FROM MUSIC AUDIO TO CHORD TABLATURE: TEACHING DEEP CONVOLUTIONAL NETWORKS TO PLAY GUITAR

Eric J. Humphrey and Juan P. Bello

Music and Audio Research Laboratory (MARL) @ NYU

ABSTRACT

Automatic chord recognition is conventionally tackled as a general music audition task, where the desired output is a time-aligned sequence of discrete chord symbols, e.g. CMaj7, Esus2, etc. In practice, however, this presents two related challenges: one, the act of decoding a given chord sequence requires that the musician knows both the notes in the chord and how to play them on some instrument; and two, chord labeling systems do not degrade gracefully for users without significant musical training. Alternatively, we address both challenges by modeling the physical constraints of a guitar to produce *human-readable representations* of music audio, i.e guitar tablature via a deep convolutional network. Through training and evaluation as a standard chord recognition system, the model is able to yield representations that require minimal prior knowledge to interpret, while maintaining respectable performance compared to the state of the art.

***Index Terms*—** deep networks, chord recognition, representation learning, guitar tablature

1. INTRODUCTION

Given sustained effort and interest by the research community, automatic chord recognition is now one of the seminal tasks in music informatics. As the state of the art has advanced, most approaches adopt the same basic architecture: first, acoustic features, known as pitch class profiles or chroma, are computed from short-time observations of an audio signal [1]; then, Gaussian mixture models (GMMs) are fit to each chord class in the target vocabulary, typically 12 Major, 12 Minor, and one waste-basket "no-chord" class [2]; finally, frame-wise predictions are decoded using a Hidden Markov Model with transition probabilities computed from the same data used for training the GMMs [3]. Alternatively, deep learning methods have recently garnered attention for chord recognition, such as convolutional neural networks (CNNs)[4] and recurrent neural networks (RNNs) [5].

Generally speaking, the majority of prior research in automatic chord recognition is based on the two-fold premise that (a) this is fundamentally a classification problem and (b) the ideal output is a time-aligned sequence of singular chord names, motivated by the spirit of empowering anyone to play any song. Modern online guitar communities, however, have continued to place a high demand on guitar tablature, given the prevalence of user-curated websites like Ultimate Guitar[1], which sees an average 1.7M unique visitors in the US per month[2]. As illustrated in Figure 1, guitar tablature is a form of music notation that requires minimal knowledge to interpret because the representation directly maps notes in a chord to frets on a guitar. Therefore, it is an inherent design challenge of human-facing expert systems that the output must be easily interpreted by the user; and, more importantly, graceful degradation is a function of that user's capacity to understand and recover from errors. Though some previous work embraces this position in the realm of transcribing guitar recordings [6] or arranging music for guitar [7], there is, to our knowledge, no existing work in estimating guitar tablature directly from polyphonic recordings.

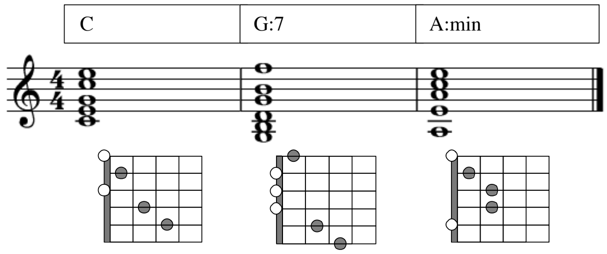

Fig. 1. A chord label sequence (top), traditional staff notation (middle), and guitar tablature (bottom) of the same musical information, in decreasing levels of abstraction.

This paper presents a novel approach to bootstrapping the task of automatic chord recognition to develop an end-to-end system capable of representing polyphonic music audio as guitar tablature by modeling the mechanics of the guitar with a deep convolutional network. To enforce playability, a finite vocabulary of chord shape templates are defined and the network is trained by minimizing the distance between its output and the best template for an observation. Our experiments show that the model achieves the goal of faithfully mapping

This material is based upon work supported by the National Science Foundation under grant IIS-0844654.

[1] http://www.ultimate-guitar.com/

[2] Based on Compete.com analytics data, accessed on 2 November, 2013.

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

audio to a fretboard representation, while still performing respectably as a chord recognition system. The output of the network is human-readable, allowing the system to be used by anyone regardless of musical ability. Additionally, trained networks are not constrained to any particular vocabulary, and are able to represent previously unseen chord shapes.

2. PROPOSED SYSTEM

2.1. Input Representation

The first stage of our system is a constant-Q filterbank [8], motivated by two reasons. First, a filterbank frontend greatly reduces the dimensionality of the input data. Second, the constant-Q transform is linear in time and pitch, a property that can be exploited by a convolutional architecture to simplify the learning task. Acting as multi-band automatic gain control, local contrast normalization is then applied to the time-frequency representation, as outlined in [9], with a slight modification to the smoothing kernel. Typically the kernel used to low-pass filter the representation is a symmetric, truncated Gaussian in both dimensions, but we instead use a half-Hanning window in time to mimic the precedence effect and reduce phase delay.

2.2. Designing a Fretboard Model

Deep trainable networks have proven to be a versatile, powerful, and practical approach to solving complex machine learning problems in a variety of fields. A deep network transforms an input X_{in} into an output Z_{out} via a composite nonlinear function $F(\cdot|\Theta)$ given the parameter set Θ, often realized as a cascade of L simpler nonlinear functions $f_l(\cdot|\theta_l)$, referred to as layers, indexed by l:

$$F(X_{in}|\Theta) = f_{L-1}(...f_1(f_0(X_{in}|\theta_0)|\theta_1))...|\theta_{L-1}) \quad (1)$$

such that $F = [f_0, f_1, ...f_{L-1}]$ is the set of layer functions, $\Theta = [\theta_0, \theta_1, ...\theta_{L-1}]$ is the corresponding set of layer parameters, and the output of one layer is passed as the input to the next, as $X_{l+1} = Z_l$. Here, based on previous experience [4], we define our deep network with $L = 4$; diagrammed in Figure 2, the first two layers are convolutional, the third is a fully connected matrix product, and the fourth is a fully connected tensor product with a variant of the softmax operation, the details of which will be addressed in turn.

The two convolutional layers, f_l for $l \in [0, 1]$, are defined as follows:

$$f_l(X_l|\theta_l) = pool(h(X_l \circledast W + b), p), \theta_l = [W, b, p] \quad (2)$$

The operator $\circledast$ is evaluated by convolving a 3D input tensor X, known as a set of *feature maps*, with a 4D weight tensor W, known as a set of *kernels*, followed by a vector bias term b. Note that the nonlinearity, denoted by $h(\cdot)$, is a rectified linear unit, defined as $h(x) = max(x, 0)$, and is common to all layers. In this formulation, X has shape (N, d_0, d_1) where N is the number of feature maps, and (d_0, d_1) is the shape of each map, W has shape (M, N, m_0, m_1), where M is the output number of feature maps, N aligns with the first dimension of X, and (m_0, m_1) correspond to the feature dimensions of X. The bias b has length M, and the parameter p is a two-element tuple that defines the neighborhood of the max operator along the feature dimensions. Explicitly, the convolution $\hat{X}_m = X \circledast W_m$ can be written as follows:

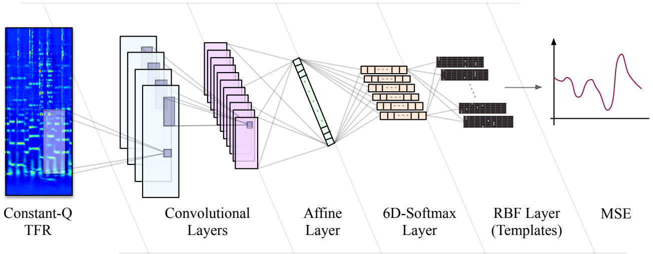

Fig. 2. Full diagram of the proposed network during training.

$$\hat{X}[a, b]_m = \sum_{i=0}^{N} \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} X[i, x, y] W_m[i, a-j, b-k] \quad (3)$$

There are two primary benefits to using convolutional layers in a deep network; the number of parameters to learn is greatly reduced because only a local neighborhood, rather than the entire input, is processed by each kernel; and two, the kernel is translated across all shifts of the input, allowing it to encode the same feature regardless of absolute position; we refer the interested reader to [9] for a more extensive review. Having designed the input time-frequency representation to be linear in both time and pitch, this property is particularly useful in the task of chord recognition, where the quality of a chord is defined by the relative intervals of its notes.

The third layer, f_2, is a fully-connected, or affine, layer, defined by the following:

$$f_2(X_2|\theta_2) = h(W \bullet X_2 + b), \theta_i = [W, b] \quad (4)$$

Here, the input X_2 is flattened to a column vector of length L and the dot-product is computed with a weight matrix W of shape (M, L), followed by a vector bias term with length M.

Finally, the last layer, f_3, that produces the fretboard-like output, Z_{out}, is defined similarly to f_2, with the exception that W is now a tensor, and the result of each matrix-vector product is normalized by the softmax operation, defined as:

$$\sigma(x) = \frac{\exp(x)}{\sum_{m=1}^{M} \exp(x[m])} \quad (5)$$

This layer is then given by the following:

$$f_3(X_3|\theta_3) = \sigma(h(W[k] \bullet X_3 + b)), k \in [0:K), \theta_3 = [W, b] \quad (6)$$

Again, as in the previous layer, the input is a flat vector of length N, but the weights are now a tensor of shape (K, M, N) and the bias is a matrix with shape (K, M). This layer is equivalent to performing K parallel affine transformations, each normalized by the softmax function. The intuition for this design is straightforward; a guitar consists of a finite number of strings, and, because the instrument is fretted, each can only be pressed at one of a few discrete positions on the neck. In this way, the K-parallel softmax surfaces behave like probability mass functions for each string of the guitar, and the output Z_{out} is shaped (K, M).

3. GUITAR LESSONS FOR DEEP NETWORKS

3.1. Chord Shape Vocabulary

In this work,we consider five chord qualities (maj, min, maj7, min7 and 7) in all 12 pitch classes, plus one no-chord class, for a total of 61 classes. Chord shapes are designed such that all qualities with the same root are formed in same neck position, as seen in the first column of Figure 3. Consequently, all chords with the same root will be near-neighbors in this representation, and we should expect that the most common confusions will occur between qualities. Additionally, though guitar chords can be voiced in variety of ways, here we consider a single shape for each as an initial simplification.

3.2. Loss Function

Having designed a fretboard model, we turn our attention to designing a loss function such that the machine can learn to faithfully reproduce it. Following the lead of [10], we train the network through an additional Radial Basis Function (RBF) layer, given as follows:

$$\mathcal{L}(Z_{out}|W_T) = \sum (Z_{out} - W_T[i])^2 \quad (7)$$

where Z_{out} is the output of the fretboard model, W is a tensor of chord shape templates with shape (C, K, M), C is the number of chord shapes, and i is the index of the correct class. Note that these templates will impose the proper organization on the output of the model, and thus remain fixed during the learning process. Since these weights are constant, minimizing this function does not require a contrastive penalty or margin term to prevent it from collapsing, i.e. making all the squared distances zero.

3.3. Training Strategy

For training the model, we use mini-batch stochastic gradient descent with a constant learning rate of 0.02. When sampling the data, each observation is circularly shifted in pitch randomly on the interval [-12,12]. This allows the variance of each chord quality to be evenly distributed across classes, effectively turning the task into a 6-class problem as each datapoint contributes equally to pitch class. Empirically, we observed that a mini-batch with a uniform quality distribution led to poor discrimination of Major chords. This is likely due to the wide intra-class variance of major chords, which we offset by constituting batches with 3 Major observations for 1 of each other quality, and use a total batch size of 64 (24 Major, 8 Minor, 8 no-chord, etc).

4. METHODOLOGY

4.1. Dataset

Similar to [4], we conduct our work on a set of 475 music recordings, consisting of 181 songs from Christopher Harte's Beatles dataset[3], 100 songs from the RWC Pop dataset and 194 songs from the US Pop dataset[4]. Importantly, chord classes exhibit a power-law distribution where a small number of classes —mostly major and minor chords— live in the short head, while more obscure chords occur only a handful of times. We split this data into 5 folds for cross validation, using 4:1 as our train-to-test ratio. Additionally, some 200 disjoint tracks from McGill's Billboard dataset [11] were identified on YouTube and added to all training sets.

4.2. Model Parameters

For our constant-Q input, we use 24 bins per octave, over 8 octaves, and grouped into 4 second windows at a framerate of 20Hz, and thus the input to the network is a matrix with shape (80,192). Again, motivated by previous work, the parameters of the fretboard model are as follows: layer f_0 uses $K = 16, m_0 = 15, m_1 = 27$, and $p = (2, 2)$; layer f_1 uses $K = 20, m_0 = 15, m_1 = 13$, and $p = (2, 2)$; layer f_2 uses $N = 8140$, and $M = 512$; layer f_3 uses $K = 6, M = 9$, and $N = 512$; and layer f_4 uses $C = 61, K = 6$, and $M = 9$.

4.3. Experimental Design

In lieu of measuring subjective experience of using the system, we evaluate the quality of this model in the context of chord recognition. To also quantify the impact of the guitar-specific constraints, we define a second model with the same parameter complexity outlined in Subsection 4.2, but modify the transformation of f_3 to be a fully-connected layer like f_2 and use a linear softmax classifier rather than the RBF-templates, trained to minimize the negative log-likelihood. This allows us to control for model complexity and determine the influence of the design constraints on the learning problem. Both models are run for 30k iterations.

[3] http://isophonics.net/content/reference-annotations-beatles

[4] https://github.com/tmc323/Chord-Annotations

The conventional evaluation metric in chord recognition is known as frame-wise recognition rate (FWRR), defined as the flat average of correct predictions over the entire set. Though standard, this metric is sensitive to the distribution of the data and will prefer models that do well on the predominant classes, i.e major chords. We then complement this precision measure with a conceptual parallel to recall, referred to here as average chord quality accuracy (ACQA), defined as the mean of the individual chord quality accuracies. Lastly, ground-truth chord names that are not among the qualities named in Subsection 3.1 are counted as classification errors, comprising roughly 10% of the dataset.

5. DISCUSSION

5.1. Results

After training, we evaluate these two models with the appropriate holdout data and average performance across folds. The fretboard and unconstrained models attain FWRRs of 58.26% and 58.72%, and ACQAs of 61.36% and 62.02%, respectively. For comparison, the best known system reaches a FWRR of 63.72% and a ACQA of 65.59% [12]; two other high-performing systems achieve FWRRs of 63% [13] and 64.46% [14] in a similar formulation, but the results are not directly comparable. To understand why these two deep network models have higher ACQAs than FWRRs, the accuracy for each quality is given in Table 1. Here we see that the models perform much worse on minor (≈ -10%) and 7 (≈ -20%) chords than in [12], but particularly better on no-chord (≈ +15%), helping to balance out the ACQA. This behavior causes a subpar FWRR, as the models make more errors on the more frequently occurring chords and the uneven distribution of the data manifests in the metric. Therefore ACQA better characterizes how a model will perform across a wide vocabulary of chords.

To further explore the behavior of the fretboard model, Figure 3 illustrates three instances of the model processing previously unseen data. Figure 3-a confirms that the model is indeed able to produce representations similar to ideal tablature. Figure 3-b demonstrates a chord quality confusion, a common source of error where the root is correct but the specific shape is wrong. Here, the network confuses A:7 for A:maj, as the probability of the flat-7 (0^{th} fret on the G-string) is slightly smaller than that of the octave (2^{nd} fret on the G-string). To quantify the frequency of these kinds of confusions, we map this chord vocabulary to the classic Major-Minor task. In this scenario, the fretboard model now out performs the unconstrained variant, with FWRRs of 77.42% and 76.42%, respectively; as a point of reference, the previously discussed models achieve 82.12% [12], 80% [13], 79.37% [14]. This demonstrates that vast majority errors in the fretboard model are only between sevenths. Lastly, we even observe some, albeit infrequent, instances where the fretboard model is able to correctly represent chords outside the known vocabulary. Figure 3-c is one such example, where the network correctly produce the fingering for E♭:hdim7, a chord never seen during training. Thus, the fretboard model is able to learn useful representations that are open to interpretation, and is not limited to a predetermined set of classes.

Table 1. Average accuracies by chord quality for the fretboard (FB) and unconstrained (UC) models.

	maj	min	maj7	min7	7	N
FB	69.52	55.79	63.18	55.52	46.29	77.85
UC	69.58	57.24	62.08	55.38	49.60	78.21

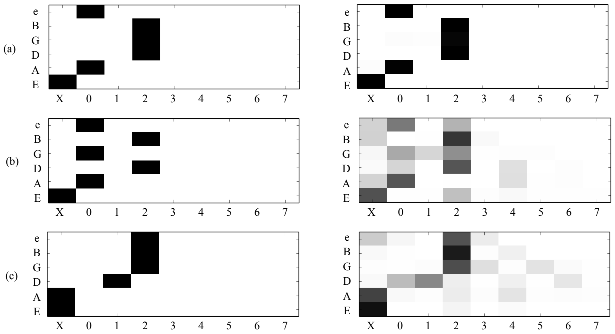

Fig. 3. Three instances of correct chord templates (left) and fretboard outputs (right) for data in the test set; (a) a correct A:maj, (b) a confusion between A:7 and A:maj, two chords that differ in only a single note, and (c) a correct E♭:hdim7, a never-before seen chord.

5.2. Conclusions and Future Work

Here we have demonstrated a novel method to produce guitar tablature from music audio that performs competitively as a chord recognition system, while offering several notable advantages over previous work. Human-readable representations of music are directly usable by the large online guitarist communities discussed previously. Additionally, this graphical representation would enable users to easily correct errors made by the system, facilitating large-scale data collection and reducing the skill necessary to provide annotations. Lastly, and somewhat remarkably, this model is able to generalize to never-before seen chord shapes.

Looking to future work, it is noteworthy that these results are obtained without any post-filtering or the application of a music language model. Empirical experimentation with HMMs failed to improve performance, but perhaps an alternative algorithm could help stabilize spurious errors and prevent unlikely transitions. That said, using this system as a means of collecting more ground-truth annotations could allow for expanded training and performance evaluation, and aid in the curation of a dataset specific to this task.

6. REFERENCES

[1] T. Fujishima, "Realtime chord recognition of musical sound: a system using common lisp music," in *Proc. Int. Computer Music Conf.*, 1999.

[2] J. P. Bello and J. Pickens, "A robust mid-level representation for harmonic content in music signals," in *Proc. ISMIR*, 2005.

[3] A. Sheh and D.P.W. Ellis, "Chord segmentation and recognition using em-trained hidden markov models," in *Proc. ISMIR*, 2003.

[4] E. J. Humphrey and J. P. Bello, "Rethinking Automatic Chord Recognition with Convolutional Neural Networks," in *Proc. Int. Conf. on Machine Learning and Applications*, 2012.

[5] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Audio chord recognition with recurrent neural networks," in *Proc. ISMIR*, 2013.

[6] A. Barbancho, A. Klapuri, L. J. Tardón, and I. Barbancho, "Automatic transcription of guitar chords and fingering from audio," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 20, no. 3, pp. 915–921, 2012.

[7] G. Hori, H. Kameoka, and S. Sagayama, "Input-output hmm applied to automatic arrangement for guitars.," *JIP*, vol. 21, no. 2, pp. 264–271, 2013.

[8] C. Schoerkhuber and A. Klapuri, "Constant-q transform tool-box for music processing,," in *Proc. Sound and Music Computing Conference*, 2010.

[9] Y. LeCun, K. Kavukvuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Proc. ISCAS*, 2010.

[10] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[11] J. A. Burgoyne, J. Wild, and I. Fujinaga, "An expert ground truth set for audio chord recognition and music analysis," in *Proc. ISMIR*, 2011.

[12] T. Cho, *Improved Techniques for Automatic Chord Recognition from Music Audio Signals*, Ph.D. thesis, New York University, 2012.

[13] M. Mauch and S. Dixon, "Simultaneous estimation of chords and musical context from audio," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 18, no. 6, pp. 1280–1289, 2010.

[14] Y. Ni, M. McVicar, R. Santos-Rodriguez, and T. De Bie, "An end-to-end machine learning system for harmonic analysis of music," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 20, no. 6, pp. 1771–1783, 2012.