STRUCTURED TRAINING FOR LARGE-VOCABULARY CHORD RECOGNITION

Brian McFee[1,2]
[1]Center for Data Science
New York University
brian.mcfee@nyu.edu

Juan Pablo Bello[2]
[2]Music and Audio Research Laboratory
New York University
jpbello@nyu.edu

ABSTRACT

Automatic chord recognition systems operating in the large-vocabulary regime must overcome data scarcity: certain classes occur much less frequently than others, and this presents a significant challenge when estimating model parameters. While most systems model the chord recognition task as a (multi-class) classification problem, few attempts have been made to directly exploit the intrinsic structural similarities between chord classes.

In this work, we develop a deep convolutional-recurrent model for automatic chord recognition over a vocabulary of 170 classes. To exploit structural relationships between chord classes, the model is trained to produce both the time-varying chord label sequence as well as binary encodings of chord roots and qualities. This binary encoding directly exposes similarities between related classes, allowing the model to learn a more coherent representation of simultaneous pitch content. Evaluations on a corpus of 1217 annotated recordings demonstrate substantial improvements compared to previous models.

1. INTRODUCTION

Automatic chord recognition has been an active area of research within music informatics for nearly two decades [8]. Chord recognition systems take as input an audio signal, and produce a time-varying symbolic representation of the signal in terms of *chord labels*, which encode simultaneous pitch class content, such as `C:maj` or `G:hdim7`. Many systems focus on simplified versions of this task, by predicting only the root note and *major* or *minor* qualities, or *no-chord* (`N`). Recently, interest has shifted toward the *large-vocabulary* regime, where a broader class of chord qualities must be estimated, such as triads, sixths, sevenths, and suspended chords.

Typical chord recognition systems model the task as a time-varying multi-class classification problem. This approach may be reasonable for the small-vocabulary regime, where the classes are sufficiently distinct to be modeled as unrelated, and each class may be observed with approximately uniform probability. However, in the large-vocabulary setting, the multi-class formulation ignores the structural similarity between related chords, such as the shared notes between `C:min` and `C:min7`. Moreover, the distribution of classes becomes highly skewed, thereby making it difficult to model these relationships from purely symbolic representations with no additional structure. We hypothesize that leveraging known relationships between chord classes in terms of common roots and shared pitch classes can help mitigate the problem of observation bias, resulting in more accurate models of rare classes.

1.1 Our Contributions

We address the problem of large-vocabulary chord recognition by introducing a structured representation of chord qualities, which decouples the problem of detecting roots and pitch classes from the problem of mapping these properties onto symbolic labels. We integrate this representation with deep, convolutional-recurrent neural networks, which are trained end-to-end to predict time-varying chord sequences from spectral audio representations. The proposed models achieve substantially higher accuracy than previous models based on convolutional networks and hidden Markov models, resulting in absolute gains of 4–5% in the most difficult categories (sevenths and tetrads).

2. RELATED WORK

Chord recognition has received a substantial amount of attention in the MIR literature, and a comprehensive survey of existing methods is beyond the scope of this paper. Here, we highlight the work that is most closely related to the proposed methods in this paper.

Hidden Markov models (HMMs) have been a popular method for designing chord recognition systems, and provide a flexible framework in which to integrate musical domain knowledge. The general HMM approach models chord identities as latent state variables to be inferred from observed time-series features (*e.g.*, chroma vectors). Systems like Chordino [17] and HPA [20] extend this idea by introducing additional latent variables to model key, bass, and metrical position. In these systems, bass is modeled by weighting or partitioning the frequency range to produce distinct *bass* and *treble chroma* observations. The K-stream HMM takes this idea a step further by modeling K

distinct frequency sub-bands, though it does not explicitly infer bass [5]. The structured representation we describe in Section 3 differs in that root, bass, and chord quality are jointly inferred from the entire spectrum, and it makes no assumptions about absolute height. Weller *et al.* [23] also adapted structured training techniques for chord recognition, but at the level of dynamics rather than the chord vocabulary.

In recent years, deep learning methods have been increasingly popular for chord recognition. The majority of existing systems are trained in two stages. First, a model is built first to encode short patches of audio, *e.g.*, as an idealized chroma vector [2, 16] or likelihood distribution over chord categories [7, 11, 22, 24]. Second, a dynamics model integrates the time-series of learned representations to produce a sequence of predicted labels, *e.g.*, using an HMM [11, 24], recurrent neural network (RNN) [2, 22], or simple point-wise prediction [16]. The models proposed here differ in that they are jointly trained end-to-end from spectral features, and learn the internal representation along with the dynamics using multiple recurrent layers.

Regardless of the model architecture, it is common to exploit some structural properties of chords, *e.g.*, by tying model parameters for the same chord quality across roots [11], or rotating chroma vectors through all possible root positions during training [5]. Although the methods we propose do not model quality independent of root, they do model active pitch class sets independently. Chroma rotation can be viewed as a form of data augmentation, and the models we develop benefit substantially from a slightly more general form of augmentation described in Section 3.4. To the best of our knowledge, the proposed method is the first to exploit similarities between chords by jointly modeling labels and structured encodings.

3. METHODS

This section outlines the data preparation, architectures, and training strategies for the models under comparison. We consider three independent design choices: convolutional or recurrent decoding, the inclusion of structured output training, and the use of data augmentation. This results in eight model configurations.

3.1 Encoder-decoder models

The models we investigate fall under the umbrella of *encoder-decoder* architectures [3]. The *encoder* component maps time-varying input (audio) into a latent feature space, while the *decoder* component maps from the latent feature space to the output space (chord labels).

3.1.1 The encoder architecture

The encoder, and depicted in Figure 1, is common to all models considered in this paper. Input audio is represented as a $T \times F$ time-series of log-power constant-Q transform (CQT) spectra (for T frames and F frequency bands). After batch normalization [13], the first convolutional layer consists of a single two-dimensional 5×5 filter, followed by a bank of 36 single-frame, one-dimensional convolutional filters, resulting in a $T \times 36$ feature map. Both layers use rectified linear (ReLU) activations. The first layer can be interpreted as a harmonic saliency enhancer, as it tends to learn to suppress transients and vibrato while emphasizing sustained tones. The second layer summarizes the pitch content of each frame, and can be interpreted as a local feature extractor.

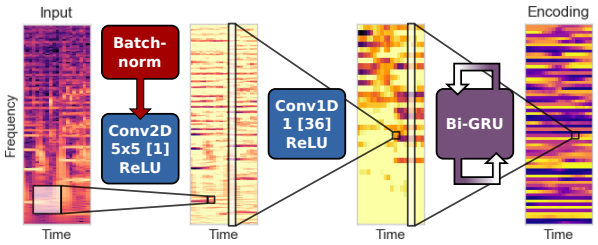

Figure 1. The encoder module uses a convolutional-recurrent network architecture to map the input (CQT frames) to a sequence of hidden state vectors $h(t) \in \mathbb{R}^D$.

Finally, the local features are encoded by a bi-directional gated recurrent unit (GRU) model [4]. The GRU model is similar to the long-short-term memory (LSTM) model [10], but has fewer parameters and performs comparably in practice [14]. For a sequence of d-dimensional input vectors $x(t) \in \mathbb{R}^d$, a GRU layer produces a sequence of D-dimensional hidden state vectors $h(t) \in [-1, +1]^D$ as follows:

$$r(t) = \sigma\left(W_r x(t) + T_r h(t-1) + b_r\right) \quad (1)$$
$$u(t) = \sigma\left(W_u x(t) + T_u h(t-1) + b_u\right) \quad (2)$$
$$\hat{h}(t) = \rho\left(W_h x(t) + T_h\left(r(t) \odot h(t-1)\right) + b_h\right) \quad (3)$$
$$h(t) = u(t) \odot h(t-1) + (1 - u(t)) \odot \hat{h}(t), \quad (4)$$

where $r(t), u(t) \in [0, 1]^D$ are the *reset* and *update* vectors, each of which are controlled by RNN dynamics depending on the input $x(t)$ and previous hidden state $h(t-1)$, $\sigma(x) = (1 + \mathsf{e}^{-x})^{-1}$ denotes the logistic function, and $\rho = \tanh$. The parameters are the input mappings $W_* \in \mathbb{R}^{D \times d}$, transition operators $T_* \in \mathbb{R}^{D \times D}$, and bias vectors $b_* \in \mathbb{R}^D$.

When an element j of the update vector $u_j(t) \approx 1$, the corresponding element of the previous hidden state is copied directly to the current state $h_j(t) \leftarrow h_j(t-1)$. Otherwise, if $r(t) \approx 1$, then $h(t)$ evolves according to standard RNN dynamics. However, when both $u(t), r(t) \approx 0$, the h term in (3) goes to 0 and the update *resets*, depending only on the input $x(t)$. This allows the GRU model to persist a hidden state across arbitrarily long spans of time, and capture variable-length temporal dependencies. These properties make the GRU model appealing for chord recognition, where dependencies may span long ranges (compared to frames), and are subject to sudden changes rather than gradual evolution.

The bi-directional variant consists of two independent GRUs, one running in each temporal direction, whose

hidden state vectors are concatenated to produce the bi-directional hidden state vector $h(t)$. This layer integrates over the entire input signal, and provides temporal smoothing and context for the encoded feature representation.

3.1.2 *Decoder architectures*

We investigate two models, depicted in Figure 3, for decoding $h(t)$ to the sequence of chord labels $\hat{y}(t)$. The first model, denoted *CR1*, decodes each frame independently:

$$\hat{y}(t) := \text{softmax}\left(W_y h(t) + b_y\right), \tag{5}$$

where the soft-max operates over the chord vocabulary V, producing a likelihood vector $\hat{y}(t) \in [0,1]^{|V|}$. For the *CR1* architecture, we set the dimensionality of the hidden state vector to 512 (256 for each temporal direction).

The second model, denoted *CR2*, uses a bi-GRU layer to map $h(t)$ to an intermediate representation $h_2(t)$ prior to frame-wise decoding by eq. (5). To keep the number of parameters roughly comparable between *CR1* and *CR2*, we set the dimensionality *CR2*'s recurrent layers to 256.

For each configuration, all model parameters Θ are jointly trained to maximize the empirical log-likelihood:

$$\underset{\Theta}{\text{argmax}} \sum_t \sum_{c \in V} y_c(t) \log \hat{y}_c(t), \tag{6}$$

where the reference labels are one-hot encoded vectors $y(t) \in \{0,1\}^{|V|}$. While both architectures have access to the entire observation sequence, *CR2* may be better at capturing long-range interactions. This should allow the encoder to focus on short-term smoothing and local context, while the decoder can model chord progressions and global context. In the *CR1* model, the encoder is responsible for both short- and long-range interactions.

At test time, the maximum likelihood label is selected for each frame, and the series of chord labels is run-length encoded to form the estimated annotation for the track.

3.2 Chord vocabulary simplification

To formulate chord recognition as a classification task, we define a mapping of all valid chord descriptions to a finite vocabulary V.[1] First, inversions and suppressed or additional notes are discarded, *e.g.*:

`D♭:maj(9)/3` $\mapsto$ `D♭:maj/3` $\mapsto$ `D♭:maj`.

Next, labels are decomposed into *root* and *pitch classes* (relative to the root) using `mir_eval` [21]:

$$\texttt{D♭:maj} \mapsto \begin{cases} 1 & \text{root} \\ (0,4,7) & \text{pitch classes} \end{cases}.$$

The set of active pitch classes is matched against 14 templates: `min`, `maj`, `dim`, `aug`, `min6`, `maj6`, `min7`, `minmaj7`, `maj7`, `7`, `dim7`, `hdim7`, `sus2`, `sus4`. The root and matched template are combined, and mapped to a canonical form to resolve enharmonic equivalences:

$$(1,(0,4,7)) \mapsto \texttt{C♯:maj}.$$

If the pitch class set does not match one of the templates, it is mapped to the unknown chord symbol `X`; the no-chord symbol is represented distinctly as `N`. The final vocabulary contains 170 classes: 2 special symbols (`N`, `X`), and $12 \times 14 = 168$ combinations of root and quality.

3.3 Structured training

The CR models described above map each hidden state vector $h(t)$ to a fixed vocabulary produced described in Section 3.2. They can be optimized in the usual way to maximize (6), but this approach has some clear drawbacks.

First, it does not leverage the inherent structure of the space of chords. If the model predicts `B:maj` instead of `B:7`, it is penalized just as badly as if it had predicted `C:maj`. This is at odds with evaluation, where predictions are evaluated along multiple dimensions, such as capturing the root, third, or fifth. More generally, some mistakes are simply more severe than others, and this is not reflected in a 1-of-K classification formulation.

Second, the chord simplification strategy is *lossy* in that it discards information such as suppressed or additional notes. This can render certain chords ambiguous, and can introduce discrepancies between the (simplified) annotation and the corresponding acoustic content. Continuing the `D♭:maj(9)/3` example, the simplification `C♯:maj` implies the absence of `D♯`, although it was explicitly included in the original annotation and should be expected in the signal. This introduces label noise to the model, and may negatively impact accuracy.

Third, out-of-gamut chords all map to a common class `X`, despite having disparate roots and tonal content. This class provides little useful information to the model while training. At test time, it would be beneficial if the model could predict "nearby" chords, but multi-class training provides little incentive to learn this behavior.

To counteract these effects, we introduce a structured representation, depicted in Figure 2. This is inspired by the standard evaluation criteria for chord recognition, which operate over a decomposed representation of (*root*, *pitch classes*, *bass*) [21]. This representation can be computed for any valid chord label, and provided as supervision to the model, thereby helping it learn common features shared by similar chords. At prediction time, the structured representation is used as an intermediate representation which contributes to the chord label prediction, which can now be interpreted as a human-readable decoding of the structured representation.

The structured models (denoted as *CR1/2+S*), depicted in Figure 3, predict for each frame t the root pitch class (`C`–`B`, plus `N` for no-root), the bass pitch class, and the active pitch classes from the hidden state vector $h(t)$. Root and bass estimation are modeled as a multi-class prediction with a soft-max non-linearity. Pitch class prediction

[1] A *valid chord* is any string belonging to the formal language of Harte *et al.* [9], or the extended grammar implemented by JAMS [12].

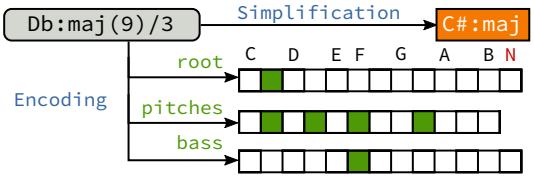

Figure 2. Target chords are represented in both simplified canonical form (Section 3.2), and as binary vectors encoding the root, bass, and pitch classes (Section 3.3). The special symbols `N`, `X` map to an extra root/bass class `N`, and the all-zeros pitch vector.

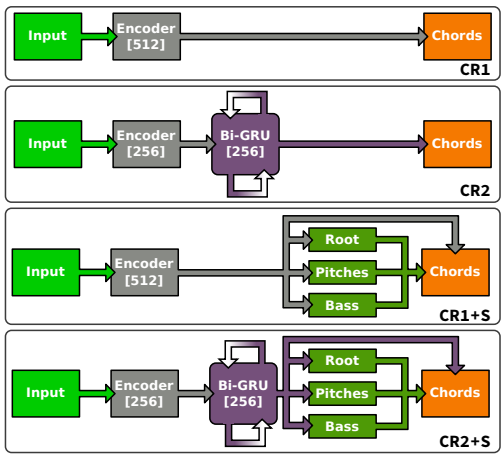

Figure 3. Block diagrams of all architectures described in Section 3. The *encoder* block is depicted in Figure 1.

is modeled as a multi-label prediction, and uses a logistic (sigmoid) non-linearity. This results in an idealized chroma representation similar to that of Korzeniowski and Widmer [16], but estimated from the full input observation rather than a fixed spectrogram patch. An illustrative example of this predicted encoding is provided in Figure 4.

It is generally non-trivial to invert the root-pitch-bass representation to a unique chord label. Therefore, these three layers are concatenated, along with the hidden state $h(t)$, to produce the structured representation from which the chord label is predicted. During training, the structured models learn to minimize the sum of losses across all outputs: root, pitches, bass, and label $\hat{y}(t)$. Minimizing the *root* and *pitches* losses corresponds to maximizing the *root* and *tetrads* recall scores during training, while eq. (6) learns the decoding into the human-readable chord vocabulary. This formulation effectively decouples the problems of root and pitch class identification from chord annotation, which is known to be subjective [11].

3.4 Data augmentation

To increase training set variability, we apply pitch-shifting data augmentation using MUDA [18]. For each training example, 12 deformations are generated by shifting up or down by between 1–6 semitones. Because each observation exists in all twelve root classes, this provides a brute-force, approximate root invariance to the model. Models trained with data augmentation are denoted by *+A*.

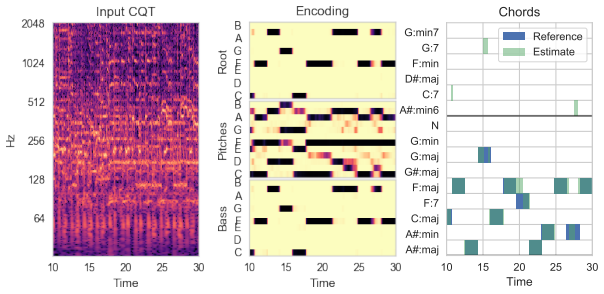

Figure 4. The predicted chord encodings and labels for *The Beatles — Hold Me Tight* by model *CR2+S+A*.

4. EVALUATION

For evaluation, we used the dataset provided by Humphrey and Bello [11], which includes 1217 tracks from the Isophonics, Billboard, RWC Pop, and MARL collections. To facilitate comparison with previous work, we retain the same 5-fold cross-validation splits, and randomly hold out 1/4 of each training set for validation. We compare to two strong baselines: a deep convolutional network [11] (denoted *DNN*), and the K-stream HMM [5] (*KHMM*). [2]

4.1 Pre-processing

Feature extraction was performed with librosa 0.5.0 [19]. Each track was represented as a log-power constant-Q spectrogram with 36 bins per octave, spanning 6 octaves starting at C1, and clipped at 80dB below the peak. Signals were analyzed at 44.1KHz with a hop length of 4096 samples, resulting in a frame rate of approximately 10.8Hz.

4.2 Training

All models are trained on 8-second patches (86 frames), though they readily support input of arbitrary length. For tracks with multiple reference annotations, the output is selected uniformly at random from all references for the patch, which reduces sampling bias toward specific annotators. Models are trained using mini-batches of 32 patches per batch, and 512 batches per epoch. We use the ADAM optimizer [15], and reduce the learning rate if there is no improvement in validation score after 10 epochs. Training is stopped early if there is no improvement in validation score after 20 epochs, and limited to a maximum of 100 epochs total. For all models, validation score is determined solely by label likelihood (eq. (6)). All models were implemented with Keras 2.0 and Tensorflow 1.0 [1, 6]. [3]

4.3 Results

The main results of the evaluation are listed in Figure 5, which illustrates the median weighted recall scores achieved by each model. [4] Each subplot reports the recall

[2] Comparisons were facilitated using the pre-computed outputs provided at `https://github.com/ejhumphrey/ace-lessons`.

[3] Our implementation is available at `https://github.com/bmcfee/ismir2017_chords`.

[4] The trends for the mean scores are qualitatively similar, but the scores are lower for all models. We report the median here to reduce the in-

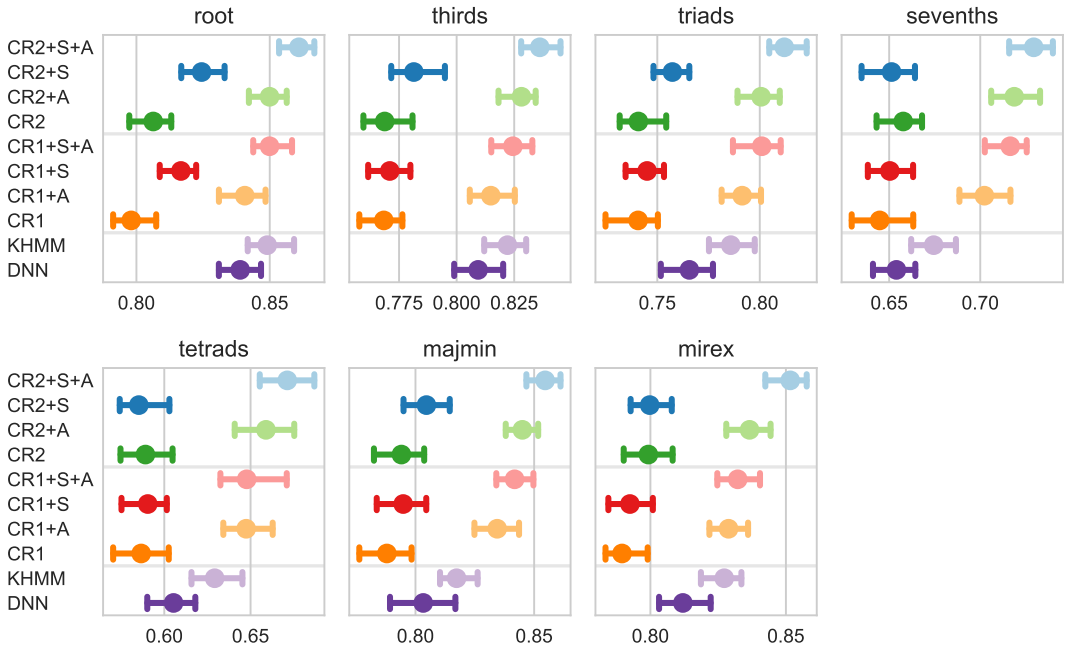

Figure 5. Weighted recall scores for all methods under comparison. Each dot represents the median score across all test points, with error bars covering the 95% confidence interval estimated by bootstrap sampling. *KHMM* denotes the K-stream HMM of Cho [5]; *DNN* denotes the convolutional network of Humphrey and Bello [11].

scores computed by `mir_eval`: 1. *root*; 2. *thirds*: root and third; 3. *triads*: root, third, and fifth; 4. *sevenths*: root, third, fifth, and seventh; 5. *tetrads*: all intervals; 6. *maj-min*: 12 major, 12 minor, and `N` class; and 7. *MIREX*: at least three correct notes.

From Figure 5, several trends can be observed. First, data augmentation (+*A* variants) provides a consistent and substantial improvement for all models. This is to be expected, since the *CR* models do not separate root from quality. Note that DNN models these independently, and KHMM was trained with chroma-rotation data augmentation, so it is unsurprising that augmentation is necessary to match performance of these methods.

Second, structured training (+*S* variants) provides a modest, but consistent improvement, for both the shallow *CR1* and deep *CR2* decoder models. The difference is most pronounced in the *root* evaluation, which is expected due to the explicit objective to correctly identify the root.

Third, the deep decoder models *CR2* provide another small, but consistent improvement over the shallow decoders *CR1*. The aggregate scores are reported in Table 1; for brevity, only the models with data augmentation are included. The combined effect of structured training, deep decoder, and data augmentation (*CR2+S+A*) results in the highest scoring model across all metrics.

4.4 Error analysis

To get some more insight about the mistakes made by the model at test time, we illustrate the frame-wise, within-root quality confusion matrix for the *CR2+S+A* model in Figure 6. For each frame of a test track, its (simplified) reference label is compared to the label estimated by the model if they match at the root. Results are then aggregated across all test tracks, and normalized by (reference quality) frequency to produce the confusion matrix. Under this evaluation, the *CR2+S+A* achieves 63.6% accuracy of correctly identifying the simplified chord label (root and quality) at the frame level.

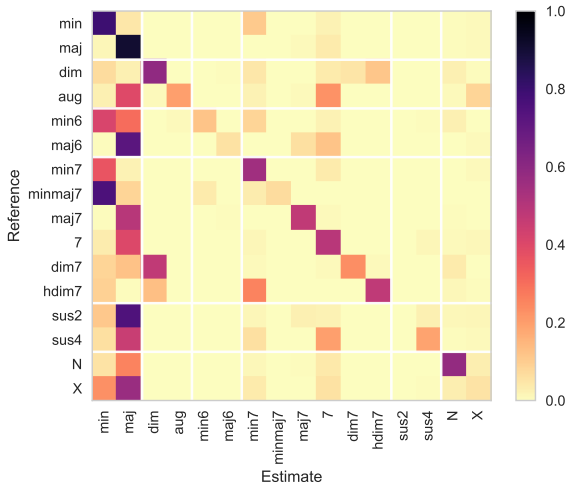

Figure 6. Within-root, frame-wise quality confusions for the best performing model *CR2+S+A*. The value at row i, column j corresponds to the fraction of frames labeled as class i but predicted as class j.

fluence of the erroneous or otherwise spurious reference annotations reported by Humphrey and Bello [11].

Method	Root	Thirds	Triads	Sevenths	Tetrads	Maj-Min	MIREX
CR2+S+A	0.861	0.836	0.812	0.729	0.671	0.855	0.852
CR2+A	0.850	0.828	0.801	0.719	0.659	0.845	0.837
CR1+S+A	0.850	0.824	0.801	0.716	0.648	0.842	0.832
CR1+A	0.841	0.815	0.791	0.702	0.647	0.834	0.829
KHMM [5]	0.849	0.822	0.785	0.674	0.629	0.817	0.827
DNN [11]	0.838	0.809	0.766	0.654	0.605	0.803	0.812

Table 1. Median weighted recall scores for methods under comparison.

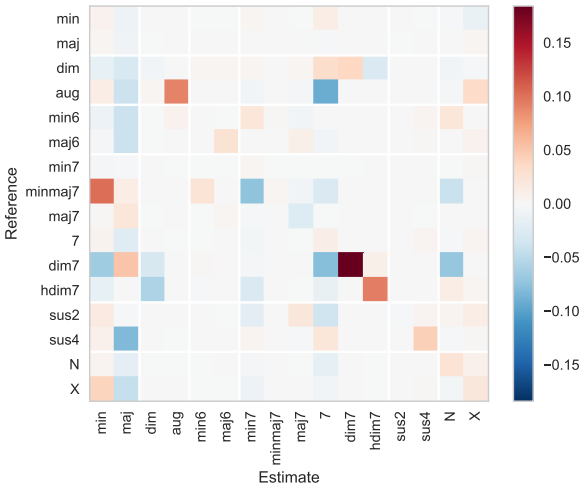

Figure 7. The difference between confusion matrices for *CR2+S+A* and the unstructured *CR2+A* (best viewed in color). Positive values along the diagonal indicate increased accuracy for *CR2+S+A*.

In Figure 6, the first obvious trend is a bias toward `min` and `maj`, in accordance with the natural bias in the training set (13.6% and 52.5% of the data, respectively, by duration). Note, however, that the confusions are generally understandable as simplifications: *e.g.*, (`min7`,`minmaj7`)→`min` and (`maj7`,`7`)→`maj`. The model still appears to struggle with 6th and suspended chords, which account for 1.5% and 2.5% of the data, respectively. The bottom row corresponds to out-of-gamut `X`-chords, which map overwhelmingly to `maj` and `min`. This can be explained by examining which labels map to `X` during simplification. There are 4557 instances of such chords in the corpus (2.2% of the data), and of these, 2091 are 1-chords (only the root) and 2365 are power chords (root+fifth), neither of which map unambiguously onto the simplified vocabulary. The model appears to resolve these toward the more commonly used `min` and `maj` qualities.

To understand the influence of structured training, Figure 7 illustrates the difference between the confusion matrices of the structured model *CR2+S+A* and the unstructured model *CR2+A*. Positive values (red) along the diagonal indicate increased accuracy for the structured model, while negative values along the diagonal (blue) indicate decreased accuracy. The net effect is positive, increasing accuracy by +0.8% over *CR2+A* (62.8%).

Despite a slight degradation for `maj7`, there are substantial improvements for `aug`, `dim7`, `hdim7`, and modest improvement for `sus4`. Moreover, the negative values in the second column reveal a consistent reduction of confusions to *maj*. This indicates that the structured model is more robust to quality bias in the training set. Compared to the unstructured model, the structured model reduces confusions from `aug` to (`maj`, `7`), and `dim7` to (`min`, `7`, `N`). The *CR2+S+A* still performs poorly on the rarest class `minmaj7` (0.03% of the data), but compared to *CR2+A*, it resolves toward `min` more often and `min7` less often. The structured model appears to be better at abstaining from predicting a seventh if it appears unlikely, rather than predict the wrong seventh.

5. CONCLUSION

This work developed deep architectures and a structured training framework for chord recognition in large vocabularies. Although the proposed models improve over the baseline methods, there are clear directions forward in extending the ideas presented here. First, although the proposed model predicts the bass note, this feature is only used for establishing context in decoding, and the model does not predict inversions. Supporting inversion prediction would be a simple extension of the method described here, and would not require creating special vocabulary entries for each potential inversion. Second, the structured representation facilitates modeling infrequently observed, complex chords, and could readily be extended to support extended chords by using a multi-octave pitch class representation. However, doing so effectively—and evaluating the resulting predictions—would require larger annotated corpora for these classes than are presently available.

6. ACKNOWLEDGMENTS

BM acknowledges support from the Moore-Sloan data science environment at NYU. We thank the NVIDIA Corporation for the donation of a Tesla K40 GPU.

7. REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek

Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

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

[3] Kyunghyun Cho, Aaron Courville, and Yoshua Bengio. Describing multimedia content using attention-based encoder-decoder networks. *IEEE Transactions on Multimedia*, 17(11):1875–1886, 2015.

[4] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 1724–1734, 2014.

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

[6] François Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[7] Junqi Deng and Yu-Kwong Kwok. A hybrid gaussian-hmm-deep-learning approach for automatic chord estimation with very large vocabulary. ISMIR, 2016.

[8] Takuya Fujishima. Realtime chord recognition of musical sound: a system using common lisp music. In *ICMC*, pages 464–467, 1999.

[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*, pages 66–71, 2005.

[10] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[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, Justin Salamon, Oriol Nieto, Jon Forsyth, Rachel M Bittner, and Juan Pablo Bello. JAMS: A JSON annotated music specification for reproducible MIR research. In *ISMIR*, pages 591–596, 2014.

[13] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 448–456, 2015.

[14] Rafal Józefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 2342–2350, 2015.

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

[16] Filip Korzeniowski and Gerhard Widmer. Feature learning for chord recognition: the deep chroma extractor. In *ISMIR*, 2016.

[17] Matthias Mauch and Simon Dixon. Approximate note transcription for the improved identification of difficult chords. In *ISMIR*, 2010.

[18] Brian McFee, Eric J Humphrey, and Juan Pablo Bello. A software framework for musical data augmentation. In *ISMIR*, pages 248–254, 2015.

[19] Brian McFee, Matt McVicar, Oriol Nieto, Stefan Balke, Carl Thome, Dawen Liang, Eric Battenberg, Josh Moore, Rachel Bittner, Ryuichi Yamamoto, Dan Ellis, Fabian-Robert Stoter, Douglas Repetto, Simon Waloschek, CJ Carr, Seth Kranzler, Keunwoo Choi, Petr Viktorin, Joao Felipe Santos, Adrian Holovaty, Waldir Pimenta, and Hojin Lee. librosa 0.5.0, February 2017.

[20] 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.

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

[22] Siddharth Sigtia, Nicolas Boulanger-Lewandowski, and Simon Dixon. Audio chord recognition with a hybrid recurrent neural network. In *ISMIR*, pages 127–133, 2015.

[23] Adrian Weller, Daniel Ellis, and Tony Jebara. Structured prediction models for chord transcription of music audio. In *Machine Learning and Applications, 2009. ICMLA'09. International Conference on*, pages 590–595. IEEE, 2009.

[24] Xinquan Zhou and Alexander Lerch. Chord detection using deep learning. In *ISMIR*, 2015.