LARGE-VOCABULARY CHORD TRANSCRIPTION VIA CHORD STRUCTURE DECOMPOSITION

Junyan Jiang[1,2,3] **Ke Chen**[1,2] **Wei Li**[1] **Gus Xia**[2]

[1] Computer Science Department, Fudan University [2] Music X Lab, NYU Shanghai
[3] Machine Learning Department, Carnegie Mellon University
[1]{jiangjy14, kchen15, weili-fudan}@fudan.edu.cn, [2]gxia@nyu.edu

ABSTRACT

While audio chord recognition systems have acquired considerable accuracy on small vocabularies (e.g., major/minor chords), the large-vocabulary chord recognition problem still remains unsolved. This problem hinders the practical usages of audio recognition systems. The difficulty mainly lies in the intrinsic long-tail distribution of chord qualities, and most chord qualities have too few samples for model training.

In this paper, we propose a new model for audio chord recognition under a huge chord vocabulary. The core concept is to decompose any chord label into a set of musically meaningful components (e.g., triad, bass, seventh), each with a much smaller vocabulary compared to the size of the overall chord vocabulary. A multitask classifier is then trained to recognize all the components given the audio feature, and then labels of individual components are reassembled to form the final chord label. Experiments show that the proposed system not only achieves state-of-the-art results on traditional evaluation metrics but also performs well on a large vocabulary.

1. INTRODUCTION

Large-vocabulary chord transcription is a difficult task, as the number of chord qualities is large, and the distribution of training chord classes is extremely biased. For example, the Billboard dataset [2], a human-annotated dataset, contains 230 different chord qualities, or equivalently, 2,749 distinct chord classes [1]. While the first 10% chord qualities cover 93.86% of the data, the last 50% chord qualities only cover 0.35% of the data altogether [2]. Such a long-tailed chord distribution makes it extremely hard to model rare chord qualities.

To bypass the problem, former systems typically adopt two kinds of strategies: chord quality simplification and

[1] We here assume that each chord quality can be combined with all possible 12 roots except for the `N` chord.

[2] In calculation, the chord quality counts are weighted by their durations.

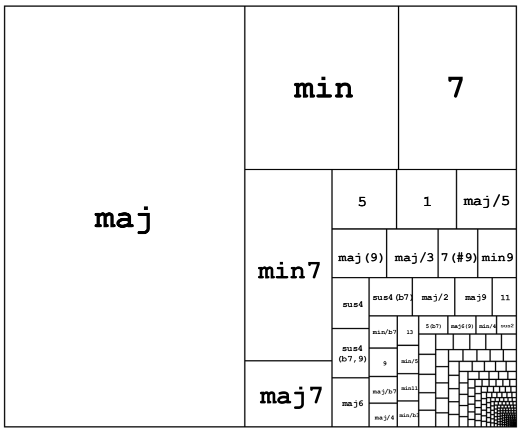

Figure 1. A visualization of chord quality distribution in our collected chord dataset. Each chord quality is denoted by a block whose size is proportional to its number of appearances. Labels for small blocks are omitted.

chord structure representation. The first approach maps complex chord qualities into simpler ones (e.g., to map `C:maj7`, `C:maj9`, `C:maj11` etc. all to `C:maj`), and therefore suppress the number of different chord classes. Then, a classifier is trained based on the simplified chord vocabulary. The most common mapped chord vocabulary is the major/minor vocabulary, which consists of merely 12 major chords, 12 minor chords, and a non-chord label. However, this is often an over-simplification, especially for some pop and jazz chords with richer expressions than major/minor chords do.

The second approach, instead, turns chord label classification into a structured estimation problem, and this study belongs to this category. To achieve this, a unified structure representation of chord symbols is required. Chromagram-like representations are often adopted for chord recognition as it is a direct reflection of acoustic features and often led to a good performance in practice [16, 20].

However, Chromagram-like representations miss some of the musical semantics that are important references for human transcribers. One example features two chords `E:min7/b3` and `G:maj6` which are very similar in the chromagram as they share the same chord notes {G, B, D, E} and bass G. However, they are quite different in musical semantics as one is a major chord and the other is minor. Moreover, in audio chord recognition, some chord degrees are more important than others. For example, a wrong triad is regarded as a more severe problem than a wrong ninth or other extensions. Such intuition is not directly reflected in a chromagram-like representation of a chord.

We believe that structured representation is the key to large-vocabulary chord recognition, yet a more musically meaningful representation is needed. In this paper, we propose a new representation method by chord structure decomposition. We also present a multitask classification model comprised of a Convolutional Recurrent Neural Network (CRNN) and a Conditional Random Field (CRF), that utilize such representation into the audio chord recognition system.

2. RELATED WORK

Audio chord recognition is an important task for content-based music information retrieval and has been actively studied for 20 years [22]. Audio chord recognition systems typically follow a two-stage process: feature extraction and chord sequence decoding. For traditional models, chromagram [19, 22, 23] is the most widely used feature for chord recognition as it reflects the acoustic properties of a chord. There are also other attempts for handcrafted features, such as the tonnetz feature [12]. For the sequence decoding model, template matching and Hidden Markov Models (HMMs) [24, 27] are adopted to decode the most likely chord sequence. Considering the relationship of chord labels and other musical concepts (e.g., beats, keys), complex HMMs and other Dynamic Bayesian Networks (DBNs) are also used for joint decoding [19, 23].

With the development of deep learning, recent studies begin to focus more on features extracted by neural networks, such as feed-forward Deep Neural Networks (DNNs) [16] and Convolutional Neural Networks (CNNs) [10, 17]. For the sequence decoding phase, beam search and the Viterbi algorithm are widely used [1, 18, 28]. While early papers in this area adopt a small and fixed vocabulary like the major/minor vocabulary [1, 16, 17], recent work focuses more and more on larger vocabularies [7, 8, 20].

Strategies for structured representation of chords has been widely discussed in the field of music generation and analysis [4, 9], but not all of them are helpful in the context of chord recognition problem. In audio chord recognition, the HPA system [23] uses two latent variables, the root-position form and the bass note, to model a chord in a Hidden Markov Model. McFee and Bello [20] adopt a 36-D vector to represent a chord, denoting a binary encoding of its root, bass and chord degrees respectively. Then, a plain 170-class classifier is adopted to decode chord symbols from the features.

3. PROPOSED METHOD

3.1 Chord Structure Representation

Most chord symbols can be regarded as a collection of musically meaningful components: root, bass (for possible inversions), a triad type, and a set of extra notes,

Chord	Root	Triad	Bass	7th	9th	11th	13th
`G:maj`	G	maj	G	N	N	N	N
`G:maj7`	G	maj	G	7	N	N	N
`G:7(b9)`	G	maj	G	b7	b9	N	N
`G:min7/b3`	G	min	Bb	b7	N	N	N
`B:hdim7`	B	dim	B	b7	N	N	N
`A:sus4(b7)`	A	sus4	A	b7	N	N	N
`C:9(13)`	C	maj	C	b7	9	N	13
`N`	N	N	N	N	N	N	N

Table 1. Some examples of chord structure decomposition.

which usually include certain 7th, 9th, 11th and/or 13th degrees above the root. Different combinations of these components form the large chord vocabulary that musicians use. However, the possible choices for each component are pretty limited. For example, the 7th degree of a chord, if exists, is most likely to be one among `7`, `b7` and `bb7` (in `dim7` chords). This property is very helpful under the context of audio chord recognition, as the number of classes for chord label classification can be greatly reduced if we break down the chords into these components and perform classification on each component instead.

We now formally introduce our chord representation method. We first define the chord components and their possible labels:

$$\begin{aligned}
\text{Root} &\leftarrow \{\text{N}, \text{C}, \text{C\#/Db}, \text{D}, ..., \text{B}\} \\
\text{Triad} &\leftarrow \{\text{N}, \text{maj}, \text{min}, \text{sus4}, \text{sus2}, \text{dim}, \text{aug}\} \\
\text{Bass} &\leftarrow \{\text{N}, \text{C}, \text{C\#/Db}, \text{D}, ..., \text{B}\} \\
\text{Seventh} &\leftarrow \{\text{N}, 7, \text{b7}, \text{bb7}\} \\
\text{Ninth} &\leftarrow \{\text{N}, 9, \#9, \text{b9}\} \\
\text{Eleventh} &\leftarrow \{\text{N}, 11, \#11\} \\
\text{Thirteenth} &\leftarrow \{\text{N}, 13, \text{b13}\}
\end{aligned}$$

Notice that we ignore incomplete triads like `C:(1,5)` and `C:(1)`, so these qualities will not appear in the triad category. To make other chord degrees (e.g., the major 6th in `maj6` and `min6`) compatible with the representation, we here adopt octave equivalent versions of these degrees that match the form of the chord extensions listed above (e.g., 6th to 13th).

By enumerating distinct labels for these components, we can map the component labels into numeral values. The enumerated values are all indexed from 1. We now define the chord vector, the encoded form of chord used by our models. The chord vector $\mathbf{c} = (c_1, ..., c_6)$ of a chord C is defined as a 6-dimension vector. For non-chord class `N`, all $c_i = 1$. For any other chord label C, we have:

$$\begin{aligned}
c_1 &= (\text{TriadIndex}(C) - 1) \cdot 12 + \text{RootIndex}(C) + 1 \\
c_2 &= \text{BassIndex}(C) \\
c_3 &= \text{SeventhIndex}(C) \\
c_4 &= \text{NinthIndex}(C) \\
c_5 &= \text{EleventhIndex}(C) \\
c_6 &= \text{ThirteenthIndex}(C)
\end{aligned}$$

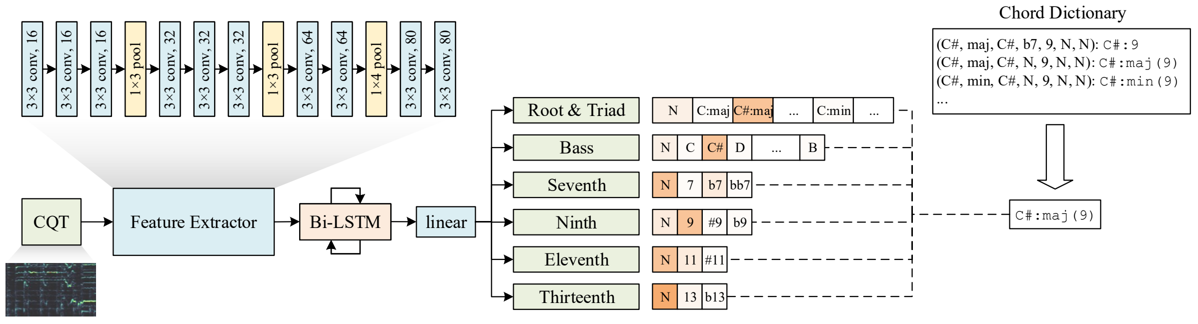

Figure 2. An overview of the system.

3.2 Model Architecture

The proposed chord recognition system consists of two parts, the feature processor and the decoding model. The feature processor transforms the low-level audio features to the frame-wise activations for each chord component value. The decoding model then decodes the final chord sequence given the activations. An overview of the system is shown in Figure 2.

3.2.1 Feature Processor

We first apply the Convolutional Recurrent Neural Network (CRNN), a powerful architecture for audio feature processing [3, 6, 20], to the input spectrogram. The audio feature is first fed into convolutional layers. After each convolutional layer, we perform batch normalization and then a rectified linear function to introduce non-linearity. Each convolutional layer adopts a 3×3 kernel size with 1×1 zero padding to the input, except for the last two layers where no padding is applied on the frequency axis, in order to reduce the feature size.

After that, we apply a Bi-directional Long Short-Term Memory (Bi-LSTM) layer to introduce temporal context for chord recognition. Each direction has a hidden size of 96. Then, for each frame t, a linear unit transforms the hidden states for both directions into six vectors $\mathbf{s}^{(t,1)}...\mathbf{s}^{(t,6)}$. The softmax function is performed on each vector to get the activation $\mathbf{a}^{(t,1)}...\mathbf{a}^{(t,6)}$ for each component of the chord vector:

$$a_k^{(t,i)} = \frac{\exp(s_k^{(t,i)})}{\sum_{k'=1}^{N_i} \exp(s_{k'}^{(t,i)})} \ \forall i = 1...6, \forall k = 1...N_i \quad (1)$$

Here, N_i denotes the vocabulary size of the i-th component category. The loss of the neural network given the ground-truth chord sequence $\mathbf{C} = \{C^{(1)}, ..., C^{(T)}\}$ is defined as weighted cross-entropy loss:

$$L = -\sum_{t=1}^{T} \sum_{i=1}^{6} \sum_{k=1}^{N_i} w_k^{(i)} \mathbb{I}[k = c_i^{(t)}] \log(a_k^{(t,i)}) \quad (2)$$

Here, $\mathbb{I}[\cdot]$ is the indicator function, and $c_i^{(t)}$ denotes the i-th component of the chord vector of $C^{(t)}$. $w_k^{(i)}$ is the class weight factor for index k of the i-th component. We will further explain it in section 3.3.

3.2.2 Chord Decoding Model

To decode the final chord sequence from the activation vectors $\mathbf{a}$, an intuitive way is to pick the class with the largest activation values for each component and each frame. However, this approach tends to produce excessive chord changes as there is no transition penalty between two frames. Also, it will be hard if we want to control the output chord vocabulary V. Therefore, we propose a decoding model by a linear Conditional Random Field (CRF).

The linear CRF takes the following form:

$$P(\mathbf{C} \mid \mathbf{F}) \propto \phi(C^{(1)}, \mathbf{F}) \prod_{t=2}^{T} \phi(C^{(t)}, \mathbf{F}) \psi(C^{(t-1)}, C^{(t)}) \quad (3)$$

Here, $\mathbf{F}$ is the audio feature of the whole song and $\mathbf{C} = \{C^{(1)}, ..., C^{(T)}\}$ is the target chord sequence with each $C^{(i)} \in V$. The observation potential function ϕ takes the form:

$$\phi(C^{(t)}, \mathbf{F}) = \exp \sum_{i=1}^{6} \sum_{k=1}^{N_i} \mathbb{I}[k = c_i^{(t)}] \log a_k^{(t,i)} \quad (4)$$

where $\mathbf{a}^{(t,i)}$ are the activation vectors given by the Convolutional Recurrent Neural Network (CRNN) and $\mathbf{c}^{(t)}$ is the encoded chord vector for $C^{(t)}$.

The transition potential function ψ takes the form:

$$\psi(C^{(t-1)}, C^{(t)}) = \exp\left(-d \cdot \mathbb{I}[C^{(t-1)} \neq C^{(t)}]\right) \quad (5)$$

where $d = 30$ is a hyper-parameter that controls the degree of transition penalty. A larger d penalize more on chord transition and vice versa.

3.3 Class Re-weighting

Although we do not directly perform classification on distinct chord labels, the problem of class imbalance still persists during model training. For example, some chord extensions are very infrequent in the training set, leading to

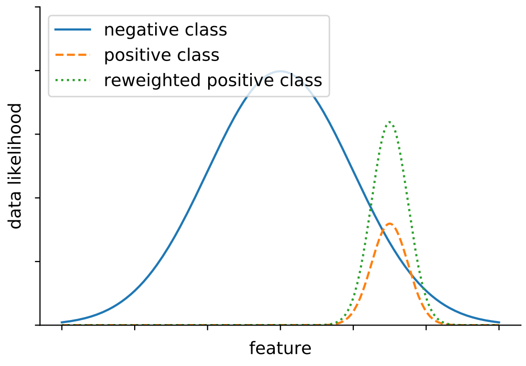

Figure 3. An illustration of why ambiguous class labeling causes problems. In this example, the class boundaries of the positive class and the negative class cross each other. Under the maximal likelihood principle, the accuracy of the positive class (with smaller prior than the negative class) is greatly harmed as its data likelihood is mostly covered by the negative class. Class re-weighting alleviates this problem to some extent.

excessive negative samples and insufficient positive samples for training. Generally, imbalanced training samples make the training process biased as the model tends to focus more on optimizing classes with more samples.

We here stress another issue that potentially aggravates this bias in learning-based automatic chord recognition systems. Audio chord annotation is an ambiguous problem even for human experts [11]. This means that the "ground truth" chord label of a fixed audio piece may contain uncertainty if annotated by different experts [15]. This may cause class boundaries for different labels to cross each other. In this case, we inevitably suffer from accuracy loss for the ambiguity between classes, and classes with small priors often suffer most (see Figure 3 for an example).

To make up for the bias, we introduce the class re-weighting strategy by introducing a class-wise weight factor $w_k^{(i)}$ for each possible chord component value in equation (2). By the re-weighting term, we want classes with fewer training samples to gain larger weights. We adopt the following weight term:

$$w_k^{(i)} = \min\left\{\left(\frac{n_k^{(i)}}{\max_{k'} n_{k'}^{(i)}}\right)^{-\gamma}, w_{max}\right\} \tag{6}$$

Here, $n_k^{(i)}$ denotes the number of training samples for class k for the i-th component. γ and w_{max} are hyper-parameters, where $0 \leq \gamma \leq 1$ is the balance factor and $w_{max} \geq 1$ is the clamping value. A smaller γ will result in a more balanced weights and vice versa.

4. EXPERIMENTS

As previous chord recognition systems adopt smaller chord vocabularies compared to us, it is hard to make a direct comparison between different systems. Therefore, we divide the experiment results into two parts.

In the first part, we will compare the performance of our system on traditional chord evaluation metrics, most of which perform evaluations on simplified chord vocabularies. The evaluation metrics are calculated by the python package `mir_eval` [26]. In the second part, we will directly evaluate the system performance on larger chord vocabularies.

4.1 Datasets

We use 1217 songs from Isophonics, Billboard and MARL collections collected by Humphrey and Bello [11, 20] to form the dataset. To make a fair comparison, we adopt the 5-fold cross-validation with the same train/validation/test splits (60% for training, 20% for validation and 20% for testing) as in [20].

4.2 Pre-processing

We extract the Constant-Q Transform (CQT) spectrogram from the audio by the librosa [21] package with a sample rate of 22050 and a hop length of 512. We use the pitch range from midi note C1 (inclusive) to C7 (exclusive) with 36 filter banks per octave (252 CQT filter banks in total).

Data augmentation is performed by the pitch-shifting operation (from -5 semitones to +6 semitones) on the training set. Augmented features are directly calculated by shifting CQT spectrograms. The annotated chord labels are shifted accordingly.

4.3 Model Training

We use the Adam optimizer [14] to optimize the neural network parameters with a scheduled learning rate adjustment. To begin with, we use a learning rate 1e-3. If the validation loss does not improve in 5 epochs, we decrease the learning rate by 90%. We stop training after the learning rate drops below 1e-6.

In each epoch, a 1000-frame segment (approximately 23.2 seconds) is randomly selected from each song. 24 such segments form a mini-batch for gradient calculation.

4.4 Comparative Results

We first evaluate the performance of our system under traditional metrics proposed by [25]. For different metrics, the model outputs and ground truth chords are first mapped to some small vocabularies by removing extra notes and/or inversions. The scores are calculated by their matching percentages. The adopted metrics are: root only, root and thirds, major/minor chords, basic triads, sevenths, tetrads and MIREX (at least three correct notes between reference and estimated chords). All of these scoring methods ignore extended chord degrees (e.g., ninth, eleventh and thirteenth).

To evaluate the effect of our chord representation versus a plain classifier, we also perform evaluations on a modified version of the model, which contains a linear layer and a *flat* softmax output layer for all possible combinations of roots and chord types present in the dataset. The model is denoted by "Flat" in Figure 4. Other compared models are (1) Ours: the proposed model without re-weighting; (2) Ours+R: the proposed model trained with re-weighting factors ($\gamma = 0.5, w_{max} = 10$); (3) ACE18: an early design

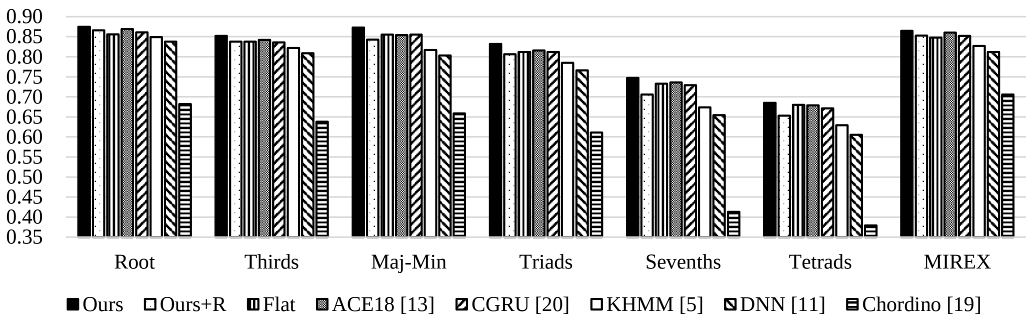

Figure 4. Comparison of median weighted recall scores.

of our model submitted to the MIREX 2018 contest [13] trained on the full vocabulary without beat alignment post-processing; (4) CGRU: A Convolutional Gated Recurrent Units (CGRU) model by [20]; (5) KHMM: A k-stream HMM by [5]; (6) DNN: A deep convolutional network by [11]; (7) Chordino: the classic baseline with template matching and HMM by [19].

In the results, Our system outperforms the baseline systems in all metrics, indicating that our system has a good performance on simplified chord vocabularies, even if the system is not trained for these vocabularies on purpose.

We can also see that the model with class re-weighting actually does not outperform the model without class re-weighting. The main reason is that class re-weighting assumes a more balanced class distribution, which is not the case of our test dataset. We will show in section 4.5.1 that class re-weighting actually has a better performance if we penalize the misclassification error of different classes equally, regardless of their frequency in the test dataset.

4.5 Performance on Large Chord Vocabulary

4.5.1 Evaluation on Common Chord Labels

To evaluate the chord recognition system on a large chord vocabulary, we first evaluate the system performance on common chord labels. We construct the target chord vocabulary V that includes the following chord qualities:

- Basic triads: `maj`, `min`, `aug`, `dim`
- Inverted triads: `maj/3`, `maj/5`, `min/b3`, `min/5`
- Seventh chords: `maj7`, `7`, `min7`, `dim7`, `hdim7`
- Extended chords: `maj9`, `9`, `min9`, `11`, `13`
- Suspended chords: `sus4`, `sus2`, `sus4(b7)`
- Slash chords: `maj/2`, `maj/b7`, `min/2`, `min/b7`
- Non-chord class: `N`

All chord qualities except `N` can be applied to 12 roots, resulting in 301 distinct chord classes. Under such a constraint, the model will not output other chord classes. To define the evaluation metrics, we use D to denote all pairs of estimated chords and reference chords (in vocabulary V) over the test dataset on the same frame. The per-class accuracy $\text{acc}_{\text{chord}}(C)$ for chord class $C \in V$ can be defined as:

$$\text{acc}_{\text{chord}}(C) = \frac{\sum_{(C_{\text{est}},C_{\text{ref}})\in D} \mathbb{I}[C_{\text{est}} = C]\mathbb{I}[C_{\text{ref}} = C]}{\sum_{(C_{\text{est}},C_{\text{ref}})\in D} \mathbb{I}[C_{\text{ref}} = C]} \quad (7)$$

Model	Frame-wise Acc.	Class-wise Acc.
No Re-weighting	**0.7719**	0.3475
(0.3,10.0)	0.7609	0.3745
(0.5,10.0)	0.7459	**0.4022**
(0.7,20.0)	0.7146	0.3738
(1.0,20.0)	0.6577	0.3832

Table 2. Mean frame-wise accuracy and mean class-wise accuracy for the proposed model with different class re-weighting factors (γ, w_{max}) for training. The re-weighting factors are shown in the model column.

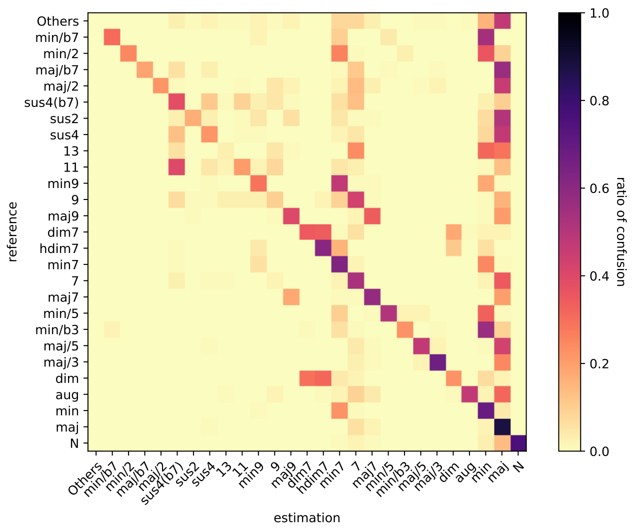

Figure 5. The quality confusion matrix of model trained with re-weighting factors $\gamma = 0.5, w_{max} = 10$.

We believe that a good chord recognition system should have high frame-wise accuracy, as well as a low bias among different chord classes. For example, a system with high bias may recognize all samples of one chord class C into another, resulting in a low score for $\text{acc}_{\text{chord}}(C)$. Therefore, we want the average of $\text{acc}_{\text{chord}}(C)$ over all chord classes as high as possible in the class-wise measurement. Formally, we define the mean frame- and class-wise

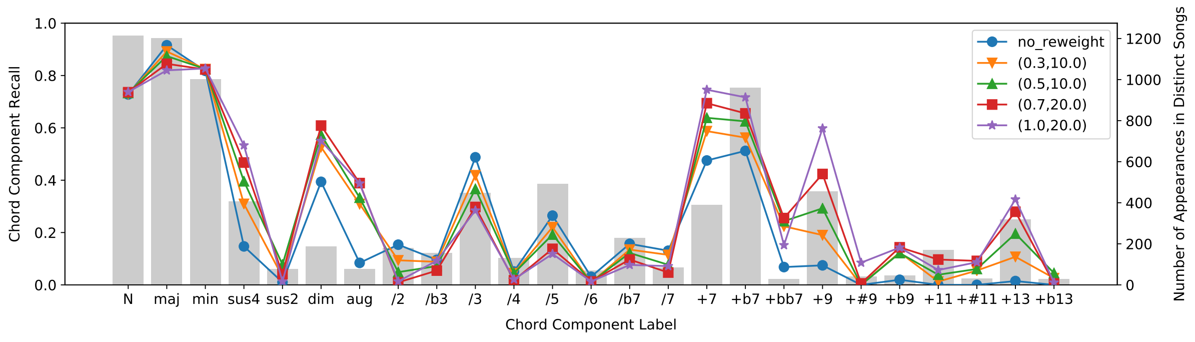

Figure 6. Recall values for different chord components on test sets (denoted by lines) and song-level component counts in the whole dataset (denoted by bars). Different lines denote the models trained with different re-weighting factors ((γ, w_{max})).

accuracy as:

$$\mathrm{acc}_{\mathrm{frame}} = \frac{1}{|D|} \sum_{(C_{\mathrm{est}}, C_{\mathrm{ref}}) \in D} \mathbb{I}[C_{\mathrm{est}} = C_{\mathrm{ref}}] \quad (8)$$

$$\mathrm{acc}_{\mathrm{class}} = \frac{1}{|V|} \sum_{C \in V} \mathrm{acc}_{\mathrm{chord}}(C) \quad (9)$$

We evaluate our system with different class re-weighting parameters under these two metrics. The results are shown in Table 2. From the results, we can see that all models with class re-weighting have a lower mean frame-wise accuracy and a higher mean class-wise accuracy compared to the model without class re-weighting. This indicates that class re-weighting alleviates the class bias problem to some extent while leading to a trade-off that the frame-wise accuracy is harmed.

We also show the within-root quality confusion matrix (i.e., quality confusion matrix when the estimated root is correct) of the model trained with re-weighting factors ($\gamma = 0.5, w_{max} = 10$) in Figure 5 for error analysis. Two major trends can be observed. First, most quality errors are from mapping complex chord qualities to simpler ones. For example, `maj9`, `9` and `min9` are mapped to `maj7`, `7` and `min7`; tetrads are mapped to triads. Second, certain extended chord qualities like `13` and `11` are hard to be detected. Despite of the label bias, we also suspect that annotated extended qualities may omit some degrees in practice (e.g., a missing eleventh in a `13` chord) but the omission is not always annotated in the test set. The misclassification of `11` chords to `sus4(b7)` is also reasonable as a `11` chord without a third and a ninth has the same pitch classes as a `sus4(b7)` chord.

4.5.2 *Evaluation on Chord Components*

To further evaluate the system's performance on a large chord vocabulary, we adopt a full chord vocabulary (i.e., a chord vocabulary containing all chord qualities present in the dataset) for the decoding process and evaluate the class-wise accuracy for each chord component. For the root and bass category, we are interested in if their relationship (chord inversion) is correctly detected. The slash notation is defined as the interval between bass and root. For example, In the chord quality `maj/5`, the fifth of the chord is the bass note. Accuracy on slash notations reflects the model's ability to detect chord inversions.

Also, we want to evaluate how class re-weighting affects the accuracy of rare chord component labels. Therefore, we experiment with different hyper-parameters for class re-weighting and make a comparison as shown in Figure 6.

In Figure 6, it is clear to show that the model with no re-weighting declines to predict certain extensions including `11` and `13`. When class re-weighting is adopted, we can observe an accuracy boost for these components. However, the trade-off of class re-weighting is the fact that it harms the accuracy of some other classes as well as providing false positive predicts.

Also, it can be observed that the accuracy for a specific component class has a positive correlation with the number of its appearances in the dataset. Some label components surely have too few samples for the model to learn the correct acoustic and semantic properties, resulting in a low accuracy even if class re-weighting is performed.

5. CONCLUSION

In this paper, we propose a method to perform audio chord recognition on a large chord vocabulary. The core idea of the model is to recognize each chord component instead of the whole chord label itself, and each chord component has a much smaller vocabulary that is easier for the model to handle. We show by experiment that the model acquires state-of-the-art performance on traditional chord evaluation metrics. Also, we demonstrate its ability to detect rare chord component values.

However, the model still has several unsolved issues. First, the model still has strong class bias even if we adopt the re-weighting strategy, and the model performance on some rarest chord component values are still not satisfactory. Second, the proposed representation is not yet perfect. More theoretical analysis and experiments are required to design the best chord representation method for the audio chord recognition task.

We also stress the class ambiguity issue with regard to the datasets. Although a class-balanced dataset is nearly unfeasible in the realm of audio chord recognition, we would like to call for datasets with more precise annotations as well as a more thorough analysis of currently available datasets.

6. REFERENCES

[1] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Audio chord recognition with recurrent neural networks. In *ISMIR*, pages 335–340. Citeseer, 2013.

[2] John Ashley Burgoyne, Jonathan Wild, and Ichiro Fujinaga. An expert ground truth set for audio chord recognition and music analysis. In *ISMIR*, volume 11, pages 633–638, 2011.

[3] Emre Cakır, Giambattista Parascandolo, Toni Heittola, Heikki Huttunen, and Tuomas Virtanen. Convolutional recurrent neural networks for polyphonic sound event detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(6):1291–1303, 2017.

[4] Emilios Cambouropoulos, Maximos A Kaliakatsos-Papakostas, and Costas Tsougras. An idiom-independent representation of chords for computational music analysis and generation. In *ICMC*, 2014.

[5] Taemin Cho. *Improved techniques for automatic chord recognition from music audio signals*. PhD thesis, New York University, 2014.

[6] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. Convolutional recurrent neural networks for music classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2392–2396. IEEE, 2017.

[7] Jun-qi Deng and Yu-Kwong Kwok. A hybrid gaussian-hmm-deep learning approach for automatic chord estimation with very large vocabulary. In *ISMIR*, pages 812–818, 2016.

[8] Jun-qi Deng and Yu-Kwong Kwok. Large vocabulary automatic chord estimation with an even chance training scheme. In *ISMIR*, pages 531–536, 2017.

[9] Christopher Harte, Mark B Sandler, Samer A Abdallah, and Emilia Gómez. Symbolic representation of musical chords: A proposed syntax for text annotations. In *ISMIR*, volume 5, pages 66–71, 2005.

[10] Eric J Humphrey and Juan P Bello. Rethinking automatic chord recognition with convolutional neural networks. In *2012 11th International Conference on Machine Learning and Applications*, volume 2, pages 357–362. IEEE, 2012.

[11] Eric J Humphrey and Juan Pablo Bello. Four timely insights on automatic chord estimation. In *ISMIR*, pages 673–679, 2015.

[12] Eric J Humphrey, Taemin Cho, and Juan P Bello. Learning a robust tonnetz-space transform for automatic chord recognition. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 453–456. IEEE, 2012.

[13] Junyan Jiang, Ke Chen, Wei Li, and Guangyu Xia. Mirex 2018 submission: A structural chord representation for automatic large-vocabulary chord transcription. *MIREX evaluation results*, 2018.

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[15] HV Koops, WB de Haas, J Bransen, and A Volk. Chord label personalization through deep learning of integrated harmonic interval-based representations. In *Proc. of the Int. Workshop on Deep Learning and Music. Anchorage, US.*, 2017.

[16] Filip Korzeniowski and Gerhard Widmer. Feature learning for chord recognition: The deep chroma extractor. *arXiv preprint arXiv:1612.05065*, 2016.

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

[18] Filip Korzeniowski and Gerhard Widmer. Improved chord recognition by combining duration and harmonic language models. In *ISMIR*, pages 10–17, 2018.

[19] Matthias Mauch and Simon Dixon. Approximate note transcription for the improved identification of difficult chords. In *ISMIR*, pages 135–140, 2010.

[20] Brian McFee and Juan Pablo Bello. Structured training for large-vocabulary chord recognition. In *ISMIR*, pages 188–194, 2017.

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

[22] Matt McVicar, Raúl Santos-Rodríguez, Yizhao Ni, and Tijl De Bie. Automatic chord estimation from audio: A review of the state of the art. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 22(2):556–575, 2014.

[23] Yizhao Ni, Matt McVicar, Raul Santos-Rodriguez, and Tijl De Bie. An end-to-end machine learning system for harmonic analysis of music. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(6):1771–1783, 2012.

[24] Hélene Papadopoulos and Geoffroy Peeters. Large-scale study of chord estimation algorithms based on chroma representation and hmm. In *2007 International Workshop on Content-Based Multimedia Indexing*, pages 53–60. IEEE, 2007.

[25] Johan Pauwels and Geoffroy Peeters. Evaluating automatically estimated chord sequences. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 749–753. IEEE, 2013.

[26] Colin Raffel, Brian McFee, Eric J Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, Daniel PW Ellis, and C Colin Raffel. mir_eval: A transparent implementation of common mir metrics. In *ISMIR*. Citeseer, 2014.

[27] Yushi Ueda, Yuki Uchiyama, Takuya Nishimoto, Nobutaka Ono, and Shigeki Sagayama. Hmm-based approach for automatic chord detection using refined acoustic features. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5518–5521. IEEE, 2010.

[28] Yiming Wu and Wei Li. Music chord recognition based on midi-trained deep feature and blstm-crf hybird decoding. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 376–380. IEEE, 2018.