 \times 3 \times 3$	$32 \times 107 \times 17$
Conv-ReLU	$32 \times 3 \times 3$	$32 \times 105 \times 15$
Pool-Max	2×1	$32 \times 52 \times 15$
Conv-ReLU	$64 \times 3 \times 3$	$64 \times 50 \times 13$
Conv-ReLU	$64 \times 3 \times 3$	$64 \times 48 \times 11$
Pool-Max	2×1	$64 \times 24 \times 11$
Conv-ReLU	$128 \times 12 \times 9$	$128 \times 13 \times 3$

Table 4. Part A of our proposed CNN architecture for multi objective chord estimation, including activations and kernel sizes. There was no padding used and the stride was 1 for each layer. Batch normalization is performed after each convolution layer. Dropout with probability 0.5 is applied after horizontal rules in the table. This is the common feature extraction part of our network and is shared between all objectives

Layer Type	Parameters	Output Size
Conv-ReLU	$128 \times 3 \times 3$	$128 \times 13 \times 3$
Conv-ReLU	$64 \times 3 \times 3$	$64 \times 13 \times 3$
Conv-Linear	$n_{root\_note} \times 1 \times 1$	$n_{root\_note} \times 13 \times 3$
Pool-Avg	13×3	$n_{root\_note} \times 1 \times 1$
Softmax		n_{root_note}

Table 5. Part B of our proposed CNN architecture for multi objective chord estimation, including activations and kernel sizes. There was padding used and the stride was 1 for each layer. Batch normalization is performed after each convolution layer. This is the part of the network that focuses on getting the root note information out of the data. n_{root_note} denotes the amount of different root note classes which is 13 (12 semitones + no tone class).

Layer Type	**Parameters**	**Output Size**
Conv-ReLU	$128 \times 3 \times 3$	$128 \times 13 \times 3$
Conv-ReLU	$128 \times 3 \times 3$	$128 \times 13 \times 3$

Table 6. Part C of our proposed CNN architecture for multi objective chord estimation, including activations and kernel sizes. There was padding used and the stride was 1 for each layer. Batch normalization is performed after each convolution layer. This part is the common feature adaption for the tasks chord description and inversion.

Layer Type	**Parameters**	**Output Size**
Conv-ReLU	$128 \times 3 \times 3$	$128 \times 13 \times 3$
Conv-ReLU	$64 \times 3 \times 3$	$64 \times 13 \times 3$
Conv-Linear	$n_{chord\_desc} \times 1 \times 1$	$n_{chord\_desc} \times 13 \times 3$
Pool-Avg	13×3	$n_{chord\_desc} \times 1 \times 1$
Softmax		n_{chord_desc}

Table 7. Part D of our proposed CNN architecture for multi objective chord estimation, including activations and kernel sizes. There was padding used and the stride was 1 for each layer. Batch normalization is performed after each convolution layer. This is the part of the network that focuses on getting the chord description out of the data. n_{chord_desc} denotes the amount of different chord description classes which is 7.

Layer Type	**Parameters**	**Output Size**
Conv-ReLU	$128 \times 3 \times 3$	$128 \times 13 \times 3$
Conv-ReLU	$64 \times 3 \times 3$	$64 \times 13 \times 3$
Conv-Linear	$n_{inversion} \times 1 \times 1$	$n_{inversion} \times 13 \times 3$
Pool-Avg	13×3	$n_{inversion} \times 1 \times 1$
Softmax		$n_{inversion}$

Table 8. Part E of our proposed CNN architecture for multi objective chord estimation, including activations and kernel sizes. There was padding used and the stride was 1 for each layer. Batch normalization is performed after each convolution layer. This is the part of the network that focuses on getting the inversion information out of the data. $n_{inversion}$ denotes the amount of different chord description classes which is 4 (3 different kinds of inversion + no inversion).

5.3.1 Note Basics

First, lets cover how the frequency and the tone of a note correlate, to better understand how these data augmentation methods work. If you know the octave and the tone you want to replicate, you can calculate the frequency of the new tone with the following formula:

$$f_{note} = f_{base} \times 2\hat{}(\delta_{octave} + \frac{\delta_{semitone}}{12})$$

where f_{base} is the frequency of the base tone (mainly 440Hz for base tone a' is used), δ_{octave} is the difference in octaves between the base note and the new note, and $\delta_{semitone}$ is the difference in semitones. Please note that one octave difference also equals a difference of 12 semitones, but it is uncommon to think of it as such. This means that notes are exponentially spaced in frequency space and brings us back to our filter bank. Since we used exponentially spaced bins with the same base as the tones, the distance between two (semi-)tones is linear in our feature space.

$$\begin{aligned}
\Delta &= ld(f_1) - ld(f_2) \\
&= ld(f_{base} \times 2\hat{}(o1 + \frac{s1}{12})) - ld(f_{base} \times 2\hat{}(o2 + \frac{s2}{12})) \\
&= ld(f_{base}) + o1 + \frac{s1}{12} - ld(f_{base}) - o2 - \frac{s2}{12} \\
&= o1 + \frac{s1}{12} - o2 - \frac{s2}{12} \\
&= o1 - o2 + \frac{s1 - s2}{12} \\
&= \delta_o + \frac{\delta_s}{12}
\end{aligned}$$

5.3.2 Pitch shift

To get more variety in the training data we shift the pitch of our samples by up to 4 semitones in either direction and adjust the label accordingly. A correct pitch shift is impossible for a magnitude spectrum, since that would require phase information. Hence, we approximate this by shifting the contents of the logarithmic frequency bins by a certain amount of bins. The amount of octaves covered by our spectrum is $ld(2600) - ld(60) = 5.437$ where ld is the logarithm with base 2. This means we have $5.437 \times 12 = 65.249$ semitones in our spectrum and shifting by 1 bin represents a shift of $\frac{65.249}{113} = 0.577$ semitones. So, we can shift by $\lfloor \frac{n_{shift}}{0.577} \rceil$ bins. Finally, using the same algorithm we use for detuning described in Section 5.3.3, we can compensate for the error $err_{shift} = n_{shift} - \lfloor \frac{n_{shift}}{0.577} \rceil \times 0.577$ we made by rounding, this also guarantees that $|err_{shift}| < 0.577$.

5.3.3 Detuning

To simulate an improperly tuned instrument, an imperfect singer or musical accents like vibrato, we simulate a pitch shift of less then 0.4 semitones in either direction while leaving the label unchanged. This is done by calculating the overlap of the shifted bin and the neighboring bin and then just assigning it based on that percentage. As an example consider a shift by $+0.2885$, which represents a shift

App	R	RM	RMB	RMS	RMSB
RF-C	48.42	-	-	-	-
RF-R	36.84	-	-	-	-
KBK2	86.30	86.02	83.12	61.12	58.75
CNN	82.38	TBD	TBD	TBD	TBD

Table 9. Accuracy for the MIREX sub-tasks ((R) root note, (M) major and minor chords, (B) bass note, and (S) Sevenths) of our approaches (RF-C) Random Forest Classifier, (RF-R) Random Forest Regressors, and (CNN) Convolutional Neural Network with unseen data. As reference value (KBK2) the CNN-CRF from [2] is used. This table will be updated based on the results on the MIREX2018 chord estimation challenge once we get the results

of half a bin or mathematically:

$$L_i^j = \frac{0.2885}{0.577} \times L_i^{j-1} + \left(1 - \frac{0.2885}{0.577}\right) \times L_i^j$$

This can be efficiently computed using 1-D convolution. This formula is only correct for shifts smaller than one bin.

5.4 Training

The neural network was trained on the following data sets: Beatles, Queen and Zweieck [2], Robbie Williams [3], RWC Popular [4], and the public part of McGill Billboard [5]. We used a mini-batch size of 512 and the Adam optimizer. Additionally every convolutional layer in the network had a L_2 weight penalty with $\lambda = 10^{-7}$. For testing and parameter search we separated all songs from all our datasets into 8 approximately equally sized folds for 8-Fold cross-validation. While training our batches were composed of random samples from randomly drawn songs from the respective datasets.

6. RESULTS

In Table 9 we present the results of our approaches tested with unseen data from our training set. When we get the actual results from the MIREX validation, we will replace them.

The calculation of these values is given by MIREX and can be seen in [5]. The formula for calculation depending on the time where the estimated chord labels map the ground truth or not is as follows:

$$CSR = \frac{DurationWhereAnnotationEqualsEstimation}{TotalDurationOfAnnotatedSegments}$$

Sadly we were not able to beat last years top submission in the root note category, but we expected that, since there are a lot of little tricks and advanced methods that could be changed that are already part of previous submissions, that we didn't add. Methods like sequence decoding using an HMM or CRF, or heuristic methods like snapping chord changes to certain events (beats, note onsets, etc.).

[2] http://isophonics.net/datasets

[3] https://www.researchgate.net/publication/260399240_Chord_and_Harmony_annotations_of_the_first_five_albums_by_Robbie_Williams

[4] https://staff.aist.go.jp/m.goto/RWC-MDB/ and https://github.com/tmc323/Chord-Annotations

[5] http://ddmal.music.mcgill.ca/research/billboard

7. CONCLUSION

During our testing procedures for the fully CNN approach described in Section 5 we discovered that the data augmentation methods described by Korzeniowski and Widmer in [4] had little impact on our results, less than the differences between two consecutive runs which only differ in the sequence of random mini-batches that they receive. Sadly the only thing Korzeniowski and Widmer wrote regarding the data augmentation used in their approach is that it was crucial to prevent overfitting. Even though we have no concrete numbers this leads us to believe that the approach to separate chord estimation into the three objectives described in Section 3 was an important factor in preventing overfitting. In conclusion splitting a task into multiple meaningful sub objectives can lead to less overfitting.

8. REFERENCES

[1] Christopher Harte, Mark B. Sandler, Samer A. Abdallah, and Emilia Gómez. Symbolic representation of musical chords: A proposed syntax for text annotations. In *ISMIR 2005, 6th International Conference on Music Information Retrieval, London, UK, 11-15 September 2005, Proceedings*, pages 66–71, 2005.

[2] Filip Korzeniowski, Sebastian Böck, Florian Krebs, and Gerhard Widmer. Mirex submissions for chord recognition and key estimation 2017. *MIREX evaluation results*, 2017.

[3] Filip Korzeniowski and Gerhard Widmer. Feature learning for chord recognition: The deep chroma extractor. *CoRR*, abs/1612.05065, 2016.

[4] Filip Korzeniowski and Gerhard Widmer. A fully convolutional deep auditory model for musical chord recognition. *CoRR*, abs/1612.05082, 2016.

[5] Johan Pauwels and Geoffroy Peeters. Evaluating automatically estimated chord sequences. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013*, pages 749–753, 2013.