A MODULAR SYSTEM FOR THE HARMONIC ANALYSIS OF MUSICAL SCORES USING A LARGE VOCABULARY

Andrew McLeod
EPFL
andrew.mcleod@epfl.ch

Martin Rohrmeier
EPFL
martin.rohrmeier@epfl.ch

ABSTRACT

The harmonic analysis of a musical composition is a fundamental step towards understanding its structure. Central to this analysis is the labeling of segments of a piece with chord symbols and local key information. In this work, we propose a modular system for performing such a harmonic analysis, incorporating spelled pitches (i.e., not treating enharmonically equivalent pitches as identical) and using a very large vocabulary of 1540 chords (each with a root, type, and inversion) and 70 keys (with a tonic and mode), leading to a full harmonic characterization similar to Roman numeral analysis. Our system's modular design allows each of its components to model an aspect of harmony at an appropriate level of granularity, and also aids in both flexibility and interpretability. We show that our system improves upon a state-of-the-art model for the task, both on a previously available corpus consisting mostly of pieces from the Classical and Romantic eras of Western music, as well as on a much larger corpus spanning a wider range from the 16th through the 20th centuries.

1. INTRODUCTION

The analysis of the harmonic content of a musical composition or performance is fundamental to the field of MIR. It can take many forms, depending on the input format and desired output representation and specificity, but typically involves two basic steps: the segmentation of a musical input, and the labeling of each segment with a harmonic symbol from some vocabulary. The vocabulary is typically either a key or a chord symbol, but more recent work has begun to investigate a joint approach, identifying both.

A musical key can be defined by its tonic pitch and a mode. In this work, like most previous work, we only consider major and minor mode, but many others exist in practice (see [1] for a recent discussion). Most work on key detection, be it from audio [2–4] or some symbolic format [5–8], only classifies each piece as having a single key (i.e., its global key), disregarding any modulation to different local keys that might occur. One reason for this is simply the lack of labelled datasets with specific local key information, and the expertise required to produce (and evaluate) such annotations. More recently, however, some systems have addressed the problem of local key changes, incorporating a segmentation step into the key detection pipeline [9,10], or producing a continuous distribution over local keys throughout the duration of a piece [11].

Chord detection (also called chord transcription) and chord sequence prediction are very widely researched tasks in MIR, though the vocabulary used can vary dramatically. At its most basic, each chord in a vocabulary has a root pitch and a chord type (major triad, minor triad, etc.). Much existing work only has a limited chord vocabulary of size 24 (12 semitone pitch classes and either major or minor triads) [12, 13], often due again to the difficulty of creating datasets with more specific chord types (and the difficulty of inducing spelled pitches from MIDI or audio input). More recently, some work has included common additional triad types, such as diminished and augmented triads [14], as well as various 7th chords, and further extensions, like suspended or 9th chords [15, 16]. Additionally, each chord can have an inversion—describing which of its pitches is its bass note—the inclusion of which is also becoming more common [17].

In this work, we take as input a musical score, and label each segment with both a key and a chord symbol, including a very large vocabulary of 12 different chord types and inversions for each. As in standard Roman Numeral Analysis (RNA; [18]), this allows each chord's root pitch to be interpreted relative to the corresponding key's tonic pitch (including any local key modulations). In fact, our system's output is nearly equivalent to a full RNA, lacking only altered chordal tones such as suspensions, and pedal tones, which we intend to include in future work. We also use spelled pitches (where an A♯ is a different pitch from a B♭), which is still uncommon in existing work.

A few prior models have been proposed for the joint labelling of keys (including modulation) and chords. Statistical models have been used for the purpose [19–21], though they use an enharmonic MIDI pitch representation (where A♯ and B♭ are equivalent), a small vocabulary of chord types, and no inversions. Structurally, however, these models have somewhat inspired the design of our system (though they use Markov models instead of neural networks), explicitly modelling chord and key changes in a sequential fashion. More recently, deep learning models have also been proposed which use a large vocabulary of chord types as well as inversions. In [22], a transformer model is used, though it takes a piano-roll representation as input, and therefore uses the more reduced MIDI pitch representation rather than spelled pitch. This model is extended in [23] using the same input format, but now using spelled pitch output, though still using a reduced set of pitch classes compared to ours. Finally, in [24], a convolutional network is proposed for the task. This model uses the same pitch representation as ours, and roughly the same chord vocabulary, so we use it for comparison.

These existing deep learning models consider sequential information either at the frame level (an eighth or 16th note) [22, 24], or by grouping frames together into larger blocks [23]. Our modular design is such that each component models one specific aspect of the full harmonic analysis task at the appropriate level, receiving only input that is relevant to that aspect, and at the appropriate step length. For example, the component which models chord progressions sees only chord symbols as input, once per chord, while the component that performs the chord segmentation sees individual notes as input. Our hypothesis is that this design allows each component to better capture the patterns relevant to the task at hand.

The modular design also gives a practical, benefit over the single-model end-to-end design that is common in existing work. Its output is highly interpretable, which has helped significantly in its development (we note a few specific instances of this in the paper). In particular, it enabled the following development process: (1) locate mislabeled segments in its output; (2) examine the outputs of each component for that segment, comparing them to our intuition about the analysis (this is easy because each component corresponds to a well-defined aspect of the analysis); and (3) integrate additional features into the corresponding component, depending on our analysis.

2. PROPOSED MODEL

2.1 Vocabulary

We use a large vocabulary of chords and keys as they appear in scores, taking on a full characterization as used in music theory [18]. Chord roots may be any pitch A–G, double-flat to double-sharp (35 total), and we include 12 chord types: major, minor, augmented (each as a triad, or with a major or a minor 7th), and diminished (as a triad, or with a minor or diminished 7th). Each chord can be in any inversion (3 for triads, 4 for tetrads), for a total of 1540 possible chords. Keys may be major or minor with the same pitch range, for a total of 70 possible keys. Our model does not output applied chords (e.g., secondary dominants like V/V) directly. Rather, we treat them as brief, potentially recursively embedded, key changes as in [25].

2.2 Overview

Our system is composed of 6 modules, each with a well-defined input and output (see Figure 1). The system's input is a sequence of notes N ordered temporally by onset position, where notes with equal onset position are ordered by increasing pitch. The ith note in this sequence is denoted by n_i, and a note's onset position is denoted by $on(n_i)$. Specific implementation details (e.g., the precise encoding of each note) are explained in Section 2.3.

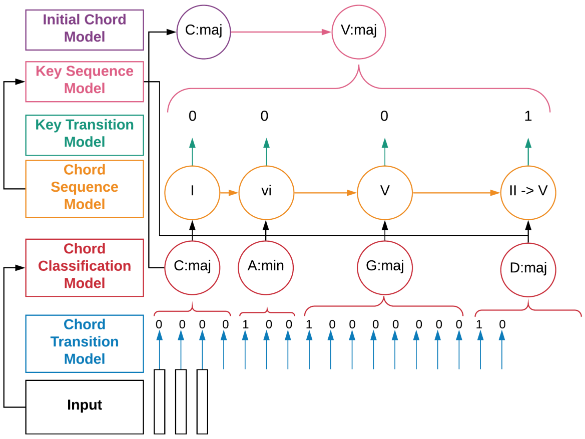

Figure 1. Overview of our fully integrated system. Each component depends on the component directly below it, and additional dependencies are indicated by arrows.

The *Chord Transition Model* (CTM) takes this sequence N and predicts whether each note is the first (temporally) of a new chord. We denote ct_i as the ith output of the CTM. Of these outputs, the first (ct_0) is set to 1, and any ct_i where $on(n_i) = on(n_{i-1})$ are set to 0. Thus, chord transitions correspond with vertical slices in the musical score.

The *Chord Classification Model* (CCM) takes a sub-sequence of notes from N, and outputs a distribution over all 1540 possible absolute chord symbols for it. We denote the sub-sequence from the ith to the jth note as $n_{i...j}$.

The *Chord Sequence Model* (CSM) takes a sequence of relative chord symbols (whose roots are represented as an interval above the tonic, not as an absolute pitch class), and outputs a distribution over the next relative chord in the sequence. We denote the $i$the relative chord symbol in this sequence as c_rel_i. At inference time, it's output always assumes that the key will not change on the next chord, and is unused in the case of a key change.

The *Key Transition Model* (KTM) takes as input a sequence of relative chord symbols starting from the beginning of the piece ($c\_rel_{0...i}$, $i > 1$, denoting the first $i + 1$ relative chord symbols), and predicts whether chord c_rel_i is in a different key than c_rel_{i-1}. At inference time, c_rel_i is initially represented relative to the key tonic of c_rel_{i-1}. Then, in the case of a key change, the hidden state of the KTM (if there is one) is reverted to its previous state, and given c_rel_i in the new key.

The *Key Sequence Model* (KSM) takes as input the same sequence of relative chord symbols as the KTM ($c\_rel_{0...i}$, $i > 1$), and outputs a probability distribution over a new key for c_rel_i, including both the mode (major or minor) and a tonic pitch class (represented as an interval over the previous key's tonic pitch class). It's output is only used when there is a key change and ignored otherwise. Similarly to the KTM, at inference time, c_rel_i is initially represented relative to the key tonic of c_rel_{i-1}. Then, after the key change, the hidden state of the KSM (if there is one) is reverted to its previous state, and given the updated c_rel_i in the new key.

Finally, the *Initial Chord Model* (ICM) outputs a prior probability distribution over the first relative chord of a piece c_rel_0 given a mode, and is used in combination with the first absolute chord symbol to generate a distribution over possible first absolute keys for the annotation. For example, the ICM may determine that the c_rel_0 in major mode has a 0.4 probability of being a I chord. It is used as a replacement for the CSM for the first chord.

2.3 Implementation

Due to the modular design of our system, the precise implementation of each model (including the representations used therein) is very flexible. For example, although the CTM is well-defined to take as input a sequence of notes in a particular order, the precise way in which each note is represented is left as an implementation detail, and can easily be changed without affecting the other models. Such details, as well as the design of each model individually, are described in this section.

As input to the CTM, each note n_i is represented as the concatenation of one-hot vectors (or scalars) encoding certain musical features of each note: the note's pitch class A–G, double-flat to double-sharp (one-hot of length 35); the note's octave 0–11, where C4 is in octave 6 to allow for negative octaves (one-hot of length 12); the note's normalized MIDI pitch height (0–1), where C-1 (MIDI note 0) = 0 and G9 (MIDI note 127) = 1 (scalar); the metrical levels of the note's onset and offset positions (downbeat, beat, sub-beat, or other; two one-hots of length 4); the duration of the note, where a whole note has duration 1 (scalar); and the duration from the note's onset to that of the previous note and that of the following note, again measured in whole notes (two scalars).

Our CTM is a neural network composed of a single feed-forward layer followed by a Bi-LSTM (each using ReLU activation). At each step, the LSTM's outputs are concatenated together and fed into two additional feed-forward layers (the first with ReLU activation and the last with sigmoid activation to produce a probability value).

As input to the CCM, each note vector is embedded into its musical context in two ways. First, we append to this sub-sequence the vectors of the two notes on each side of this sub-sequence (or vectors of zeros if this goes beyond the range of the piece). Second, we append to each note vector more musical features related to the note's context: onset and offset position relative to the chord window on a linear scale where the beginning of the chord is at position 0 and the end of the chord is at position 1 (2 scalars); duration as a proportion of the chord's duration (scalar); relative normalized pitch, where instead of C-1 being 0 and G9 being 1, the minimum pitch of a note in the window is 0 and the maximum pitch is 1 (scalar); and relative octave, where the octave of the lowest note in the window is subtracted from each note's octave before embedding (one-hot of length 12). The relative pitch and octave are examples of where our system's interpretability helped in its design: we noticed that the CCM was struggling to output the correct inversions when a passage doesn't contain any notes in a low octave (it had learned to depend on bass notes for this), and therefore added the relative features, which helped in that regard.

Its output is a distribution over all 1540 absolute chord symbols. It would be possible to treat each aspect of a chord symbol (root, type, and inversion) as a separate feature of each chord, and have the CCM output one distribution over each, as has been done in previous work (e.g., [24, 26]). However, while this approach makes sense in terms of reducing the size of a model, it doesn't make sense conceptually: there may be a situation in which the model sees a C in the bass, and thinks the chord is either a C major triad in root position or an A minor 7th chord in 1st inversion. In cases such as this, it is important that every feature of a chord is considered holistically as a unit, rather than potentially classifying the chord as a C minor 7th chord in 1st inversion.

Our CCM is a neural network composed of a single feed-forward layer followed by a Bi-LSTM (each using a ReLU activation). The outputs from the last LSTM state in each direction are concatenated together and fed into a single feed-forward layer with softmax activation.

The CSM, KTM, and KSM all take the same input, where each relative chord is encoded as a vector of concatenated musical features: the root pitch class and bass note pitch class, each represented as the interval above the key tonic on the line of fifths (-14–14; two one-hots of length 29); the chord type (one-hot of length 12); the inversion (one-hot of length 4); the metrical levels of the chord's onset and offset positions (same as for the note encoding; two one-hots of length 4); the duration of the chord in whole notes (scalar); and a flag indicating the current key's mode (1 for major; 0 for minor). At every chord at which there is a key change, we also append a key change vector, which encodes the following: the tonic of the new key, represented as the interval above the previous key's tonic on the line of fifths (-14–14; one-hot of length 29); the mode (major or minor; one-hot of length 2); and a flag indicating that this is a key change (binary 1). For each chord at which the key does not change, this key change vector is filled with all 0s.

The CSM's output is a distribution over chord symbols consisting of a root (-14–14 on the line of fifths from the key tonic), chord type, and inversion (1276 chords in total). Some of these chords may correspond with a valid absolute chord symbol (e.g., one with relative root -14 in the key of B♭♭). These are simply ignored. Likewise, some valid chords are not covered by the CSM's range (e.g., a B♯ chord in the key of B♭♭, which is exceedingly unlikely [27]). The prior for these chords is 0. Similarly, the KSM's output is a distribution over key symbols consisting of a tonic (-14–14 on the line of fifths from the previous key tonic) and a mode (58 keys in total). Invalid (e.g., 14 fifths below B♭♭ major) and uncovered (e.g., 20 fifths above B♭♭ major) key changes are treated in the same way.

Each of these models has a single feed-forward layer, followed by an LSTM whose last output is sent through another feed-forward layer (all with ReLU activations). A final feed-forward layer is then used, with softmax activation in the CSM and KSM, and sigmoid in the KTM.

Since there can be many reasonable choices for the next chord in a sequence, we noticed when inspecting the system's outputs that the CSM's distribution was more flat than desired. Therefore, we compare two additional versions of it: an inversion invariant CSM-I, which outputs a distribution over relative roots and chord types (348 chords in total), in which case probabilities are shared between different inversions of the same chord; and a triad-reduced CSM-T, which is includes the CSM-I's inversion invariance, and further shares weights between chords built upon the same triad (e.g., a V7, Vmaj7, and V all share outputs).

The ICM is a much simpler model than the rest, and we simply count the proportion of each chord—grouped by chord type, inversion, and relative root—as the first chord of a piece in our training data. We then apply additive smoothing to this estimate, adding $\frac{1}{1540}$ (where 1540 is the number of chords in our vocabulary) to each count.

2.4 Inference

Given the interactions between the modules of our system, the inference procedure required to use it to label a score is not trivial. However, since each component can be treated as a black box, once such a process is defined, we can flexibly use different modules interchangeably. This section gives an overview of the inference process, which, at its core, is the process of finding the most probable labeling of the musical score according to our system. We explore the search space iteratively using beam search decoding.

First, we run the CTM on the full input sequence N, generating ct_i for each note n_i. Next, we find all valid chord windows for the piece. For a chord window from $n_{i\ldots j}$ to be valid, six conditions must be satisfied: (1) $ct_i \geq ct_{min}$, where ct_{min} is a minimum threshold for a chord change; (2) $ct_{j+1} \geq ct_{min}$ (unless $j = |N| - 1$); (3) $ct_k \leq ct_{max}$ for all $i < k \leq j$ (ct_{max} is a maximum threshold for allowing a note to *not* be a chord change); (4) the duration of the resulting chord window (i.e., $on(n_{j+1}) - on(n_i)$) must be less than a maximum value C_dur_{max}; (5) a valid chord window exists which ends at n_{i-1} (unless $i = 0$); and (6) a valid chord window exists which begins at n_{j+1} (unless $j = |N| - 1$). (1)–(3) together ensure that the CTM is not ignored; (4) ensures that the resulting chord labeling is well-formed (extremely long passages with no chord changes are quite rare); and (5) and (6) ensure that each chord window can be part of a complete labeling of the musical score.

Having found all possible chord windows, the search process involves finding the most probable complete and labeled path through the score. A complete path C is a sequence of consecutive chord windows $c_{0\ldots|C|-1}$, where $c_m = n_{i_m \ldots j_m}$ ($i_0 = 0$, $j_{|C|-1} = |N| - 1$, and $\forall m, i_m = j_{m-1} + 1$). A labeled path is a complete path where each chord window c_m therein is labeled by assigning it a chord ($Ch(c_m)$) and a key ($K(c_m)$), with the constraint that no chords are repeated (i.e., $\forall m, Ch(c_m) \neq Ch(c_{m+1})$).

One exception to the above constraints is the *merge* rule, which allows the CCM to override the CTM's ct_{min} threshold in some cases. It is legal for two consecutive chord windows c_m and c_{m+1} to be merged if: (1) the resulting chord window's duration ($on(n_{j_{m+1}+1}) - on(n_{i_m})$) is still less than C_dur_{max}; and (2) $Ch(c_m) = Ch(c_{m+1})$ (which only occurs if the CCM assigns each a high probability). The merge step can be repeated as many times as possible as long as the two constraints are still met. This rule is another example of the interpretability of our system helping in its development: we noticed that the CTM was over-segmenting the input, and the thresholds were difficult to tune. However, the CCM's outputs were relatively accurate. Thus, we created the merge rule, which allows us to tune the CTM thresholds to over-transition, letting the more accurate CCM merge windows later.

Given a complete labeled path through a score, its probability is the product of the probabilities of each chord window c_m, where an individual chord window's probability is calculated as a product of its CTM probability $P_{ct}(c_m)$, its CCM probability $P_{cc}(c_m)$, its KTM probability $P_{kt}(c_m)$, and its sequence probability $P_{seq}(c_m)$, as shown in Eqn. 1. The exponent $|c_m|$ is used so that each path's probability is a product of an equal number of probabilities, no matter the number of chord windows (without this exponent, a path with fewer, longer windows would be preferred).

$$P(c_m) = P_{ct}(c_m)\big(P_{cc}(c_m)P_{kt}(c_m)P_{seq}(c_m)\big)^{|c_m|} \quad (1)$$

A window's CTM probability is calculated as in Eqn. 2 (ct_{j_m+1} is ignored if $m = |C| - 1$).

$$P_{ct}(c_m) = ct_{i_m} ct_{j_m+1} \prod_{k=i_m+1}^{j_m} (1 - ct_k) \quad (2)$$

A window's CCM probability is the CCM prior for the range $n_{i_m \cdots j_m}$ for the assigned chord $Ch(c_m)$. A window's KTM probability is likewise taken directly from the KTM: it is either the KTM's output at that window (if $K(c_m) \neq K(c_{m-1})$), or one minus the KTM's output (in all other cases). The first KTM probability ($P_{kt}(c_0)$) is always 1, since the key will never change on the first chord.

A window's sequence probability (the probability of a chord and key given the previous) is taken from the ICM ($P_{ic}(c_m)$) for $m = 0$, the KSM ($P_{ks}(c_m)$) if there is a key change, or the CSM ($P_{cs}(c_m)$) otherwise, as shown in Eqn. 3. Initially, the system predicted far too many key changes, and inspecting its output, made the cause clear: since the KSM outputs a distribution over 70 keys, while the ICM and CSM output distributions over more than 1000 chords, the KSM's outputs will naturally be greater. Thus, we introduced a parameter α, ensuring that the values output by each model lie in a similar range.

$$P_{seq}(c_m) = \begin{cases} P_{ic}(c_m) & \text{if } m = 0 \\ P_{ks}(c_m)^{\alpha} & \text{if } K(c_m) \neq K(c_{m-1}) \\ P_{cs}(c_m) & \text{otherwise} \end{cases} \quad (3)$$

3. EXPERIMENTS

3.1 Setup

We use two different corpora: (1) an internal corpus of harmonic annotations, including the publicly available Mozart [28] and ABC [29] corpora, and additional pieces which are private, but pending future release; and (2) functional-harmony (F-H) [24], consisting of a combination of prior existing corpora: TAVERN [30], BPS-FH [22], and Roman Text [31]. The internal corpus contains a much wider range of composers: 742 pieces from the 16th to the 20th centuries, with J.S. Bach, Couperin, Grieg, Beethoven, Schütz, Mozart, Corelli, Chopin, Kozeluh, Monteverdi, Mendelssohn, and Schubert all having more than 20 pieces, and 99 by other composers. F-H is smaller: 201 pieces in total, with 119 by Beethoven, 29 by Schubert, 24 by J.S. Bach, and 19 by other composers. We treat the two corpora separately, and in each, randomly take 80% for training, and 10% each for testing and validation.

For evaluation, we use a type of Chord Symbol Recall (CSR) [32]: for each piece, the proportion of time during which the estimated label matches the ground truth label. Given our large vocabulary, we apply CSR at various levels of specificity, similar to [24] [1]. We report CSR with regards to the chord root, chord root+type, chord root+type+inversion (the full chord), key (ignoring the chord), and full (including chord and key). We intend to investigate more advanced metrics (e.g., where some label substitutions are penalized less than others) in future work.

All components of our system were trained with *Adam* [33] and a learning rate of 0.001. A scheduler cut the learning rate in half when the validation loss didn't improve for 10 epochs, and training was stopped when the validation loss failed to improve for 20 epochs. When training the CCM, we transposed each input and target by -7–7 fifths (a similar data augmentation technique was used by [24]).

For this study, we did not perform a large grid search over deep model structures (e.g., more feed-forward layers). Rather, we grid searched over layer sizes of 64, 128, and 256 for each of the feed-forward and LSTM layers of each model. The system-wide inference parameters were set using a grid search on the validation set of each corpus, resulting in ct_{max} and ct_{min} of 0.3 and 0.4 for the F-H corpus, and 0.35 and 0.55 for the internal corpus; C_dur_{max} of 10 for both corpora (this parameter has little effect); and α of 30 for the F-H corpus and 50 for internal corpus. α has by far the largest effect of any parameter, where low values result in much more frequent key changes. We leave an investigation of the performance of each module for future work, concentrating here on overall performance.

3.2 Results

We compare versions of our system using the standard CSM, the inversion invariant CSM-I, and the triad-reduced CSM-T. As a baseline, we use the pre-trained model of [24], computing new results on our internal corpus, and using the full CSR from [24] for the F-H corpus (the component-wise metrics are calculated differently). The results can be found in Table 1. Here, we can see that our modular system outperforms the baseline on both the internal corpus and, more notably, on the F-H corpus.

	Model	Root	+Type	+Inv.	Key	Full
Internal	[24]	57.0	47.7	37.6	64.9	29.0
	CSM	76.6	68.8	62.1	66.9	44.7
	CSM-I	76.5	68.7	62.0	69.0	46.3
	CSM-T	**77.6**	**70.0**	**62.8**	**70.2**	**46.9**
F-H	[24][2]	—	—	—	—	42.8
	CSM	73.3	65.4	55.6	60.8	40.5
	CSM-I	75.0	66.8	56.9	67.0	44.6
	CSM-T	**75.4**	**67.8**	**58.1**	**69.4**	**45.9**

Table 1. The results of our system compared to the model of [24] on our internal corpus of annotations (top), and the functional-harmony meta-corpus used in [24] (bottom).

There seems to be a systematic difference between the annotations of the two corpora, given the relatively poor performance of the pre-trained baseline [24] on the internal corpus. We would expect this to be due to the wider range of styles, but while this does appear to play some role, the baseline actually performs *worse* (22.9 overall) on a subset of the internal corpus' test set including only Beethoven pieces. We plan to more thoroughly investigate differences between the annotations of the two corpora in future work.

Interestingly, the baseline performs relatively well on key detection (even on our internal corpus), which points to one downside of our modular approach: the key depends on outputs from the other components, adding noise to the process, which isn't a factor for the end-to-end baseline.

For our system, we see that the three versions perform similarly on the internal corpus, but the two invariant CSMs outperform the standard one—significantly on the F-H corpus. Interestingly, this difference mainly shows in a more accurate key labelling, rather than chord. Inspecting the results, we see that the standard CSM usually assigns a low probability to the more rare chords like minor 7th chords or inverted chords. This makes the model prefer key changes in these cases, which avoid using the CSM's output distribution. The CSM-I and CSM-T avoid this problem due to their invariances. In future work, we intend to investigate alternate reductions where, for example, dominant 7th chords (which are quite common) are not reduced to major in the CSM-T.

Figure 3 shows the CSM-I's chord classification performance on the internal corpus as a confusion matrix. It performs best on major triads, minor triads, and dominant 7th chords, which comprise 44%, 22%, and 18% of the corpus, respectively. For the less common chord types, when the root is correct, the system often misclassifies the chord

[1] In the metrics reported in [24], applied roots are not considered key changes as they are here, but instead included in the "Degree" metric.

[2] The values in [24] are calculated slightly differently: applied chords are included in Root (rather than Key here), and Type and Inversion refer to only those features of a chord (in [24]), rather than in combination with those to their left. "Full" is comparable.

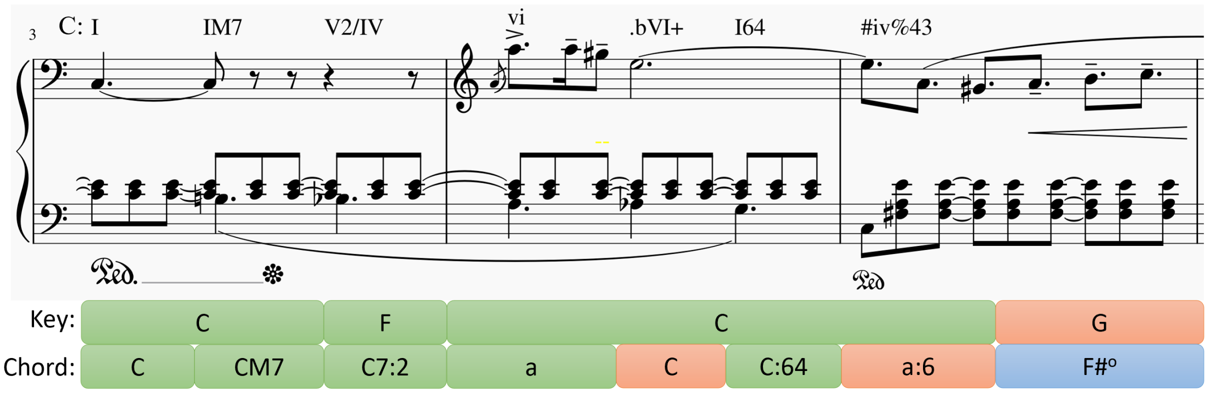

Figure 2. The CSM-I system's output (below) for bars 3–5 of Grieg's Notturno, Op. 54, No. 4, given the annotations (above). Green indicates a correct label, red is incorrect, and blue is partially correct. Inversions are indicated by the figured bass after each colon.

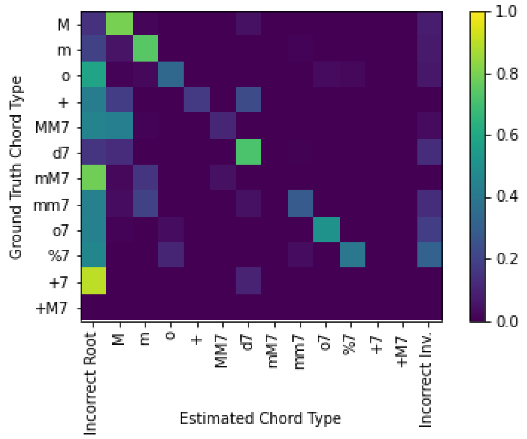

Figure 3. A confusion matrix of the CSM-I system's chord classifications on the internal corpus. "Incorrect Inv." indicates the proportion of otherwise correct labels with an incorrect inversion.

type as the corresponding triad (e.g., MM7 chords are classified as major triads). In the rightmost column, we see that if the system gets the correct root and type, the inversion is usually correct, with the largest proportion of errors occurring for diminished and half-diminished 7th chords.

Figure 4 shows CSR split by mode and chord type, where it performs slightly better in minor keys, particularly for minor triads and diminished triads and 7th chords. Overall, it achieves 44.6 CSR in minor keys but only 40.9 in major. Chord inversions also have a large effect on performance, with our system classifying 71.5% of root position chords correctly, but only 54.4%, 38.3%, and 40.5% of 1st, 2nd, and 3rd inversion chords correctly, respectively. This makes sense, because root position chords make up 38% of the corpus, while 1st, 2nd, and 3rd inversion chords comprise 25.4%, 8.8%, and 3.7% respectively.

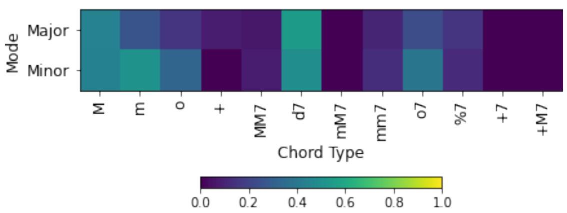

Figure 4. The CSM-I system's full CSR (including key and chord) per mode and chord type on the internal corpus.

An example output of the CSM-I is shown in Figure 2, for bars 3–5 of Grieg's Notturno, Op. 54, No. 4. The annotations (in C major) are shown above the staff, and outputs are below it. The outputs are correct in bar 3, finding the applied dominant V2/IV as a key change to F. In bar 4, it incorrectly classifies the augmented A♭ triad (a ♭VI, uncommon prior to the late Romantic era) as a C major triad, ignoring the A♭. It over-segments bar 5, finding an inverted A minor triad (ignoring the F♯) followed by an F♯ diminished triad (since it has missed the C from the downbeat), and modulates to G major (essentially, it has classified this chord as a ♯vii°/V, which is possible, but incorrect here).

4. CONCLUSION

We have presented a modular system for the harmonic analysis of musical scores. We showed how the system's modularity, in addition to the theoretical benefit of modelling each aspect at the appropriate level, makes its output interpretable, describing where this property helped in its design. All code and models are available online. [3]

In future work, we will leverage the system's modularity further. We intend to apply the system to MIDI and audio input, in which case we would only need to retrain the CTM and CCM with new data, while the other components can remain the same, or be supplemented with additional training data from the MIDI and audio files. We also intend to design a human-in-the-loop annotation tool using this system, where expert annotators can first get the system's output, change some labels as they see fit, and then re-run the search process, constraining the system to a path that includes the manually corrected labels. This process could be faster than the current fully manual approach to harmonic annotation.

[3] http://github.com/apmcleod/harmonic-inference

5. ACKNOWLEDGMENTS

Research supported through the Swiss National Science Foundation within the project "Distant Listening – The Development of Harmony over Three Centuries (1700–2000)" (Grant no. 182811). This project was conducted at the Latour Chair in Digital and Cognitive Musicology, generously funded by Mr. Claude Latour.

6. REFERENCES

[1] D. Harasim, F. C. Moss, M. Ramirez, and M. Rohrmeier, "Exploring the foundations of tonality: statistical cognitive modeling of modes in the history of Western classical music," *Humanities and Social Sciences Communications*, vol. 8, no. 1, pp. 1–11, dec 2021.

[2] Á. Faraldo, S. Jordà, and P. Herrera, "A Multi-Profile Method for Key Estimation in EDM," in *AES International Conference on Semantic Audio*. Audio Engineering Society, jun 2017.

[3] G. Bernardes, M. E. P. Davies, and C. Guedes, "Automatic musical key estimation with adaptive mode bias," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, mar 2017, pp. 316–320.

[4] F. Korzeniowski and G. Widmer, "Genre-agnostic key classification with convolutional neural networks," in *ISMIR*, 2018.

[5] H. C. Longuet-Higgins and M. Steedman, "On Interpreting Bach," *Machine Intelligence*, vol. 6, pp. 221–241, 1971.

[6] C. L. Krumhansl, *Cognitive foundations of musical pitch*. Oxford University Press, 2001.

[7] D. Temperley, "What's Key for Key? The Krumhansl-Schmuckler Key-Finding Algorithm Reconsidered," *Music Perception*, vol. 17, no. 1, pp. 65–100, 1999.

[8] D. Temperley and E. W. Marvin, "Pitch-Class Distribution and the Identification of Key," *Music Perception: An Interdisciplinary Journal*, vol. 25, no. 3, pp. 193–212, mar 2008.

[9] C. Weiss, H. Schreiber, and M. Mueller, "Local Key Estimation in Music Recordings: A Case Study Across Songs, Versions, and Annotators," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, pp. 1–1, oct 2020.

[10] L. Feisthauer, L. Bigo, M. Giraud, and F. Levé, "Estimating keys and modulations in musical pieces," in *17th Sound and Music Computing Conference*, 2020.

[11] R. Lieck and M. Rohrmeier, "Modelling hierarchical key structure with pitch scapes," in *ISMIR*, 2020.

[12] H. Papadopoulos and G. Peeters, "Joint Estimation of Chords and Downbeats From an Audio Signal," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 1, pp. 138–152, jan 2011.

[13] F. Korzeniowski and G. Widmer, "Improved chord recognition by combining duration and harmonic language models," in *ISMIR*, 2018.

[14] Y. Wu, T. Carsault, E. Nakamura, and K. Yoshii, "Semi-supervised Neural Chord Estimation Based on a Variational Autoencoder with Latent Chord Labels and Features," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, pp. 1–1, 2020.

[15] T. Rocher, M. Robine, P. Hanna, and R. Strandh, "Dynamic chord analysis for symbolic music," in *International Computer Music Conference*, 2009, pp. 41–48.

[16] T. Carsault, A. McLeod, P. Esling, J. Nika, E. Nakamura, and K. Yoshii, "Multi-Step Chord Sequence Prediction Based on Aggregated Multi-Scale Encoder-Decoder Networks," in *IEEE International Workshop on Machine Learning for Signal Processing, MLSP*, vol. 2019-Octob, 2019.

[17] B. McFee and J. P. Bello, "Structured training for large-vocabulary chord recognition," in *ISMIR*, 2017.

[18] E. Aldwell, C. Schachter, and A. Cadwallader, *Harmony and voice leading*. Cengage Learning, 2018.

[19] C. Raphael and J. Stoddard, "Functional harmonic analysis using probabilistic models," *Computer Music Journal*, vol. 28, no. 3, pp. 45–52, 2004.

[20] D. Temperley, "A Unified Probabilistic Model for Polyphonic Music Analysis," *Journal of New Music Research*, vol. 38, no. 1, pp. 3–18, mar 2009.

[21] C. Aitken, T. O'Donnell, and M. A. Rohrmeier, "A Maximum Likelihood Model for the Harmonic Analysis of Symbolic Music," in *Proceedings of the Sound and Music Computing Conference (SMC)*, Cyprus, 2018.

[22] T. P. Chen and L. Su, "Functional harmony recognition of symbolic music data with multi-task recurrent neural networks," in *ISMIR*, 2018, pp. 90–97.

[23] T.-P. Chen and L. Su, "Attend to chords: Improving harmonic analysis of symbolic music using transformer-based models," *Transactions of the International Society for Music Information Retrieval*, vol. 4, no. 1, 2021.

[24] G. Micchi, M. Gotham, and M. Giraud, "Not all roads lead to rome: Pitch representation and model architecture for automatic harmonic analysis," *Transactions of the International Society for Music Information Retrieval (TISMIR)*, vol. 3, no. 1, pp. 42–54, 2020.

[25] M. Rohrmeier, "The syntax of jazz harmony: Diatonic tonality, phrase structure, and form," *Music Theory and Analysis (MTA)*, vol. 7, no. 1, pp. 1–63, 2020.

[26] T.-P. Chen and L. Su, "Harmony Transformer: Incorporating Chord Segmentation into Harmony Recognition," in *ISMIR*, nov 2019.

[27] F. Moss, "Transitions of tonality: a model-based corpus study," Ph.D. dissertation, 2019.

[28] J. Hentschel, M. Neuwirth, and M. Rohrmeier, "The Annotated Mozart Sonatas: Score, Harmony, and Cadence," *Transactions of the International Society for Music Information Retrieval*, vol. 4, no. 1, pp. 1–14, 2021.

[29] M. Neuwirth, D. Harasim, F. C. Moss, and M. Rohrmeier, "The annotated beethoven corpus (ABC): A dataset of harmonic analyses of all beethoven string quartets," *Frontiers in Digital Humanities*, vol. 5, p. 16, 2018.

[30] J. Devaney, C. Arthur, N. Condit-Schultz, and K. Nisula, "Theme and variation encodings with roman numerals (tavern): A new data set for symbolic music analysis," in *ISMIR*, 2015.

[31] D. Tymoczko, M. Gotham, M. S. Cuthbert, and C. Ariza, "The romantext format: A flexible and standard method for representing roman numeral analyses," in *ISMIR*, 2019.

[32] C. Harte, "Towards automatic extraction of harmony information from music signals," Ph.D. dissertation, Queen Mary University of London, 2010.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.