

# Rethinking Automatic Chord Recognition with Convolutional Neural Networks

Eric J. Humphrey and Juan P. Bello  
*Music and Audio Research Lab (MARL)*  
*New York University*  
*New York, NY USA*  
*{ejhumphrey, jpbello}@nyu.edu*

**Abstract**—Despite early success in automatic chord recognition, recent efforts are yielding diminishing returns while basically iterating over the same fundamental approach. Here, we abandon typical conventions and adopt a different perspective of the problem, where several seconds of pitch spectra are classified directly by a convolutional neural network. Using labeled data to train the system in a supervised manner, we achieve state of the art performance through this initial effort in an otherwise unexplored area. Subsequent error analysis provides insight into potential areas of improvement, and this approach to chord recognition shows promise for future harmonic analysis systems.

**Keywords**-chord recognition; automatic music transcription; convolutional neural nets;

## I. INTRODUCTION

Even from the earliest efforts in music informatics research, automatic music transcription stands apart as one of the Holy Grails of the field. It has proven substantially more difficult than once thought however, and since fractured into a variety of smaller subtopics. Automatic chord recognition is one such task, receiving healthy attention for more than a decade and is an established benchmark at the annual MIREX challenge<sup>1</sup>. Given the prerequisite skill necessary to produce these transcriptions manually, there is strong motivation to develop automated systems capable of reliably performing this task. Applications of a computational chord recognition system are myriad, ranging from straightforward annotation to the development of compositional tools and musically meaningful search and retrieval systems.

The identification of chords is also intriguing from a musical perspective, being a high level cognitive process that is often open to multiple interpretations between knowledgeable experts. One common definition of a *chord* is the “simultaneous sounding of two or more notes” [7], but music is seldom so simple. Though the explicit use of chords can be straightforward – strumming a root-position C major on guitar, for instance – real music is typically characterized by complex tonal scenes that only *imply* a certain chord or harmony, often in the presence of nonchord tones and with no guarantee of simultaneity; one such example of this is the simple monophonic melody given in Figure 1, which clearly suggests F Major. Skilled human listeners are quite robust in

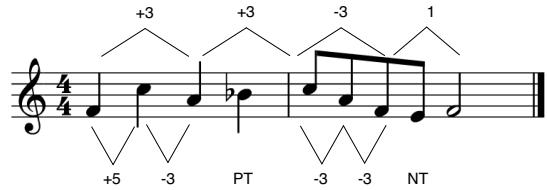


Figure 1. One interpretation of a simple melody in F. The implied chord of the phrase is an F Major triad, despite the presence of two nonchord tones falling on unaccented beats; a B $\flat$  passing tone (4th scale degree) and an E neighboring tone (7th scale degree). Intervals between notes are also indicated, showing the relative relationships between nearby pitches.

the ability to assign chord labels to musical scenes despite an occasionally opaque decision making process, and it is an exciting challenge to produce a computational system with a similar capacity.

Historically speaking, automatic chord recognition research is mostly summarized by a few seminal works. Arguably, the two most influential systems are those of Fujishima, who proposed the use of *chroma* features [3], and Sheh and Ellis, who introduced the use of Hidden Markov Models (HMMs) to stabilize chord classification [9]. The former, also known as pitch class profiles, are a short-time estimate of octave-equivalent pitch calculated several times a second. Each chroma vector is classified as a chord independently, and HMMs greatly improve performance by smoothing spurious behavior and effectively “stitching” together feature vectors that produce large classification likelihoods. Based on this early success, many current approaches adopt this interpretation of the problem and incorporate the same three stage process: fast frame-level features are calculated over short-time observations of an audio signal, instantaneous feature vectors are assigned to chord classes, and post-filtering is performed to identify the best chord classification path.

Though the implementation details have continued to evolve over the last decade, the brunt of chord recognition research has concentrated not on the fundamental system per se, but rather the tuning of its components. In particular, much time and energy has been invested in

<sup>1</sup>[http://www.music-ir.org/mirex/wiki/MIREX\\_HOME](http://www.music-ir.org/mirex/wiki/MIREX_HOME)

developing not just better features, but specifically better *chroma* features [8]. Acknowledging the challenges inherent to designing good features, Pachet et al pioneered work in automatic feature optimization [11], and more recently deep learning methods have been employed to produce robust Tonnetz features [4]. Alternatively, some work leverages the repetitive structure of music to smooth a chroma features prior to classification [1]. Various classification strategies have been investigated to a lesser extent [10], but Gaussian Mixture Models (GMM) are conventionally preferred for the probabilistic interpretation. The choice of post-filtering methods has been shown to significantly impact classification accuracy, and much research has focused on properly tuning HMMs [2], in addition to exploring other post-filtering methods such as Dynamic Bayesian Networks (DBNs) [6].

Despite steady incremental improvements, there are two important observations to draw from this research tradition: performance increases are undeniably diminishing, and the now de facto standard interpretation of the problem exhibits a few crucial shortcomings. After a period of quick progress, improvements in automatic chord recognition have stalled well below what one would be deemed to be a solved problem. Even if an optimal transform existed, chroma is conceptually limited in the vocabulary of chords it can uniquely represent. Some previous work attempts to compensate for this deficiency by supplementing chroma features with other hand-crafted statistics, e.g. bass note features [7], but a little more than a decade of hand-crafted feature design in music informatics has shown the process to be an arduous and time consuming one. Moreover, frame-level classification assumes both simultaneity in pitch space and a predominantly Gaussian distribution of the data, neither of which are particularly realistic assumptions.

In this work, we adopt a different view of chord recognition and propose a trainable, data-driven approach that automatically learns relevant hierarchical features and its classifier simultaneously. Rather than attempting to classify short-time features and relying on post-filtering to smooth the results into a musically plausible chord path, we instead use a convolutional neural network to classify five-second tiles of pitch spectra, producing a jointly optimized chord recognition system. A significant advantage of this approach is that minimal assumptions are made regarding what statistics might be informative for the task at hand, instead relying on data to objectively tease out these features. We additionally show that optimizing to labeled data provides insight into not only the task at hand, but also the ground truth data itself. The remainder of this paper is outlined as follows: Section II addresses the motivation and concepts behind the proposed system; Section III describes our experimental methodology; Section IV presents and discusses the experimental findings; and finally, Section V offers conclusions and directions for future work.

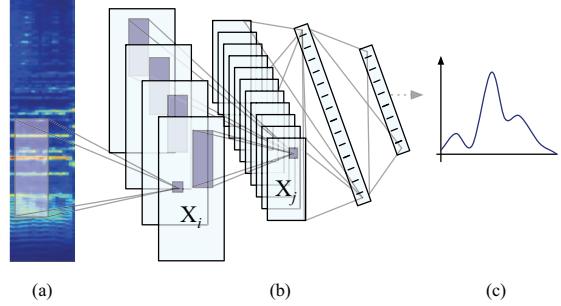


Figure 2. The CNN Chord Recognition Architecture: A time-pitch tile (a) is input to a CNN (b), yielding a probability surface (c).

## II. THE HIERARCHY OF HARMONY

Like most facets of music, chords are characterized by the hierarchical composition of more atomic elements, i.e. individual pitches combine in time to form intervals, and similarly into chords, harmonic progressions and ultimately songs. Noting that these musical building blocks of intervals take shape along the dimensions of both pitch and time, the task of chord recognition can be conceptually reformulated as a natural hierarchy of spectro-temporal events; we again refer to Figure 1, which illustrates the intervallic relationships of nearby notes over both pitch and time.

Framed in the context of relative intervals, it is possible to explain the full range of harmony, such as monophonic melodies, inverted chords, or nonchord tone embellishments through the combination of simpler parts. Chord recognition can then be reduced to two separate challenges: how should a system be architected to encode musical harmony as a hierarchy of parts, and how can we determine what these parts are? Deep convolutional architectures provide a potential answer to the first question, and automatic feature learning can be used to discover optimal feature representations.

### A. Convolutional Neural Networks

Initially inspired by research of the cat's visual cortex by Hubel & Weisel in the 1960's, convolutional neural networks (CNNs) are trainable, hierarchical nonlinear functions; we refer to [5] for a modern review. CNNs can be seen as a special instance of classic Artificial Neural Networks (ANN) where weights are *shared* over an input vector, acting like local receptive fields that "move" over an input. This movement manifests as translation invariance, allowing the same feature to be characterized at different absolute positions. For the purposes of chord recognition, a convolutional architecture learns relative intervals separate from absolute pitch height or position in time; this architecture is diagrammed in Figure 2.

Defining explicitly, a CNN is a multistage architecture composed of both convolutional and fully-connected layers

that operates on an input  $X_{in}$ , produces an output  $Z_{Prob}$ , and is described by a set of weight parameters  $W_l$ , where  $l$  indexes the layer of the machine. Layers are stacked consecutively, such that the output  $Z_l$  is taken as the input  $X_{l+1}$  of the following layer. Each layer consists of a linear projection  $G(X, W)$ , a hyperbolic tangent activation and an optional down-sampling, or pooling, operation, expressed generally by

$$Z_l = \text{pool}(\tanh(G(X_l, W_l) + W_{l_0})). \quad (1)$$

The function  $G$  of a convolutional layer is a 3-dimensional operation, where an input tensor  $X_l$ , known as a collection of  $N$  *feature maps*, is convolved with another tensor of weights  $W_l$ , referred to as kernels, defined by (2); the input  $X_{in}$  can be considered as a special instance of a feature map where  $N = 1$ .

$$G(X, W) = \sum_{i=0}^N \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} X[i, j, k] W_i[m-j, n-k] \quad (2)$$

The function  $G$  of a fully-connected layer is the product of a flattened input vector  $X_l$  and a matrix of weights  $W_l$ , given in

$$G(X, W) = (X \times W). \quad (3)$$

Finally, the output  $Z_{Prob}$  is defined as a softmax activation over  $C$  chord classes, as in (4), such that the  $i^{th}$  output is bounded on the interval  $[0, 1]$  and the output vector sums to one.

$$\text{softmax}(Z_l) = \frac{\exp(Z_l)}{\sum_{i=1}^C \exp Z_{i,l}} \quad (4)$$

### B. Feature Learning

Given a trainable architecture and a differentiable objective function, the true power of data-driven methods is realized in the numerical optimization of its parameters. Despite early concerns to the contrary, stochastic gradient methods have been shown to converge to “good” solutions and labeled data, when plentiful, can be used to fit the function directly. This process is not always trivial, however, and these two facets – labeled ground truth data and training strategy – require due consideration.

1) *Ground Truth Data:* Originating from a variety of different sources, we conduct our work on a set of 475 music recordings, consisting of 181 songs from Christopher Harte’s Beatles dataset<sup>2</sup>, 100 songs from the RWC Pop dataset and 194 songs from the US Pop dataset<sup>3</sup>. Based on the provided annotations, there are some 800 unique chord labels provided for just over 50k distinct chord instances.

<sup>2</sup><http://isophonics.net/content/reference-annotations-beatles>

<sup>3</sup><https://github.com/tmc323/Chord-Annotations>

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 may occur only a handful of times. Traditionally, due to both a sparse representation of chord classes in the long tail and a general simplification of the task, all chord labels are resolved to twenty-five classes: 12 major, 12 minor, and a waste-basket, “no-chord” class. While this mapping reduces the space of possible classes, it is worth noting that this process also introduces additional intra-class variance.

2) *Training Strategy:* Being mindful of nuances of the dataset, special attention must be paid to how training is conducted. The size of the dataset requires  $k$ -fold cross validation, but it is poor practice to stratify the data arbitrarily. As we intend on classifying observations on the order of seconds, musical context – the information *around* a given chord – is actually a very strong prior, and it is advisable to split the data such that all folds are, ideally, identically distributed with respect to chord transitions. Doing so should result in roughly equivalent performance across various splits of the data, and in some scenarios even eliminate the need to train and test over all  $k$  folds.

To create similarly distributed folds of the data, we use the genetic algorithm (GA) to minimize the variance of chord transition distributions. First chord transition histograms are tallied separately for each track, yielding a  $(25 \times 25 \times 475)$  tensor  $H$ . A 1-of- $k$  binary assignment matrix  $A$ , with a shape of  $(475 \times k)$ , can then be used to determine fold ownership. The fitness of a matrix  $A_i$  can be computed as the variance of its matrix product with  $H$ . We run the GA by randomly initializing a “population” of assignment matrices, followed by evaluating the fitness of each, assigning probabilities accordingly, and randomly selecting “parents” to merge. A new population is created by randomly keeping rows from each pair of parents, yielding a new set of assignment matrices. Despite concerns about GAs being computationally expensive and having no convergence guarantees, it proved to be quite fast for this application and actually converged to a consistent local minima.

For training, we adopt a mini-batch variant of stochastic gradient descent, where the loss function, defined as the Negative log-likelihood, is evaluated over a set of training data at each update step. To prevent any class bias, mini-batches are formed as an integer multiple of 25 and a uniform class distribution is imposed; in this work, we use a batch size of 150, or 6 instances of each class per mini-batch. We split the five folds into 3–1–1 for training, validation, and test, respectively, and introduce an early-stopping criterion by computing the classification error over the entire validation set every 50 iterations. Every training run is conducted for 6000 iterations, and the weights that produce the best validation score (*best*) are stored for subsequent evaluation.

### III. METHODOLOGY

#### A. Input Representation

Though theoretically conceivable that a convolutional architecture could be applied to raw, time-domain audio, the system is simplified by using musical knowledge to produce perceptually motivated pitch spectra via the constant-Q transform. Transforming audio signals to time-frequency representations provides the dual benefits of a lower dimensional input than raw audio and is linear in pitch, allowing kernels to translate over an input. Importantly, this filterbank front-end can be interpreted as hard-coding the first layer of a larger convolutional architecture.

We implement the constant-Q transform as a time-domain filterbank of complex-valued Gabor filters, spaced at 36 filters per octave and tuned such that the duration of each filter corresponds to 40 cycles of the center frequency, yielding 252 filters spanning 27.5–1760Hz. Audio signals are first downsampled to 7040Hz and the filterbank is applied to centered windows at a frame rate of 40Hz. This over-sampled pitch spectra is then reduced to a frame rate of 4Hz by mean-filtering each subband with a 15-point window and decimating in time by a factor of 10.

There are two additional data manipulations worth mentioning here that we refer to collectively as extended training data (ETD). First, the linearity of pitch in a constant-Q representation affords the ability to “transpose” an observation as if it were a true data point in a different pitch class by literally shifting the pitch tile and changing the label accordingly. In other words, every data point can count toward each chord class of the same mode (Major or minor), effectively increasing the amount of training data by a factor of 12. Another preliminary processing stage, popularized in computer vision, is the application of subtractive-divisive contrast normalization [5], which serves as a frequency-varying gain control.

#### B. Experimental Objectives

One reality faced in the development of any moderately novel system is that there is minimal precedent to guide architectural design decisions. That being said, the classic consideration in neural network research is that of model complexity, which manifests here in two main ways: the number and the dimensionality of the kernels. Detailed in Table I, we define three kernel quantities and two kernel shapes, for a total of six architectures. Convolutional layers are described by a kernel tensor  $K$  of shape  $(N_{in}, N_{out}, \text{time}, \text{frequency})$ , and an optional pooling operation  $P$  of shape  $(\text{time}, \text{frequency})$ . Fully-connected layers are described by a single weight matrix  $W$  of  $(N_{in}, N_{out})$ .

Model complexity is of particular interest, given the size of the dataset – which is modest compared to other fields and applications – and only a limited sense of how

Table I  
CNN ARCHITECTURES

	-1	-2
3	K:(1, 16, 6, 25), P:(1, 3)	K:(1, 16, 5, 25), P:(1, 3)
	K:(16, 20, 6, 27)	K:(16, 20, 5, 13)
	K:(20, 24, 6, 27)	K:(20, 24, 5, 13)
	W:(1440, 200)	W:(7680, 200)
	W:(200, 25)	W:(200, 25)
2	K:(1, 6, 6, 25), P:(1, 3)	K:(1, 6, 5, 25), P:(1, 3)
	K:(6, 9, 6, 27)	K:(6, 9, 5, 13)
	K:(9, 12, 6, 27)	K:(9, 12, 5, 13)
	W:(720, 125)	W:(3840, 125)
	W:(125, 25)	W:(125, 25)
1	K:(1, 4, 6, 25), P:(1, 3)	K:(1, 4, 5, 25), P:(1, 3)
	K:(4, 6, 6, 27)	K:(4, 6, 5, 13)
	K:(6, 8, 6, 27)	K:(6, 8, 5, 13)
	W:(480, 50)	W:(2560, 50)
	W:(50, 25)	W:(50, 25)

Table II  
5-FOLD RECOGNITION ACCURACY

Fold	Arch 3–1			Arch 1–1		
	Train	Valid	Test	Train	Valid	Test
1	83.2	77.6	77.8	79.6	76.9	76.8
2	83.6	78.2	76.9	80.5	77.0	76.8
3	82.0	78.1	78.3	80.0	77.2	78.2
4	83.6	78.6	76.8	80.2	78.0	75.8
5	81.7	76.5	77.7	79.5	75.9	76.8
Total	82.81	77.80	<b>77.48</b>	79.97	77.00	<b>76.87</b>

complicated the underlying task might be. In addition to simply determining the best performing configuration, there are a few questions a data-driven approach is particularly suited to address: If and how does the system over-fit the training data, and are there cases when it cannot? How important is the choice of architecture? And what are the effects of using ETD?

### IV. RESULTS & DISCUSSION

As an initial step toward assessing the outlined experimental objectives, we wish to determine both the performance ceiling of this approach and the variation between folds of the data. Informal tests indicated large-kernel architectures with ETD showed the most promise, so Arch:3–1 and Arch:1–1 were trained and evaluated across all five folds. The results of this experiment, shown in Table II, provide two important insights: CNN chord recognition performs competitively with the state of the art and performance discrepancies between folds fall within a 2% margin. Previously published numbers on this dataset fall in the upper 70% range [1], but it is encouraging that a CNN could reach

this benchmark in a first attempt. Additionally, we found that median filtering the output probability surface with a window of length 5 (approximately 1 second) before taking the frame-wise *argmax()* slightly boosted performance across the board, but never more than 1%; even so, all results presented here include this operation.

Given the performance consistency across folds and our interest in general trends rather than statistical significance, we conduct a parameter sweep using a leave-one-out (LoO) strategy in the spirit of computational efficiency. Noting that training and validation performance are lowest for the fifth fold, we select this stratification scenario to train each architecture in Table I under both True and False ETD conditions based on the premise that this split of the data is likely the most difficult. Each architecture is trained similarly as before, and the best performing weight parameters over the validation set are used for final evaluation; overall results are given in Table III.

The most noticeable result of this parameter sweep is the accuracy differential in performance on the training set between ETD conditions. Transposing the training data improves generalization, in addition to reducing the extent to which the network can over-fit the training data. This is a particularly interesting outcome because transposing the input pitch spectra should have negligible effects of the learned kernels. It is therefore reasonable to assume that over-fitting occurs in the fully connected layers of the network and not necessarily in the convolutional ones. This also raises an interesting question in the process: why can't these models over-fit the ETD condition?

Focusing our on Arch:3–1, one potential cause of over-fitting is due to an under-representation of chord classes in the dataset. Figure 3 plots the accuracy differential between ETD conditions for both training and test as a function of chord class, ranked by occurrence in the dataset, and indicates that this is not the root cause. ETD reduces over-fitting, but it does so uniformly and there is only a weak positive correlation between train-test discrepancy and ranked chord type. In other words, all chord classes benefit equally from ETD, which is more characteristic of widespread intra-class variance than some classes being inadequately represented.

If this is indeed the case, there are two main sources of intra-class variance: mapping all chord classes to Major-minor, or simple labeling error. To assess the former, Figure 4 plots the accuracy for Major-minor (Mm) versus all other (O) chords that are not strictly annotated as root-position Major-minors in the train (Tr) and test (Te) conditions, with and without ETD. This figure illustrates that a good deal of over-fitting in the training set is due to Other chord types mapped to Major-minor, which ETD reduces, while generalization performance for Other chord types is almost equal in both ETD conditions. Therefore, it is logical to conclude that ETD distributes the added variance of Other chord types across all classes evenly, and the machine learns

Table III  
PARAMETER SWEEP – RESULTS

Arch	ETD:False			ETD:True		
	Train	Valid	Test	Train	Valid	Test
1-1	84.7	74.9	75.6	79.5	75.9	76.8
1-2	87.0	73.1	74.5	78.4	75.5	76.2
2-1	85.5	75.0	75.5	80.6	75.6	77.0
2-2	91.2	73.9	74.0	79.4	75.4	76.6
3-1	92.0	75.2	75.5	81.7	76.5	77.7
3-2	91.7	73.6	73.8	81.6	76.3	77.4

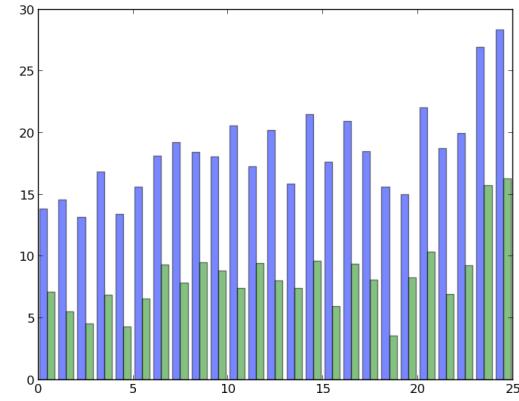


Figure 3. Accuracy differential between training and test as a function of chord class, ordered along the x-axis from most to least common in the dataset for ETD:False (blue) and ETD:True (green) conditions.

to ignore much of it as noise, while also learning a better strict Major-minor model in the process.

There is also the concern that over-fitting may be a function of specific tracks. Figure 5 shows the track-wise histograms of accuracy differential before and after ETD for the train, validation, and test datasets. Though the accuracy over most tracks is unaffected by the ETD condition, as evidenced by the near-zero mode of the distributions, certain tracks in the training set do much worse when the data is transposed. We can take this to mean that some tracks are particularly problematic, and this should be explored further in future work. This is intuitively satisfying because the repetitive nature of music would likely cause single tracks to contain multiple instances of rare chords and outliers in the dataset, and suffer the most when these few data points are not over-fit by the machine.

## V. CONCLUSIONS & FUTURE WORK

In this work, we have presented a new approach to tackling the well-worn task of automatic chord recognition. Viewing harmony as a hierarchy of intervallic relationships, we train a convolutional neural network to classify five-second tiles of pitch spectra, yielding state of the art perfor-

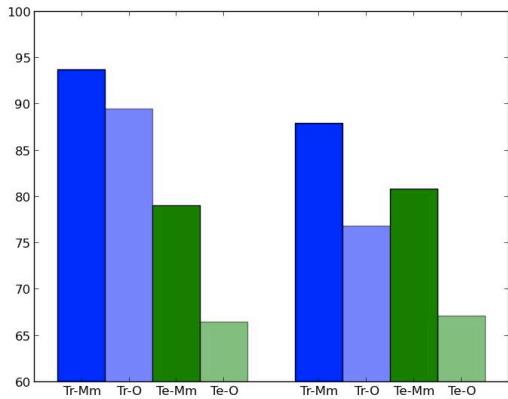


Figure 4. Effects of transposition on recognition accuracy as a function explicitly labeled Major-Minor chords (dark bars), versus mapping other chord types (lighter bars) to the nearest Major-Minor for training (blue) and test (green), for ETD:False (left) and ETD:True (right).

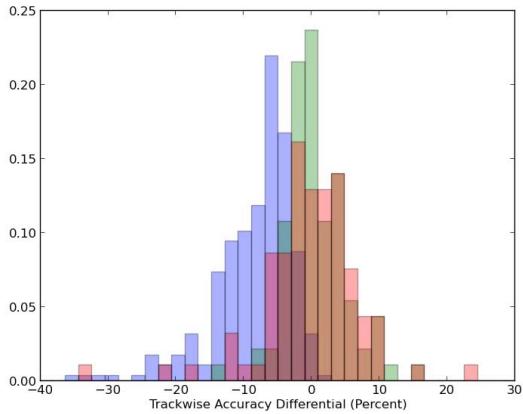


Figure 5. Histograms of trackwise accuracy differential between ETD:False and ETD:True conditions, for training (blue), validation (red) and test (green) datasets.

mance across a variety of configurations. We find that simpler architectures (Arch:1–1) perform nearly as well as over-complete ones (Arch:3–1) without any explicit regularization or weight decay penalties. When examining performance across a variety of conditions, over-fitting is used to gain insight into what and how these machines actually learn. In particular, over-fitting is due to largely to the mapping of various chord types to Major-minor and a few outlier tracks; ETD techniques reduce these effects by distributing additional intra-class variance across all chord classes, which the machine learns to mostly ignore. The inability to over-fit all data, however, suggests that a non-negligible amount may be mislabeled or mapped to Major-minor chords incorrectly.

With an eye toward future work, unsupervised pre-

training, either through convolutional Deep Belief Nets or stacked Autoencoders, may have a substantial impact on improving performance for sparsely represented chords. To this point, there is also the possibility of incorporating other work in transfer, or one-shot, learning, where new classes can be discriminated with only a few training examples. Going forward, the greatest potential of this approach will likely be realized in extending to larger, more realistic chord vocabularies.

#### ACKNOWLEDGMENT

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

#### REFERENCES

- [1] T. Cho and J. P. Bello, “A Feature Smoothing Method For Chord Recognition Using Recurrence Plots.” In *Proc. ISMIR*, 2011.
- [2] T. Cho, R. J. Weiss and J. P. Bello, “Exploring Common Variations in State of the Art Chord Recognition Systems.” In *Proc. SMC*, 2010.
- [3] T. Fujishima, “Realtime chord recognition of musical sound: a system using common lisp music,” In *Proc. ICMC*, 1999.
- [4] E. J. Humphrey, T. Cho, and J. P. Bello, “Learning a Robust Tonnetz-space Representation for Chord Recognition,” In *Proc. ICASSP*, 2011.
- [5] Y. LeCun, K. Kavukcuoglu and C. Farabet. Convolutional Networks and Applications in Vision,” In *Proc. ISCAS*, 2010.
- [6] M. Mauch and S. Dixon, “Approximate Note Transcription For The Improved Identification Of Difficult Chords.” In *Proc. ISMIR*, 2010.
- [7] M. Mauch and S. Dixon, “Simultaneous Estimation of Chords and Musical Context From Audio.” *IEEE Transactions On Audio, Speech, And Language Processing (TSALP)*, vol. 18, no. 6, pp. 1280–1289, 2010
- [8] M. Müller and S. Ewert, “Towards timbre-invariant audio features for harmony-based music,” *IEEE Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 18, no. 3, pp. 649–662, 2010.
- [9] A. Sheh and D. P. W. Ellis, “Chord segmentation and recognition using EM-trained Hidden Markov Models,” In *Proc. ISMIR*, 2003.
- [10] A. Weller, D. P. W. Ellis, and T. Jebara, “Structured Prediction Models for Chord Transcription of Music Audio.” In *Proc. ICMLA*, 2009.
- [11] A. Zils and F. Pachet. “Automatic extraction of music descriptors from acoustic signals using EDS.” In *Proc. AES*, 2004.