DETECTING CHORD TONE ALTERATIONS AND SUSPENSIONS [*]

Andrew McLeod
Semantic Music Technologies
Fraunhofer IDMT
Ilmenau
Germany
andrew.mcleod@idmt.fraunhofer.de

Martin Rohrmeier
Digital and Cognitive Musicology Lab
École Polytechnique Fédérale de Lausanne (EPFL)
Lausanne
Switzerland

ABSTRACT

Chord detection is the task of assigning chord labels to segments of an input musical piece. The labels are typically drawn from a pre-defined vocabulary in which each symbol comprises at least a root pitch and a chord quality. In this paper, we introduce a post-processing method that takes as input a musical score with such labels and adds chord tone alterations such as suspensions to them. Although it is an important aspect of harmonic analysis, it is infeasible to do so by simply adding features to the labels in the vocabulary, given the multiplicative effect on vocabulary sizes which can already be over 1000. We evaluate our method using both ground truth input and input derived from a chord detection model, showing that it outperforms a strong heuristic baseline in both cases. Furthermore, we show that our model can be used to detect the rarer chord qualities from the model's output alterations, potentially enabling chord detection models to be trained on small vocabularies containing only the more common (and easier to identify) chord qualities (e.g., only triads).[2]

***K*eywords** chord tone · alterations · suspensions · chord labeling · harmonic analysis

1 Introduction

In general terms, chord detection involves the assigning of chord labels to segments of an input musical piece, which can be audio or some symbolic format such as MIDI or a musical score (e.g., Korzeniowski & Widmer, 2018; Temperley & Sleator, 1999; Zhou & Lerch, 2015). Chord labels are typically drawn from a pre-defined vocabulary, where each symbol has at least a root pitch class and a chord quality (major triad, minor triad, etc.), but may also include other features such as inversion or cadential information. While the smallest vocabularies contain only 24 chord symbols (12 enharmonic pitch classes with only major and minor triads), in recent years, chord vocabularies have continued to grow, with some now containing over 1000 unique symbols. McLeod and Rohrmeier (2021), for example, use a vocabulary of 1540 symbols for their harmonic analysis model, including 35 spelled pitch classes for the root, 12 chord qualities, and 3 or 4 inversions, and this is not an outlier. Indeed, in recent years, automatic harmonic analysis has become quite a popular task, with many other models also being proposed, each using a similarly-sized (or larger) vocabulary (e.g., Chen & Su, 2021; Nápoles López et al., 2021; Micchi et al., 2021; Karystinaios & Widmer, 2023).

Chord extensions, (chromatic) alterations of chord tones (e.g., harmonic mixture), and voice-leading transformations (e.g., suspensions) are important aspects of traditional Western classical harmonic analyses (the focus of the above models; see Aldwell et al., 2018), and of much of harmonic analysis in general (e.g., in jazz, chord extensions and alterations are the norm; Levine, 2011). For example, the cadential 64—where the fifth and third of a V chord in a V-I cadence are each omitted at first in order to be prepared by an upper diatonic step—is relatively common. This and similar voice-leading transformations, as well as tonal alterations, play an important role in a listener's experience, exploiting the effects of expectation and tonal tension (Huron, 2008). Generally, such complex phenomena are construed

[*]*Citation*: **Andrew McLeod and Martin Rohrmeier. Detecting Chord Tone Alterations and Suspensions. *Journal of New Music Research*. DOI:10.1080/09298215.2024.2412595.**

[2]The Version of Record of this manuscript has been published and is available in Journal of New Music Research 11 October 2024 http://www.tandfonline.com/10.1080/09298215.2024.2412595.

in music theory as basic chord qualities being manipulated. Together, they cannot be understood until each note is related to the underlying chord. Therefore, the interpretive process of harmonic analysis generates insights into the function and harmonic or contrapuntal origin of each note in a composition.

Nonetheless, to our knowledge, no chord labeling model has yet been proposed that includes an analysis of such complex chord forms directly in their output chord labels. One possible reason for this is clear: adding an additional feature to each chord label results in a multiplicative increase in vocabulary size (since each alteration might occur with any existing label), which reduces the possible training data per label and weakens predictive power. Furthermore, since they are relatively rare compared to the accuracy of current analysis systems, researchers may have preferred to continue to improve their models of chord and key annotation, rather than add additional features. However, it is highly desirable for models—that should be useful for musicologists—to be able to express the full theoretical chord spectrum, rather than a highly impoverished subset that does not reflect the degree of subtlety and expression found in human expert analyses (e.g., Hentschel, Neuwirth, & Rohrmeier, 2021).

While there are many approaches to solving a problem with such a large vocabulary, the most direct is to treat the alterations as a classification problem: Gather the full set of possible suspensions (as strings) from existing corpora and rely on a model to learn what a “4” or “64” suspension implies in terms of pitch content. This is certainly possible and could be done in two ways. One would be to incorporate the alterations into the vocabulary directly, though this runs into the multiplicative problem discussed above. A second option is to treat the alterations as a separate classification problem entirely. Care would have to be taken to ensure that the chosen alteration aligns with the chosen chord label (e.g., C^6_4 and F would each be reasonable chord labels given the pitch classes C, F, and A, while the chord labels C and F^6_4 would not), but it is possible to inject this dependence into a model, as in the model proposed by Micchi et al. (2021).

Still, treating chord tone alterations as a classification problem runs into a slightly subtler issue: The meaning of an alteration is often context dependent. For example, C^4 in the key of C major implies the pitches C, F, and G, while C^4 in the key of G major implies the pitches C, F♯, and G ($\text{C}^{\natural 4}$ would rather imply C, F, and G). This disconnect is not impossible for a model to learn, but it can complicate the learning process. Furthermore, it suggests that a different way of looking at the problem might lead to a simpler, more direct solution.

We considered (and tried) two options, one of which we present in this paper. Both take as input a chord label (and the notes present in the score for the duration of that label) and output a set of pitch classes (PCs) that should be incorporated into the label. Then, a simple heuristic is used to incorporate the output PC set into the chord label. In a first attempt, we trained a model to output this PC set directly, but this led to strange errors where the model would output a PC that was never present during a given chord, just because that alteration was common in a particular context. Our next attempt, which we present in this paper, is a model that labels each note within a chord’s duration as a chord tone (a PC that should be included in the chord label) or a non-chord tone, followed by a post-processing procedure that translates the chord tone labels into a PC vector and finally a specific alteration.

It is worth noting that a similar approach was tried by Ju et al. (2017), and the model exhibited good performance, although there are some notable differences to our proposal. For one, their model is designed to work specifically on 4-part Chorales, and its input and output format require such music, while ours can be applied to any musical score. Their model also treats enharmonically equivalent PCs as identical (i.e., it considers only 12 PCs), while ours uses 35 (double-flat to double-sharp for A-G). Finally, our method takes a hypothesis chord label as input, while theirs only takes the PCs.

Importantly, we have designed our method in a way that it can be applied as a post-processing step given the output of any harmonic analysis model, including those cited above, or any future models proposed for the task, and we present an experiment to that effect (taking noisy input form a chord labeling model) in Section 3.4. Finally, we believe that our approach could be used to circumvent the problem of ever-growing vocabularies. By training chord labeling models on a reduced set of chord qualities, their accuracy will increase, and if the missing chord qualities can be inferred by our proposed method (e.g., by treating 7th chords as triads with an added 7th), it is possible that overall label accuracy may also improve. We investigate this further in our experiments in Section 3. All code for the method described here can be found online.[3]

2 Proposed Method

Our proposed method consists of three steps. First, the Chord Pitches Model (CPM) assigns each note within a chord a value between 0 and 1 indicating the probability that the given note is “important”—that it is not an ornamentation and

[3]`www.github.com/apmcleod/harmonic-inference`

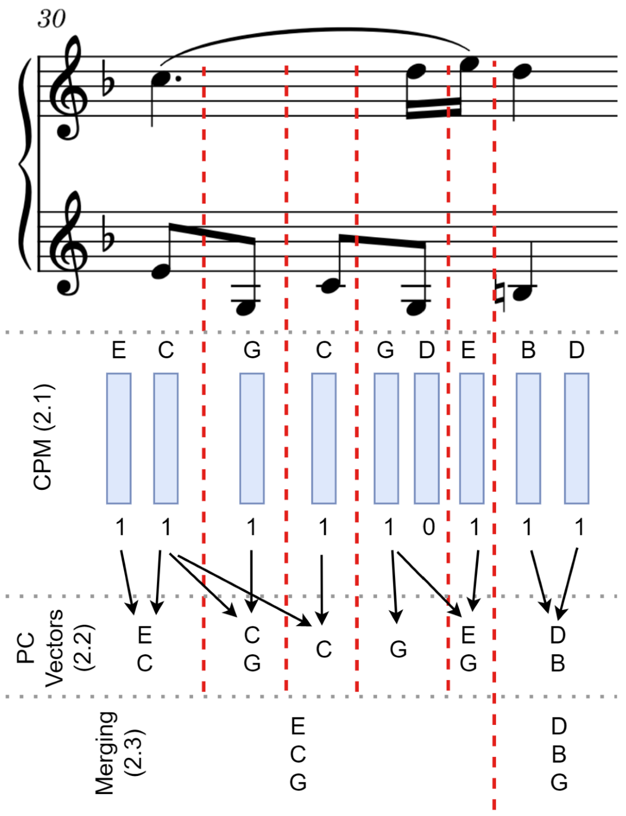

Figure 1: A diagram of our proposed method, including the CPM (Section 2.1), the PC vector creation described in Section 2.2, and the merging procedure from Section 2.3. The excerpt is from Mozart's Sonata in F major, KV 280, measure 30. The chord label is G^6_4 for the first two beats, and G for the final beat.

rather belongs in the chord label (Section 2.1). Next, the chord is split into windows at each note onset or offset, and a pitch class (PC) probability vector is created for each resulting window, estimating, for each PC, the probability that a note of that PC is important in the window. A rule-based process is then used to convert these probability vectors into binary vectors (see Section 2.2). Finally, neighboring binary PC vectors are iteratively merged (when allowed according to some heuristics), and the final binary PC vector (or vectors, if at least one merge was disallowed) is the output of our method (see Section 2.3).

We also tested a version of our method where instead of the manual, heuristic-based post-processing of the CPM, we instead threshold its output and use the resulting vectors directly as the binary PC vectors. However, due to the highly skewed nature of the data, we were not able to get such a design to work (it only ever output the default PC vector for a given chord label). We therefore redesigned our method to allow the neural network component to concentrate on a much simpler task (whether a single note is a chord tone or not) and switched to heuristic-based post-processing. Our current method is also much more similar to how a music theorist would approach the problem, identifying key notes and building the chord label based on those.

Figure 1 shows a diagram of our method, which we reference throughout this section.

2.1 Chord Pitches Model (CPM)

The input to the CPM is a sequence of vectors, each representing a single unique note from the input (shown as blue boxes in Figure 1). First, the notes of a piece are put into a sequence N in order of increasing onset position, with notes of a lower pitch coming first when onset position is identical. Then for a given chord, the CPM's input sequence corresponds to a subsequence of N from note $f - c$ to note $l + c$, where f is the index of the first note which overlaps the chord, l is the index of the last note which overlaps the chord, and c is an integer context size, which can be changed. When these indices go before the first note or past the last note of N, the corresponding vectors are set to all 0's. The vector representation of a note in the context of a chord is the concatenation of many features:

- The PC of the note, represented as the number of perfect 5ths above the root (e.g., given a C chord, we have the values . . . , B♭=-2, F=-1, C=0, G=1, D=2, . . .). One-hot vector of length 39 (for notes whose PC is more than 19 fifths away from the chord root, this vector is filled with 0's).
- The note's octave, as a one-hot of length 11.
- The metrical level of the note's onset and offset positions (two one-hots of length 4 where 3=downbeat, 2=beat, 1=sub-beat, 0=other).
- The note's onset and offset positions, as proportions of the chord's total duration. That is, the beginning of the chord is at position 0, the end of the chord is at position 1, and all other positions are linearly interpolated (2 floats).
- The note's duration, as a proportion of the chord's duration, measured in whole notes, and measured in beats (3 floats).
- The distance from the note's onset position to the previous and next notes' onset positions, each measured in both whole notes and beats (4 floats).

Each of these note vectors is then concatenated with 3 additional chord vectors. For the previous and the next chord (the vector is all 0's if there is no such chord), this vector contains:

- The chord's root, represented as the number of perfect fifths above the current chord's root (one-hot of length 39, as with the note's PC).
- The chord's quality (one-hot of length 12), representing the 12 chord qualities we use: 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).
- The chord's inversion (one-hot of length 4).
- The metrical level of the chord's onset and offset positions (one-hots of length 4).
- The duration of the chord, measured in both whole notes and beats (2 floats).
- A binary value indicating whether the chord's local key is major or minor.
- A binary value indicating whether the chord is diatonic to its key.

For the current chord, this vector is identical, except its root is not included (since it would always be 0). The final length of each input vector is therefore 67 (note vector) + 67 (previous chord) + 67 (next chord) + 28 = 229.

Our proposed Chord Pitches Model is a neural network consisting of an initial feed-forward layer, one (or more) Bi-directional Long Short-term Memory (LSTM; Hochreiter & Schmidhuber, 1997) layers, one more feed-forward layer, and a final output node of size 1. All layers use Rectified Linear Unit (ReLU; Nair & Hinton, 2010) activation functions, except the output node which uses a sigmoid activation (since the model's output should lie between 0 and 1 for each input). Outputs for notes which do not overlap the chord window are ignored. For those notes that do overlap, their target is either 1 (if its PC is included in the ground truth chord label) or 0 (otherwise).

2.1.1 Training Strategy

We use a training strategy that encourages the CPM to be more "adventurous" with its outputs (without this strategy, it much more rarely outputs a 1 for anything but the chord's default PCs). Ours is closely related to Curriculum Learning (Bengio et al., 2009), in which training is altered to gradually focus the model on increasingly complex examples. We were also inspired by a strategy used for chord labeling by Rowe and Tzanetakis (2021), where a weighting factor inversely proportional to each chord quality's frequency was applied. This factor was gradually raised throughout training, forcing the model to first concentrate on the more common chord qualities before learning the rarer ones.

In our case, initially, the loss for the model is calculated as standard binary cross-entropy. However, once the model's validation loss does not improve for 10 epochs, we multiply the loss associated with chords where all PCs are default by a weight factor w of .5, (analogous to standard learning rate scheduling). This value then remains unchanged for the rest of the training procedure. This strategy has the similar effect of forcing the model to concentrate more heavily on the non-default chords (which are extremely rare) only once it has reached a good representation of the default chords. The default chord training then functions similarly to model pre-training, before we "fine tune" on the more rare chords. Variations to this procedure are presented and evaluated in Section 3.3.

2.2 Windowed Pitch Class Vectors

The input chord is divided into non-overlapping windows by splitting the input at each position within the chord at which there is a note onset or offset (red dashed lines in Figure 1). Then, for each of these windows, a length 27 PC vector is created. The 1st element represents the PC 13 5ths below the chord's root, the 2nd element represents the PC 12 5ths below the root, and so on up to the PC 13 5ths above the root. Each of these elements are set to the maximum of the CPM's output for any note within the window with that PC (or 0 if no such notes exist). Notice that notes may belong to multiple windows, as do the first C and the second G in Figure 1. This length vector was chosen to be wider than any examples in our dataset, such that all possible chords could be output by the model.

These continuous-valued PC vectors are then binarized as follows. First, any default (according to the chord's type) PC whose value is greater than a "default" threshold d, is set to 1. Any non-default PC whose value is greater than an "add" threshold a is set to 1 (a is typically lower than d, since the model has a strong bias towards outputting high values only for default PCs). Next, any non-default PC whose value is less than a "replacement" threshold r is set to 0. This leaves only those non-default PCs which are greater than r but less than a. For these pitches, we look for any neighboring default root, 3rd, or 5th PC which is not already 1 (we do not allow for altered 7ths). A neighboring PC is any which is exactly 1 step away (e.g., any D to any E, regardless of accidentals), except in the case of the root, when only PCs one step up are considered neighboring. We then choose the default/non-default pairing which is maximally spanning (i.e., the greatest possible number of non-default PCs are paired with unique neighboring default PCs). If multiple pairings are maximally spanning, we choose that with the greatest likelihood according to the CPM (measured as the sum of the probabilities of the paired non-default PCs minus the sum of the probabilities of the paired default PCs). Each pairing represents the non-default PC replacing its paired default PC in the chord realization. The paired non-default PCs are set to 1 in the PC vector, and all remaining PCs are set to 0. Figure 1 shows these vectors by listing the PCs associated with a 1 for each.

2.2.1 Augmented 6th Chords

In our model, we represent augmented 6th chords as standard triads or tetrads with altered pitches. French 6th chords are represented as V7/V$^{\flat 5}$, Italian 6th chords as vii^{o}/V$^{\flat 3}$ in 1st inversion, and German 6th chords as vii^{o7}/V$^{\flat 3}$ in 1st inversion (the slashes here represent applied chords in a tonicized section, while the superscripts after each chord notate chord tone alterations). To handle these, we expand the definition of "neighboring" (from the previous section) to include their alterations. First, a default PC which is a minor 3rd above the root in any diminished or diminished 7th chord is considered a neighbor of the PC which is a diminished 3rd above the root. Similarly, a default PC which is a major fifth above the root in any dominant 7th chord is considered a neighbor of the PC which is a diminished 5th above the root.

2.3 Merging

Neighboring PC vectors are iteratively merged using the algorithm shown in Figure 2, where *vectors* is the ordered list of binary PC vectors from the previous section. The algorithm also keeps track of the onset and offset points of each merged window (not shown in the pseudocode). Any remaining unmerged vectors after this process are treated as separate chord labels, as they necessarily contain incompatible PCs, and would therefore be represented by the same (duplicated) underlying label, but with distinct alterations.

The `can_merge` function is defined as follows. If the two vectors are identical, `True` is returned. If the index of each 1 in one of the vectors corresponds to a 1 in the other vector, `True` is returned. Finally, in the case that each vector has an "extra" 1 (which is 0 in the other vector), they can be merged if and only if, for each of these extra PCs in both vectors: (1) it is a default PC, and no extra PC in the other vector replaces it; or (2) it is a non-default PC, and no extra PC in the other vector is a replacement of it.

As a final step, for each merged PC vector, any default PC with a value of 0, but for which no replacement PC is present in the vector is set to 1. The bottom of Figure 1 shows these final PC vectors.

3 Results

3.1 Setup

3.1.1 Data

Our data consists of a mixture of internal (but to-be-released) data and publicly available corpora, including the Annotated Beethoven Corpus (ABC; Neuwirth et al., 2018), the Annotated Mozart Sonatas (Hentschel, Neuwirth, & Rohrmeier, 2021), and a corpus of 36 trio sonatas by Arcangelo Corelli (Hentschel, Moss et al., 2021). In total, our data consists of 924 pieces, each annotated with functional harmony labels, out of which we use chord root, type, and inversion, as well as suspensions and added notes. We split the data into 80% for training, 10% for validation, and 10% for testing.

```
merged ⟵ [0]
foreach v ∈ vectors
    if can_merge(merged[-1], v)
        merged[-1] ⟵ merged[-1] or v
    else
        merged.append(v)
    endif
endfor
```

Figure 2: The basic PC vector merging algorithm.

3.1.2 Vocabulary Reduction

We use three different vocabularies in our experiments. First, the "full" vocabulary, in which all 12 of our considered chord qualities are included—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). We also use a "triad" vocabulary, where 7ths are removed from each chord quality, resulting in only 4 chord qualities: major, minor, diminished, and augmented. Finally, we use a "maj-min" vocabulary, where additionally, diminished chords are represented as minor and augmented chords as major.

To handle these vocabulary reductions, we make a few small tweaks to the CPM post-processing presented in Sections 2.2 and 2.3. Namely, with either reduction, a 7th is always considered an added tone (and thus uses the threshold a). Therefore, the presence or absence of a 7th in a CP vector does not affect its ability to merge with a neighbor. Additionally, with the maj-min vocabulary, the "default" PC a perfect fifth above the root is considered a neighbor of the PC a diminished fifth above the root (for minor triads) or the PC an augmented fifth above the root (for major triads). With these added heuristics, our proposed method can recreate the full vocabulary from either reduction (e.g., by treating a diminished triad as a minor triad with a ♭5 when using the maj-min vocabulary).

3.1.3 Chord Merging

Any consecutive chords which are identical (as a result of a vocabulary reduction) are merged on input. Likewise, consecutive chords with identical root, quality, and inversion are also merged. This merges, for example, two C major triads which only differ by a chord tone alteration. Importantly, each merge only occurs in the model's input. The ground truth targets remain unchanged: the goal is therefore that our method will leave such chords unmerged in its output.

3.1.4 Evaluation Metrics

For evaluation, we measure the accuracy of an output by first splitting the input piece into windows by cutting it at each point where either the (non-reduced) ground truth label or the model's output label changes. This includes completely new chords as well as chords which differ only by, e.g., an added tone. Each of these windows is treated as a separate data point, which we deem correct if the PCs included in the (non-reduced) ground truth label match exactly the PCs included in the model's output label. Overall accuracy is then calculated as the proportion of these windows which are correct (times 100). Default accuracy is calculated similarly, only considering those windows for which the ground truth label includes only default PCs, and non-default accuracy measures performance only on those windows for which the ground truth label includes at least one non-default PC. All significance claims are evaluated using the Binomial test.

3.1.5 Model Tuning

To train the CPM, we used the Adam optimizer (Kingma & Ba, 2014) with a learning rate of .001 and weight decay of .001. We applied a learning rate scheduler which reduces the learning rate by half if the validation loss does not improve in 10 epochs, and we used early stopping if the validation loss does not improve in 20 epochs. For each version of the CPM, we performed a small grid search over the number of LSTM layers $\in \{1, 2, 3, 4, 5\}$ and the size of the first Linear layer, the LSTM layer(s), and the penultimate Linear layer, each $\in \{128, 256\}$, saving the model which achieved the lowest validation loss for each. We used a context size c of 8 for all trials, noticing in preliminary tests that validation loss decreased with larger context sizes, but with significantly diminishing returns for larger values.

Finally, given a trained model for each test condition, we performed a grid search over the thresholds $d \in \{.7, .8, .9\}$, $a \in \{.7, .8, .9\}$, and $r \in \{.5, .6, .7\}$. However, it is not trivial to choose a single best model for the thresholds. For best overall accuracy, the optimal setting may very well be to simply always output all default PCs and no others, but that is not the point of our method. On the other hand, we do not want to simply ignore the default chords by choosing the setting that maximizes the non-default accuracy, thus ignoring the much more common default case (taking the harmonic mean essentially does this, since the non-default accuracy is much lower than the default accuracy). Therefore, we use the following heuristic to choose the best thresholds. We rank each setting's default and non-default accuracies within those of all 27 experiments. Then, we choose the set of thresholds for which the maximum of these two ranks is minimized. For example, if one setting has the 6th best default accuracy and the 7th best non-default accuracy, we prefer it over a different model with the 2nd best default accuracy and the 8th best non-default accuracy.

3.2 Baseline

For a comparison to our model, we propose one rule-based baseline component for each vocabulary. The rule-based component outputs windowed binary PC vectors directly by assigning either a 0 or a 1 to each note in a chord's window. This output is processed in the same way as our proposed method's binary PC vectors, using the merging procedure described in Section 2.3. For the full vocabulary, three types of notes are assigned an output of 1:

1. Any default PC according to the chord's type (e.g., C, E, and G in a C major triad).
2. Any diminished 5th above the chord's root in a major triad with no perfect or augmented fifth in the window (e.g., a G♭ in a C major triad window with no G).
3. Any double diminished 3rd above the chord's root in a diminished triad with no minor third in the window (e.g., an E♭♭ in a C diminished triad window with no E♭).

Case 1 handles most chord tones, while cases 2 and 3 capture our encoding of augmented 6th chords (see Section 2.2.1). With the triad vocabulary, one additional case is added:

4. Any 7th above the root with no different 7 in the window (e.g., a B♭ in a C major triad, provided there is no other B in the window).

Finally, with the maj-min vocabulary, one final case is added:

5. Any diminished or augmented 5th above the root with no different 5 in the window (e.g., a G♭ in a C major triad, provided there is no other G in the window).

The above rules handle the basic chord pitches well. We add two additional cases to each to identify the most common suspensions:

- Any 6th above the root in a window with no 5th (e.g., any A in a C major triad, provided there is no G in the window).
- Any 4th above the root in a window with no 3rd (e.g., any F in a C major triad, provided there is no E in the window).

3.3 Experiment 1: Ground Truth Input

For our first experiments, we use ground truth input. This corresponds to the task of trying to induce chord tone alterations and suspensions given an expert-labeled musical score, and the results can be found in Table 1. Here, our proposed method significantly outperforms the baseline on all vocabularies ($p < .001$ for all), mainly on the default chords where the baseline tends to over-predict chord tone alterations. On the full vocabulary, our method achieves nearly perfect default accuracy while still achieving a non-default accuracy of 36.7. Still, there is clearly a difficulty in detecting non-default chord tones, which is expected since they are the much rarer case, and there is an inherent trade-off between default and non-default accuracy.

In terms of the vocabulary reductions, our method sees a significant drop in performance from the full vocabulary to the triad vocabulary ($p < .001$). However, reducing it further to maj-min results in a comparatively small decrease of about .5 worse on the default chords, although nearly 7 for non-default, an overall significant difference ($p < .001$). Figure 3 investigates the source of these decreases in terms of chord quality, for the most common chord qualities—all of those with at least 100 examples in the test set—in order of decreasing frequency. There is no immediate trend for non-default accuracy, but default accuracy exhibits more regularity. For one, usage of the full vocabulary achieves the greatest accuracy for all chord qualities (each at $p < .001$). Furthermore, the maj-min vocabulary (in contrast to the

Vocabulary	Method	Default	Non-default	Overall
Maj-min	Baseline	65.3	37.5	62.4
	This work	84.0	28.8	77.2
Triad	Baseline	67.5	37.7	64.3
	This work	84.5	35.3	78.4
Full	Baseline	80.2	42.0	76.1
	This work	96.1	36.7	89.2

Table 1: The default, non-default, and overall accuracies of our proposed method and the baseline on each vocabulary using ground truth input.

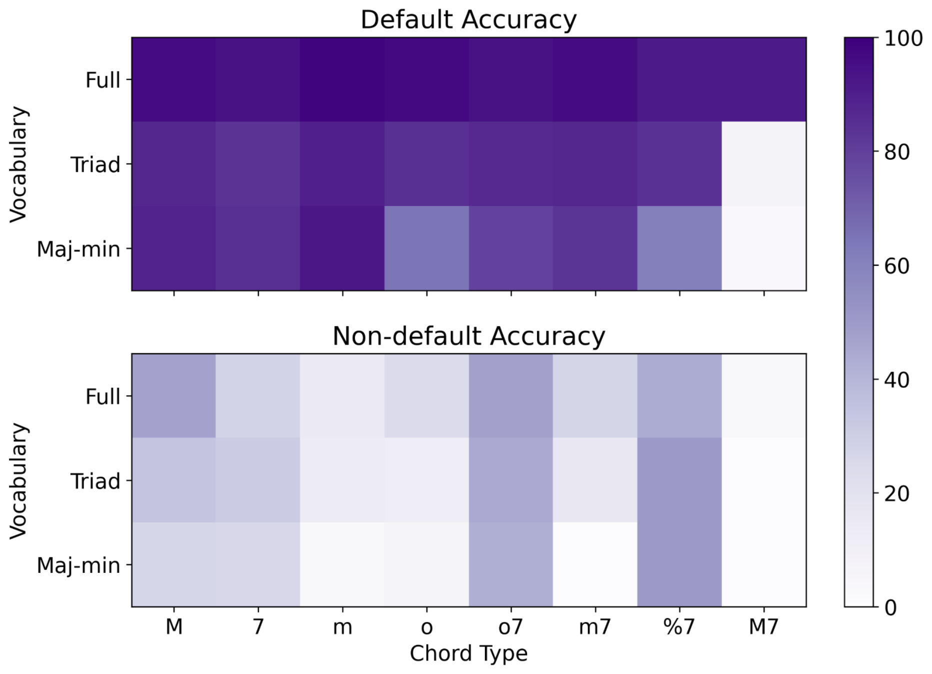

Figure 3: The default and non-default accuracy of our method on ground truth input across each vocabulary, split by chord quality (for the most common chord qualities). Chord qualities are in order of decreasing frequency.

triad vocabulary) leads to a decrease in performance specifically on those chord qualities which are no longer present: diminished, diminished 7th, and half-diminished (shown as “%7”; each at $p < .001$). That is, of course, not unexpected, but still important to note: At least when using ground truth input, reducing a chord quality out of the vocabulary reduces our method’s performance significantly on that chord quality.

Figure 4 shows our method’s output on the full vocabulary on an excerpt from Franz Schubert’s Winterreise, No. 16, measure 5 and 6. Roman numerals show the ground truth annotations in E♭ major, while green, red, and no boxes indicate correct, incorrect, and default outputs respectively. On this excerpt, our model performs well, correctly classifying the ♭3 (the C♭3) in the Italian 6th chord, as well as 64, 4, and 6 suspensions in the 2nd measure (classifying the appropriate E♭ and/or G as a chord tone each time). The only error is on the A^{o7} chord in the first bar, where our model correctly finds the B♭4 from the singing part as a chord tone but notates it as a suspension of a C rather than as an added tone. This is difficult for the model: Since there is no C present in the chord, its post-processing heuristics indicate that the B♭ should replace the C. In future work, explicitly searching for a resolution for such “suspensions” might lead to improved performance.

3.3.1 Training Strategy

To briefly investigate the effect of our training procedure (see Section 2.1.1), we train three additional CPMs, each on the full vocabulary: one without the proposed training strategy (i.e., weight factor $w = 1$), one where there is no delay before applying the weight factor (rather than having it initially ignored), and one with an even stronger reduction of w

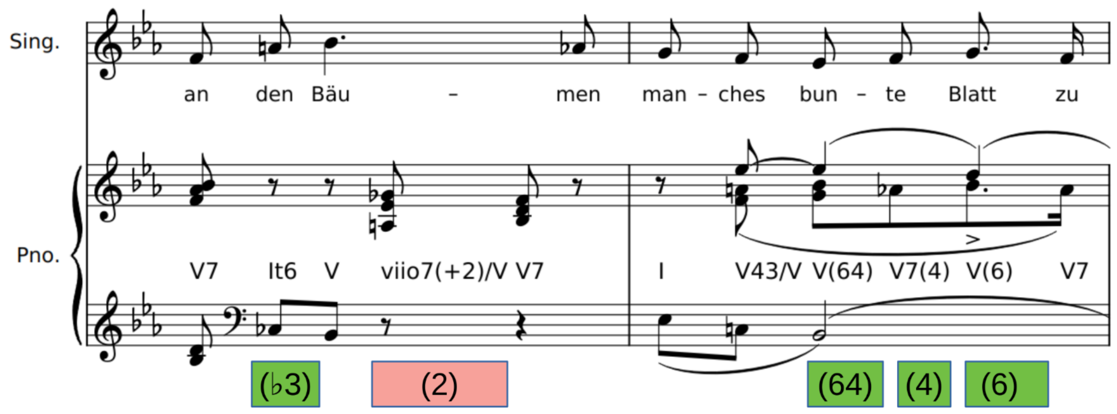
Sing.
an den Bäu – men man – ches bun – te Blatt zu
Pno.
V7 It6 V viio7(+2)/V V7
I V43/V V(64) V7(4) V(6) V7
(♭3)
(2)
(64)
(4)
(6)

Figure 4: Excerpt from Franz Schubert's Winterreise, No. 16, m. 5 and 6. Ground truth labels are given as a Roman numeral analysis in E♭ major. Our method's output (with the full vocabulary and ground truth inputs) is below the staff, where a green box indicates the correct output, a red box indicates an error, and no box is shown when the default pitches were output. Altered tones are noted in parentheses.

Strategy	Default	Non-default	Overall
No delay	94.3	29.9	86.6
w = .25	81.7	46.4	78.0
w = .5	96.3	35.3	89.4
w = 1	97.7	31.1	89.8

Table 2: The default, non-default, and overall accuracies of our proposed method using different training strategies on the full vocabulary. In our main experiments, we use w = .5.

= .25. For each model, we performed a full grid search as described in Section 3.1.5 to set its hyperparameters and thresholds. The evaluation accuracies are shown in Table 2.

There is a clear (and unsurprising) trend for larger values of w to correlate with higher default accuracy and lower non-default accuracy (all p < .001). Furthermore, our strategy of delaying the use of the weight factor improves performance significantly (p < .001). However, there is an additional factor which made us choose w = .5 as our default, even though it performs similarly to w = 1 in the best case: Many of the w = 1 models during the hyperparameter search fell into local minima that would have had it perform very poorly. Only a couple of them achieved as good results as those reported in the table. On the other hand, the w = .5 models much more consistently trained successfully, with only a few falling into local minima, and most achieving roughly similar results. It seems that too low of a value for w forces the model to concentrate too much on the non-default chords, while setting w = 1 leads to the model finding a minimum that ignores non-default chords entirely.

3.4 Experiment 2: Noisy Input

For our next experiments, we take as input the output from a chord labeling model. Specifically, we use the Chord Classification Model (CCM) proposed by McLeod and Rohrmeier (2021), trained with the hyperparameter settings used in the original paper. We trained 3 versions of the CCM: one using the full vocabulary and one after applying each vocabulary reduction procedure to its targets during training time. The results of these experiments correspond to the use of our method as a post-processing step after a chord labeling task and are presented in Table 3. Our method again significantly outperforms the baseline for all vocabularies (p < .001). However, the vocabulary reduction has a much smaller effect on our method when compared to experiment 1. The drop in accuracy from the full vocabulary to the reduced vocabularies is much smaller (4% compared to 11% with ground truth input, though still significant at p < .001), and the triad and maj-min vocabularies achieve nearly identical results (p = .81).

We investigate the vocabulary's effect on accuracy further in Figures 5 and 6. Figure 5 shows our method's default and non-default accuracy across each vocabulary, split by chord quality (just as in Figure 3). The trend is much less clear than in Figure 3. There is not as consistent of a decrease in performance across chord qualities when moving away from the full vocabulary (only major and minor triads at p < .001, with M7 at p = .002, and diminished triads and dominant

Vocabulary	Method	Default	Non-default	Overall
Maj-min	Baseline	58.5	40.2	56.5
	This work	73.5	36.6	69.0
Triad	Baseline	59.4	38.9	57.2
	This work	73.2	39.2	69.0
Full	Baseline	65.5	30.7	61.1
	This work	78.1	35.8	73.2

Table 3: The default, non-default, and overall accuracies of our proposed method and the baseline on each vocabulary using ground truth input.

7th chords at p = .01 and p = .03 respectively). The maj-min vocabulary is again worse than the triad vocabulary for diminished chords (p < .001), but to a much lesser extent.

Figure 6 offers one explanation for this observation. It shows the CCM's label accuracy for each chord quality in each vocabulary when generating the noisy input. Here, a label is considered correct when its root and chord quality are correct according to the vocabulary used. Thus, for a half-diminished chord's label to be considered correct in the full vocabulary, it must be half-diminished. However, in the triad vocabulary, it need only be diminished, and when using the maj-min vocabulary, minor is sufficient.

Analyzing Figure 6 then, each vocabulary reduction tends to bring with it an associated increase in label accuracy for the newly reduced chords. For the triad vocabulary compared to the full vocabulary, the accuracy of all 7th chords increased significantly (p < .001) except for the diminished 7th (p = .18), and for the maj-min vocabulary compared to the triad vocabulary, the diminished triad (p = .008) and the half-diminished chord (p < .001) accuracies each increased significantly, while the diminished 7th chord's accuracy did not. Intuitively, this trend is not surprising: The rarer a chord quality is, the fewer training examples the CCM will see during training to learn to detect that chord quality, and the worse it will therefore perform—this is a well-known trend in machine learning literature.

In our case, following the reduction from the full vocabulary to the triad vocabulary, our proposed chord pitches method induces many of the now missing chord qualities, even achieving an increase in default accuracy for the some of the 7th chords (p < .001 for m7, p = .01 for half-diminished). Furthermore, when considering the maj-min vocabulary, our proposed method achieves nearly identical performance to the triad vocabulary. Its purportedly worse performance on the diminished chords (as shown in Figure 3 on ground truth data) is mostly made up for in practice by the CCM's increase in accuracy on those chords (though, as stated above, the difference is still significant). Furthermore, this decrease is made up for by a proportional increase in performance and major and minor triads (p < .001 for minor, p = .15 for major).

Figure 7 shows our method's output for the first 6 measures from Edvard Grieg's Melody, Op. 47 No. 3, using the maj-min vocabulary and noisy inputs (although in this case, the CCM output all correct triads). As input to our method, the first 4 measures were all shown as a single chord (since A and Am7 both reduce to an A minor triad). Our method correctly identifies the 7th, but incorrectly merges the two chords together. Meanwhile, in the 6th measure, our method performs very well: Although it is given that the entire measure is a B minor triad, it correctly identifies all chord tones: A, G (with no G♯), and E (with no D) in the first half of the measure, and A and G (with no G♯) in the second half. Additionally, it doesn't merge the two chords, since the second already has a D. On the other hand, using the triad vocabulary, the CCM mislabels the first chord in measure 6, and with the full vocabulary, our method incorrectly merges the two chords together (including the 4 suspension).

4 Discussion

Our experiments may offer some interesting insights about the modeling of harmonic analysis and also beyond. Labeling a musical segment with a root and a chord quality is—from a model's perspective at least—a more challenging task to learn given its large vocabulary. On the other hand, in a vast majority of cases, identifying chord tones given a label is comparatively simple. Our Experiment 2 has shown a way to potentially leverage this difference to increase label accuracy on the rarer chord qualities (on which models are liable to perform worse). By making the labeling model's task easier (through a vocabulary reduction), and instead relying on our proposed method to induce the rarer chord qualities during its label post-processing, a balance that can lead to an increase in performance on those rare chords. We have shown that removing 7th chords from the vocabulary can lead to an improvement in their identification, and that the performance gap associated with diminished chords can be greatly reduced by removing them as well.

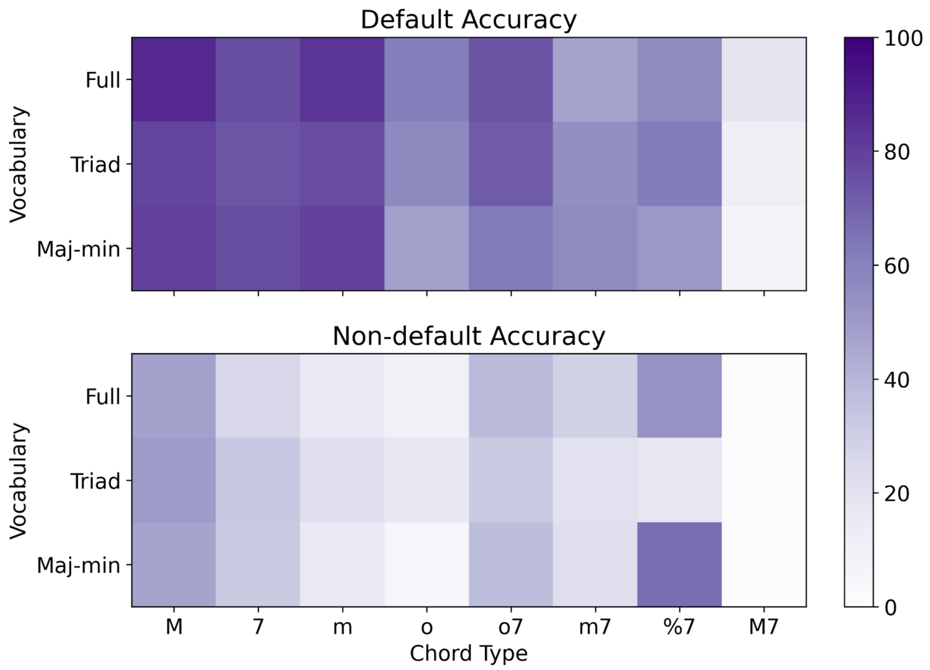

Figure 5: The default and non-default accuracy of our method on noisy input across each vocabulary, split by chord quality (for the most common chord qualities). Chord qualities are in order of decreasing frequency.

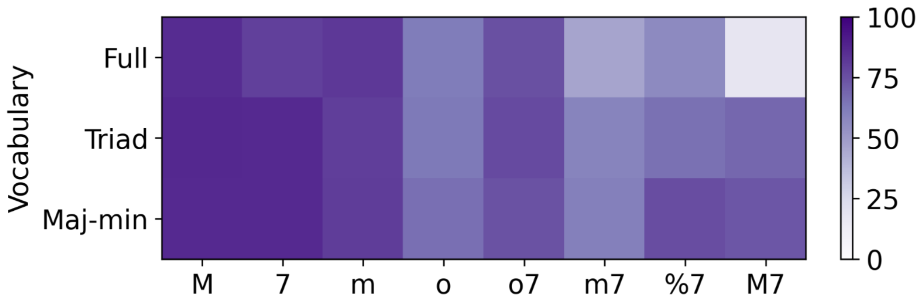

Figure 6: The CCM's label accuracy when generating the noisy input for each vocabulary. A correct label is one with the correct root pitch and chord quality. The results are split by chord quality (for the most common chord qualities) and are in order of decreasing frequency.

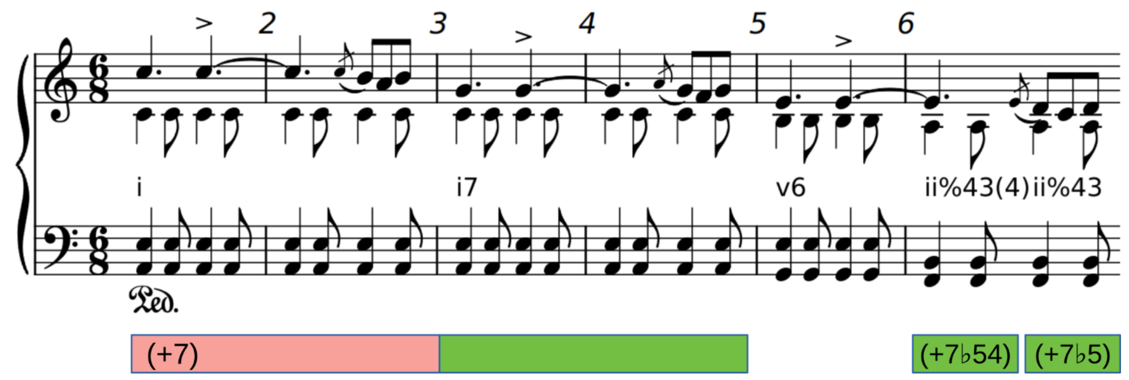

Figure 7: First 6 measures from Edvard Grieg's Melody, Op. 47 No. 3. Ground truth labels are given as a Roman numeral analysis in A minor. Our method's output (with the maj-min vocabulary and noisy inputs) is below the staff, where a green box indicates the correct output, a red box indicates an error, and no box is shown when the default pitches were output. For this excerpt, the CCM identified all correctly.

Pedagogically, this can be seen as first teaching a model to identify the most common chord qualities, and then teaching it (or, here, a 2nd model) about the rare chord qualities as extensions of those common ones. For example, a minor 7th could be modeled as a minor triad with an added minor 7th, thus implicitly imposing a two-level hierarchy between chord qualities. Our training strategy—described in Section 2.1.1 and evaluated in Experiment 1—can be seen in a similar light. Specifically, setting the initial default weight factor to 1—only lowering it to .5 once training stagnates—focuses the model on easier (i.e., more common) examples first, only emphasizing harder (i.e., rarer) examples later, once the model has learned a reasonable representation of the easier ones.

From a more general perspective, these two strategies could further be considered to be somewhat analogous to Cognitive Load Theory in cognitive psychology, an instructional theory in which complex procedures can be more easily and readily learned by storing related elements together in long-term memory (e.g. van Merriënboer & Sweller, 2005; Pass et al., 2003; Sweller, 1994). By analogy, our model first learns to recognize the "schemata" of default chords and only later to recognize non-default chords as alterations thereof relying on the simpler representation.

It should be noted that ours is not the first work to propose or use a hierarchical understanding of chord qualities (Harte, 2010; Kinnaird & McFee, 2021), nor the first to rely on something similar to train a chord labeling model. Rowe and Tzanetakis (2021), for example, use a procedure that weights each data point in an inverse proportion to its chord quality's frequency. They gradually raise the slope of this relationship throughout training, thus initially focusing the model on the most common chord qualities, and only later putting more weight on the rare types (this is explicitly done only by frequency though, not by any hierarchical relationship). Still, to our knowledge, we are the first to rely on this hierarchy explicitly with two independent models, as well as the first to incorporate unconstrained chord tone alterations and extensions directly into the hierarchical paradigm.

5 Conclusion

In this work, we have presented a method for detecting chord tones, in particular identifying suspensions and added notes, given a musical score labeled with root and chord quality information. Our method consists of three steps: (1) using a Chord Pitches Model (CPM) to label each note within a chord window as a chord tone or a non-chord tone; (2) processing this output to generate binary pitch class (PC) vectors for salami slices of the chord window; and (3) merging neighboring PC vectors together where possible. We have presented experiments showing that our method achieves strong results with ground truth and noisy labels, significantly outperforming a heuristic baseline in both cases. Its performance on noisy input indicates its suitability to be used as one step in a harmonic inference pipeline which labels an input musical score with a harmonic analysis, which we intend to do in future work.

We proposed a two-stage training procedure for the CPM where we first train it on all data points, and then force it to focus on the more rare and difficult non-default chords, and showed that this training procedure, which we relate to both Curriculum Learning (Bengio et al., 2009) and Cognitive Load Theory (Sweller, 1994), improves accuracy. Also related to Cognitive Load Theory, we proposed vocabulary reductions in which the chord labels do not express the full set of chord qualities; rather, more complicated chord qualities are induced by our proposed method. We showed that this strategy improves performance on the rare chords in some cases, in particular when used in a practical setting with noisy input.

Acknowledgments

This work was partially funded through the Swiss National Science Foundation (SNF) within the project "Distant Listening – The Development of Harmony over Three Centuries (1700–2000)" (grant no. 182811). The authors also thank Claude Latour for supporting this research through the Latour Chair in Digital Musicology at EPFL.

References

Aldwell, E., Schachter, C., & Cadwallader, A. (2018). Harmony and voice leading. Cengage Learning.

Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum learning. In International Conference on Machine Learning (ICML), (pp. 41–48).

Chen, T.-P., & Su, L. (2021). Attend to Chords: Improving Harmonic Analysis of Symbolic Music Using Transformer-Based Models. Transactions of the International Society for Music Information Retrieval, 4(1), 1–13. http://doi.org/10.5334/tismir.65

Harte, C. (2010). Towards automatic extraction of harmony information from music signals (Doctoral dissertation).

Hentschel, J., Moss, F. C., Neuwirth, M., & Rohrmeier, M. (2021). A semi-automated workflow paradigm for the distributed creation and curation of expert annotations. In Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), (pp. 262–269). Online.

Hentschel, J., Neuwirth, M., & Rohrmeier, M. (2021). The Annotated Mozart Sonatas: Score, Harmony, and Cadence. Transactions of the International Society for Music Information Retrieval, 4(1), 67–80. http://doi.org/10.5334/tismir.63

Hochreiter, S., & Schmidhuber, Jürgen. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780.

Huron, D. (2008). Sweet anticipation: Music and the psychology of expectation. MIT press.

Ju, Y., Condit-Schultz, N., Arthur, C., & Fujinaga, I. (2017). Non-chord tone identification using deep neural networks. ACM International Conference Proceeding Series, 13–16. https://doi.org/10.1145/3144749.3144753

Karystinaios, E., & Widmer, G. (2023). Roman Numeral Analysis with Graph Neural Networks: Onset-wise Predictions from Note-wise Features. In Proceedings of the International Society for Music Information Retrieval Conference (ISMIR).

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kinnaird, K. M., & McFee, B. (2021). Automatic Hierarchy Expansion for Improved Structure and Chord Evaluation. Transactions of the International Society for Music Information Retrieval, 4(1), 81–92. http://doi.org/10.5334/tismir.71

Korzeniowski, F., & Widmer, G. (2018). Improved chord recognition by combining duration and harmonic language models. In Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), (pp. 404-411).

Levine, M. (2011). The jazz theory book. " O'Reilly Media, Inc.

McLeod, A., & Rohrmeier, M. (2021). A modular system for the harmonic analysis of musical scores using a large vocabulary. In Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), (pp. 90–97).

Micchi, G., Kosta, K., Medeot, G., & Chanquion, P. (2021). A deep learning method for enforcing coherence in Automatic Chord Recognition. In Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), (pp. 443-451).

van Merriënboer, J. J. G., & Sweller, J. (2005). Cognitive load theory and complex learning: Recent developments and future directions. Educational psychology review, 17(2), 147-177. https://doi.org/10.1007/s10648-005-3951-0

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In International Conference on Machine Learning (ICML).

Nápoles López, N., Gotham, M., & Fujinaga, I. (2021). AugmentedNet: A Roman Numeral Analysis Network with Synthetic Training Examples and Additional Tonal Tasks. In Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), (pp. 404-411).

Neuwirth, M., Harasim, D., Moss, F. C., & Rohrmeier, M. (2018). The Annotated Beethoven Corpus (ABC): A dataset of harmonic analyses of all Beethoven string quartets. Frontiers in Digital Humanities, 16.

Paas, F., Renkl, A., & Sweller, J. (2003). Cognitive load theory and instructional design: Recent developments. Educational psychologist, 38(1), 1-4. https://doi.org/10.1207/S15326985EP3801_1

Rowe, L. O., & Tzanetakis, G. (2021). Curriculum Learning for Imbalanced Classification in Large Vocabulary Automatic Chord Recognition. In Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), (pp. 586-593).

Sweller, J. (1994). Cognitive load theory, learning difficulty, and instructional design. Learning and instruction, 4(4), 295-312.

Temperley, D., & Sleator, D. (1999). Modeling meter and harmony: A preference-rule approach. Computer Music Journal, 23(1), 10-27.

Zhou, X., & Lerch, A. (2015). Chord detection using deep learning. In Proceedings of the International Society for Music Information Retrieval Conference (ISMIR) (pp. 52-58).