Chord Detection Using Deep Learning

Alexander Lerch
Georgia Institute of Technology
103 PUBLICATIONS **1,031** CITATIONS
SEE PROFILE

CHORD DETECTION USING DEEP LEARNING

Xinquan Zhou
Center for Music Technology
Georgia Institute of Technology
royzxq@gmail.com

Alexander Lerch
Center for Music Technology
Georgia Institute of Technology
alexander.lerch@gatech.edu

ABSTRACT

In this paper, we utilize deep learning to learn high-level features for audio chord detection. The learned features, obtained by a deep network in bottleneck architecture, give promising results and outperform state-of-the-art systems. We present and evaluate the results for various methods and configurations, including input pre-processing, a bottleneck architecture, and SVMs vs. HMMs for chord classification.

1. INTRODUCTION

The goal of automatic chord detection is the automatic recognition of the chord progression in a music recording. It is an important task in the analysis of western music and music transcription in general, and it can contribute to applications such as key detection, structural segmentation, music similarity measures, and other semantic analysis tasks. Despite early successes in chord detection by using pitch chroma features [6] and Hidden Markov Models (HMMs) [26], recent attempts at further increasing the detection accuracy are only met with moderate success [4,28].

In recent years, deep learning approaches have gained significant interest in the machine learning community as a way of building hierarchical representations from large amounts of data. Deep learning has been applied successfully in various fields; for instance, a system for speech recognition utilizing deep learning was able to outperform state-of-the-art systems not using deep learning [10]. Several studies indicate that deep learning methods can be very successful when applied to Music Information Retrieval (MIR) tasks, especially when used for feature learning [1,9,13,16]. Deep learning, with its potential to untangle complicated patterns in a large amount of data, should be well suited for the task of chord detection.

In this work, we investigate Deep Networks (DNs) for learning high-level and more representative features in the context of chord detection, effectively replacing the widely used pitch chroma intermediate representation. We present individual results for different pre-processing options such as time splicing and filtering (see Sect. 3.2), architectures (see Sect. 3.4), and output classifiers (see Sect. 4).

2. RELATED WORK

During the past decade, deep learning has been considered by the machine learning community to be one of the most interesting and intriguing research topics. Deep architectures promise to remove the necessity of custom-designed and manually selected features as neural networks should be more powerful in disentangling interacting factors and thus be able to create meaningful high-level representations of the input data. Generally speaking, deep learning combines deep neural networks with an unsupervised learning model. Two major learning models are widely used for unsupervised learning: Restricted Boltzmann Machines (RBMs) [11] and Sparse Auto Encoders [24]. A deep architecture comprises multiple stacked layers based on one of these two models. These layers can be trained one by one, a process that is referred to as "pre-training" the network. In this work, we employ RBMs to pre-train the deep architecture in an unsupervised fashion; this is called a Deep Belief Network (DBN) [11]. DBNs, composed of a stack of RBMs, essentially share the same topology with general neural networks: DBNs are generative probabilistic models with one visible layer and several hidden layers.

Since Hinton et al. proposed a fast learning algorithm for DBNs [11], it has been widely used for initializing deep neural networks. In deep structures, each layer learns relationships between units in lower layers. The complexity of the system increases with an increasing number of RBM layers, making the structure —in theory— more powerful. An extra softmax output layer can be added to the top of the network (see Eqn (6)) [18]; its output can be interpreted as the likelihood of each class.

LeCun and Bengio introduced the idea of applying Convolutional Neural Networks (CNNs) to images, speech, and other time-series signals [15]. This approach allows to deal with the variability in time and space to a certain degree, as CNNs can be seen as a special type of neural network in which the weights are shared across the input within a certain spatial or temporal area. The weights thus act as a kernel filter applied to the input. CNNs have been particularly successful in image analysis. For example, Norouzi et al. used Convolutional RBMs to learn shift-invariant features [22].

The results of a network depend largely on the network architecture. For example, Grezl et al. used a so-called bottleneck architecture neural network to obtain features for speech recognition and showed that these features improve the accuracy of the task [8]. The principle behind

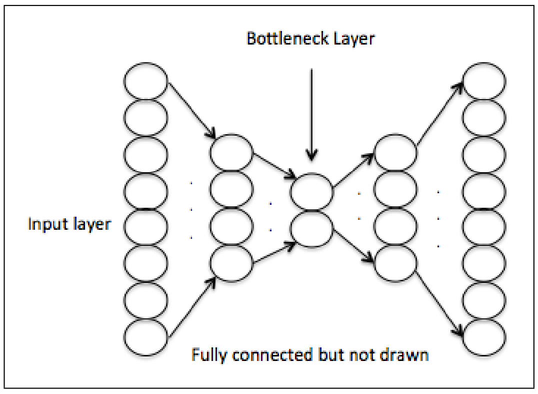

Figure 1. Visualization of a bottleneck architecture

the bottleneck-shaped architecture is that the number of neurons in the middle layer is lower than in the other layers as shown in Fig. 1. A network with bottleneck can be structured in two sections: (i) Section 1 from the first layer to the bottleneck layer, with a gradual decrease of the number of neurons per layer, functions as an encoding or compression process which compacts relevant information and discards redundant information, and (ii) Section 2 from the bottleneck layer to the last layer with a gradual increase in the number of neurons per layer. The function of this part can be interpreted as a decoding process. An additional benefit of bottleneck architectures is that they can reduce overfitting by decreasing the system complexity.

Recently, more researchers investigated deep learning in the context of MIR. Lee et al. pioneered the application of convolutional deep learning for audio feature learning [16]. Hamel et al. used the features learned from music with a DBN for both music genre classification and music autotagging [9]; their system was successful in MIREX 2011 with top-ranked results. Battenberg employed a conditional DBN to analyze drum patterns [1]. The use of deep architectures for chord detections, however, has not yet been explored, although modern neural networks have been employed in this field. For instance, Boulanger et al. investigated recurrent neural networks [2] and Humphrey has explored CNNs [12, 14]. While they also used the concept of pre-training, their architectures have only two or 3 layers and thus cannot be called "deep".

The basic buildings blocks of most modern approaches to chord detection can be traced back to two seminal publications: Fujishima introduced pitch chroma vectors extracted from the audio as input feature for chord detection [6] and Sheh et al. proposed to use HMMs for representing chords as hidden states and to model the transition probability of chords [26]. Since then, there have been a lot of studies using chroma features and HMMs for chord detection [5, 23]. Examples for recent systems are Ni et al., using a genre-independent chord estimation method based on HMM and chroma features [21] and Cho and Bello, who used multi-band features and a multi-stream HMM for chord recognition [4]. Training HMMs with pitch chroma features arguably is the standard approach for this task and the progress is less marked by major innovations but by

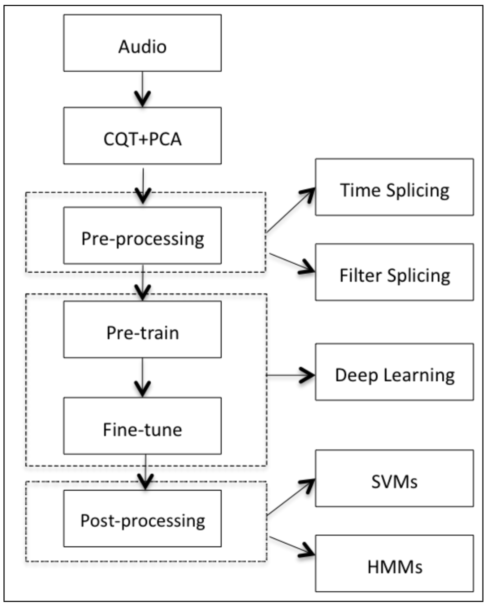

Figure 2. The overview of our system

optimizing and tuning specific components.

3. SYSTEM OVERVIEW

Figure 2 gives an overview of all components and processing steps of the presented system. The following section will discuss all of these steps in detail.

3.1 Input Representation

The input audio is converted to a sample rate of 11.025 kHz. Then, a Constant Q transform (CQT) is applied. The CQT [3] is a perceptually inspired time-frequency transformation for audio. The resulting frequency bins are equally spaced on a logarithmic ("pitch") scale. It has the advantage of providing a more musically and perceptually meaningful spectral representation than the DFT. We used an implementation of the CQT as a filterbank of Gabor filters, spaced at 36 bins per octave, i.e., 3 bins per semitone, yielding 180 bins representing a frequency range spanning from 110 Hz to 3.520 kHz. Finally, we used Principal Component Analysis (PCA) for decorrelation, and applied Z-Score normalization [27].

3.2 Pre-processing

Neighboring frames of the input representation can be expected to contain similar content, as chords will not change on a frame-by-frame basis. In order to take into account the relationship between the current frame and previous and future frames, we investigate the application of several pre-processing approaches.

3.2.1 Time Splicing

Time splicing is a simple way to extend the current frame with the data of neighboring frames by concatenating the frames into one larger superframe. In first order time splicing, we concatenate the current frame, the previous frame, and the following frame. Thus, each superframe consists of three neighboring frames. Since the same operation will be applied to all frames, there will be overlap introduced between neighboring superframes.

3.2.2 Convolution

CNNs are extensively used in tasks with highly correlated inputs (e.g., the recognition of hand-written digits). Many time series show similar properties so that CNNs seem to be an appropriate choice in the context of audio, too. Essentially, CNNs have one or more convolutional layers between the input and lower layers of the neural network. The function of a convolutional layer can be interpreted as the application of a linear filter plus a non-linear transformation, sometimes also combined with a pooling operation:

$$Y = \mathrm{pool}(\mathrm{sigm}(K * X + B)), \qquad (1)$$

in which Y is the output of a convolutional layer, K is the linear kernel filter (i.e., the impulse response), X is the input, B is the bias, $\mathrm{sigm}()$ is a non-linear transform, and $\mathrm{pool}()$ is a down-sampling operation. The uniqueness of convolutional networks stems from the convolution operation applied to the input X. Since, unfortunately, we had no access to a deep learning toolbox with support for the convolution operation in the time domain, we opted to employ an optional pre-processing step inspired by CNNs, namely by applying filters to the input of the network. However, instead of learning the filters, we evaluate several manually designed filters: a single-pole low pass filter and two FIR low pass filters with exponentially shaped impulse responses. The single pole low pass filter produces the output y for an input x, given the parameter α:

$$y_n = (1-\alpha)y_{n-1} + \alpha x_n \qquad (2)$$

We apply anti-causal filtering and filter the signal in both directions so that the resulting overall filter has a zero-phase response.

The other two low pass filters have exponential decay shaped impulse response. The difference equations are given in Eqn (3) and Eqn (4).

$$y_1(n) = \sum_{k=1}^{N} a^{-k+1} x(n-N+k) \qquad (3)$$

$$y_2(n) = \sum_{k=1}^{N} a^{-k+1} x(n+N-k) \qquad (4)$$

The filter length is N and a is the exponential base. These two filters are not centered around the current frame anymore but shifted by N frames. Their impulse responses are symmetric to each other. One could interpret these filters as focusing on past and future frames, respectively. The presented filters will be referred to as "extension filters".

The ideas of splicing and convolution can be combined, as exemplified in Fig. 3.

Furthermore, similar to the process in CNNs, a maximum pooling operation on the output of the spliced filters is optionally applied. The operation takes the maximum value among different filters per "bin".

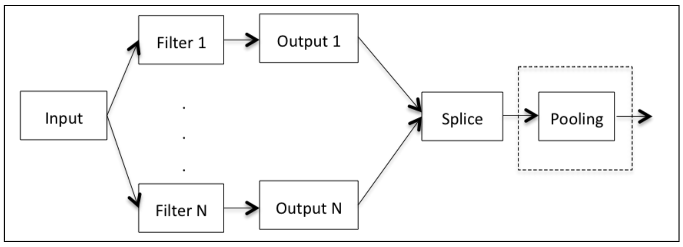

Figure 3. Splicing output of different filters

3.3 Training

It is impractical to train DNNs directly with back propagation using gradient decent due to their deep structure and the limited amount of training samples. Therefore, the network is usually initialized by an unsupervised pre-training step. As our network consists of RBMs, Gibbs sampling can be used for training [11]. The objective is to retain as much information as possible between input and output.

The computation for layer l can be represented as:

$$Y_l = \mathrm{sigm}(W_l X_l + B_l), \qquad (5)$$

which is identical to many traditional neural networks. Thus, a standard back propagation can be applied after pre-training to fine-tune the network in a supervised manner. The loss criterion we use in this work is cross-entropy.

3.4 Architecture

We investigate a deep network with 6 layers in two different architectures. The common architecture features the same amount of neurons in every layer, in our case 1024. The bottleneck architecture has 256 neurons in the middle layer and 512 neurons in the layers neighboring the middle layer. The remaining layers consist 1024 neurons each (compare [8]). A softmax output layer is stacked on top of both architectures as described by Eqn (6).

$$\mathrm{softmax}(Y_l) = \frac{\exp(Y_l)}{\sum_{k=1}^{N} \exp(Y_k)} \qquad (6)$$

The network is implemented using the Kaldi package developed by John Hopkins University [25].

4. CLASSIFICATION

The output of the softmax layer can be interpreted as the likelihood of each chord class; simply taking the maximum will provide a class decision (this method will be referred to as *Argmax*). Alternatively, the output can be treated as intermediate feature vector that can be used as an input to other classifiers for computing the final decision.

4.1 Support Vector Machine

Support Vector Machines (SVMs) are, as widely used classifiers with generally good performance. The SVM is trained using the output of the network as features, and the classification is carried out frame by frame. The classification is followed by a simple prediction smoothing.

4.2 Hidden Markov Model

HMMs are, as pointed out above, the standard classifier for automatic chord detection because the characteristics of the task fit the HMM approach well: Chords are hidden states that can be estimated from observations (feature vectors extracted from the audio signal), and the likelihood of chord transitions can be modeled with transition probabilities. Modified HMMs such as ergodic HMMs and key-independent HMMs have been also explored for this task [17, 23]. In this work we are mostly interested in the performance comparison between high-level features, so a simple first-order HMM is used. Given the probabilistic characteristic of the softmax output layer, it can be directly as emission probabilities for the HMM. Therefore, there is no need to train the HMM using, e.g., the commonly used Baum-Welch algorithm. Instead, the histogram of each class in our training is used as initial probabilities, and the bigram of chord transitions is used to compute the transition probabilities. Finally, we employ the Viterbi decoding algorithm to find the globally optimal chord sequence.

5. EVALUATION PROCEDURE

5.1 Dataset

Our dataset is a combination of several different datasets, yielding a 317-piece collection. The data is composed of

- 180 songs from the *Beatles dataset* [19],
- 100 songs from the *RWC Pop dataset* [7],
- 18 songs from the *Zweieck dataset* [19], and
- 19 songs from *Queen dataset* [19].

The pre-processing as described in Sect. 3.2 ensures identical input audio formats.

5.2 Methodology

The dataset is divided randomly into two parts: 80% for the training set and 20% for the test set. On the training scale, we use a frame-based strategy, which means we divide each song into frames, and treat each frame as an independent training sample. On average each song is divided into about 1200 frames resulting in approximately 300k training samples and approximately 76k test samples.

Within the training set, 10% of the data is used as a validation set. For the post-processing, all data in the training set will be used to train the post-classifier.

Time constraints and the workload requirements for training deep networks made a cross validation for evaluation impractical.

The chosen ground truth for classification are major and minor triads for every root note, resulting in a dictionary of $24 + 1$ chord labels. Ground truth time-aligned chord symbols are mapped to this major/minor dictionary:

$$Chord_{majmin} \subset \{N\} \cup \{S \times maj, min\} \tag{7}$$

with S representing the 12 pitch classes (root notes) and N being the label for unknown chords. In the calculation of the detection accuracy, the following chord types are mapped to the corresponding major/minor in the dictionary:

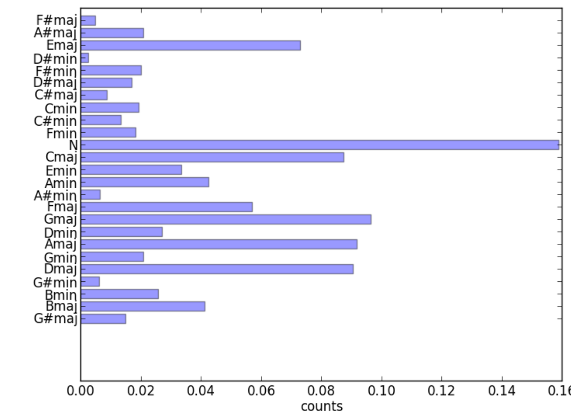

Figure 4. Chords histogram

triad major/minor and seventh major/minor. Other chord types are treated as unknown chords. For instance, G:maj and G:maj7 are mapped to 'G:maj'; G:dim and G:6 are all mapped to 'N'. The histogram of chords in our dataset after such mapping is shown in Fig. 4.

5.3 Evaluation Metric

The used evaluation metric is the same as proposed in the audio chord detection task for MIREX 2013: the Weighted Chord Symbol Recall (WCSR). WCSR is defined as the total duration of segments with correct prediction as formulated in Eqn (8):

$$WCSR = \frac{1}{N} \sum_{k=1}^{n} C_k, \tag{8}$$

in which n is the number of test samples (songs), N is the total number of frames in all test samples, and C_k is the number of frames that are correctly detected in the kth sample.

6. EXPERIMENTS

6.1 Post-classifiers

In this experiment, the network is initialized with pre-training, followed by fine tuning using back propagation. This configuration will be referred to as $DN_{DBN-DNN}$. No pre-processing is applied to the data; the input is simply the input representation (CQT followed by PCA) as described in Sect. 3.1. The chosen architecture is the bottleneck architecture. Three different classifiers are compared: the maximum of the softmax output (Argmax), an SVM, and an HMM.

The results listed in Table 1 are unambiguous and unsurprising: the HMM with Viterbi decoding outperforms the SVM; using HMMs with a model for transition probabilities is an appropriate approach to chord detection as it models the dynamic properties of chord progressions, which cannot be done with non-dynamic classifiers such as SVMs. One noteworthy result is that the SVM does not

Training Scenario	**Classifier**	**WCSR**
$DN_{\mathrm{DBN-DNN}}$	Argmax()	0.648
$DN_{\mathrm{DBN-DNN}}$	SVM	0.645
$DN_{\mathrm{DBN-DNN}}$	HMM	**0.755**

Table 1. Chord detection performance using different post-classifiers

α	Pre-processing	WCSR
0.25	Filtering	0.758
0.25	Spliced Filters	0.912
0.5	Filtering	0.787
0.5	Spliced Filters	0.857
0.75	Filtering	0.798
0.75	Spliced Filters	**0.919**

Table 2. Chord detection performance using different filter parameters

Architecture	**Pre-processing**	**Training WCSR**	**WCSR**
Common	None	0.843	0.703
Bottleneck	None	0.855	0.755
Common	Spliced Filters	**0.985**	0.876
Bottleneck	Spliced Filters	0.936	**0.919**
Common	Pooling	0.965	0.875
Bottleneck	Pooling	0.960	0.916

Table 3. Chord detection performance for different architectures and pre-processing steps

Learning Targets	**WCSR**
Single-Label — 25 Chord Classes	**0.919**
Multi-Label — 12 Pitch Classes	0.78

Table 4. Chord detection performance for single-label vs. multi-label learning

improve the WCSR compared to the direct (Argmax) output of the network. Apparently, the SVM is not able to improve separability of the learned output features.

6.2 Pre-processing

As stated in Sect. 3.2, we are interested in the application of different filters in the pre-processing stage. In the first experiment (*Filtering*), an anti-causal single pole filter (see Eqn (2)) is evaluated with the parameter α set to 0.25, 0.5, and 0.75, respectively. The second experiment (*Spliced Filters*), splices these filter outputs with the outputs of the extension filters as introduced in Sect. 3.2. These experiments are carried out with the $DN_{\mathrm{DBN-DNN}}$ training scenario, a bottleneck architecture, and an HMM classifier. Table 2 lists the results of these pre-processing variants. It can be observed that the network trained with filtered inputs slightly outperforms the network without pre-processing; splicing the filtered input with the extension filter outputs increases the results drastically.

6.3 Architecture

6.3.1 Common vs. Bottleneck

The results of Grezl et al. indicate that a bottleneck architecture should be more suitable to learn high-level features than a common architecture and reduce overfitting [8]. In order to verify these characteristics for our task, the performance of both architectures is evaluated in comparison. The results are listed in Table 3 for three pre-processing scenarios: no additional pre-processing (*None*), *Spliced Filters* and spliced filters followed by a max pooling (*Pooling*). In order to allow conclusions about overfitting, both the WCSR of the test set and the training set are reported. All results are computed for the $DN_{\mathrm{DBN-DNN}}$ training scenario with HMM classifiers.

The results show that the bottleneck architecture gives significantly better results ($p = 0.023$) on the test set (*WSCR*). Note that this is not true for the training set (*Training WSCR*), for which the common architecture achieves results in the same range or better than the bottleneck architecture. The difference between the results on the training set and the test set are thus much larger for the common architecture than for the bottleneck architecture. The bottleneck architecture is clearly advantageous to use in this task: it reduces complexity and thus the training workload and increases the classification performance significantly. Furthermore, the comparison of classifier performance between training and test set in Table 3 clearly indicates that the common architecture tends to fit more to the training data, and is thus more prone to overfitting.

6.3.2 Single-Label vs. Multi-Label

As mentioned above, the pitch chroma is the standard feature representation for audio chord detection. Since we use the output of our deep network as feature, it seems an intuitive choice to learn pitch class information (and thus, a pitch chroma) instead of the chord classes. By doing so, the number of outputs is reduced by a factor of two (or higher in the case of more chords), and there would also be a closer relation between the output and the input representation, the CQT. Therefore, the abstraction and complexity of the task might be decreased. It will, however, lead to another issue: the single-label output (one chord per output) will be changed into a multi-label output (multiple pitches per output). Therefore, the learning has to be modified to allow multiple simultaneous (pitch class) labels. The experiment is carried out with both Splicing and Filtering in the pre-processing, the $DN_{\mathrm{DBN-DNN}}$ training scenario, and HMM classifiers. Table 4 lists the results.

Boulanger-Lewandowski et al. report combining chroma features with chord labels for their recurrent neural network and report a slightly improved result [2]. They do not, however, provide a detailed description of this combination. As can be seen from the table, the result for multi-label training is clearly lower than the result for single-label training.

Method	WCSR
Chordino	0.625
Best Configuration	0.919
Best Configuration with Max Pooling	0.916

Table 5. Comparison of the performance of the best configuration with Chordino

Possible reasons for bad performance include (i) difficulties with multi-target learning, since it increases the difficulty to train; furthermore, our implementation of multi-label training might be sub-optimal as the same posterior is assigned to each target without any information on the pitch class energy, and (ii) the issue that not all pitches always sound simultaneously in a chord (or might be missing altogether) might have larger impact on the multi-label training than on the single-label training.

6.4 Results & Discussion

It is challenging to compare the results to previously published results due to varying evaluation methodologies, metrics, and datasets. It seems that the results of Cho and Bello [4], who reported a performance of about 76%, were computed with a comparable dataset. The recent MIREX results on Chord Detection generally show lower accuracy but use a different evaluation vocabulary. In order to provide a baseline result to put results into perspective, we present the results of Chordino [20] with the default settings, computed on our dataset. It should be pointed out that this comparison is unfair as Chordino is able to detect as many as 120 chords, compared to our 24. The label mapping strategies are another significant issue for Chordino. Our label mapping results in nearly sixth of the total label being "N", which might have negative impact on the Chordino results. The Chordino results are mapped to major/minor the same way as the ground truth annotations. The results are shown in Table 5. In the table, the *Best Configuration* is using Bottleneck architecture, spliced filters ($\alpha = 0.75$) as preprocessing, single label learning targets, and Viterbi decoding as post-classifier. The *Best Configuration with Max Pooling* is the same as the best configuration except applying another max pooling layer after the spliced filters. The latter configuration has a much reduced computational workload. The presented results are clearly competitive with existing state-of-the-art systems.

7. CONCLUSION & FUTURE WORK

In this work, we presented a system which applies deep learning to the MIR task of automatic chord detection. Our model is able to learn high-level probabilistic representations for chords across various configurations. We have shown that the use of a bottleneck architecture is advantageous as it reduces overfitting and increases classifier performance, and that the choice of appropriate input filtering and splicing can significantly increase classifier performance.

Learning a pitch class vector instead of chord likelihood by incorporating multi-label learning proved to be less successful. The idea has, however, a certain appeal and would allow the number of output nodes to be independent of the number of chords to be detected. It is also conceivable to investigate a different option for the network output: instead of training chords or pitch classes we could — under the assumption that we are only after chords comprised of stacked third intervals — train the output with octave-independent third intervals in a multi-label scenario with 24 output nodes.

8. REFERENCES

[1] Eric Battenberg and David Wessel. Analyzing drum patterns using conditional deep belief networks. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 37–42, 2012.

[2] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Audio chord recognition with recurrent neural networks. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 335–340, 2013.

[3] Judith C Brown. Calculation of a constant q spectral transform. *The Journal of the Acoustical Society of America*, 89(1):425–434, 1991.

[4] Taemin Cho and Juan P Bello. Mirex 2013: Large vocabulary chord recognition system using multi-band features and a multi-stream HMM. *Music Information Retrieval Evaluation eXchange (MIREX)*, 2013.

[5] Taemin Cho, Ron J Weiss, and Juan Pablo Bello. Exploring common variations in state of the art chord recognition systems. In *Proceedings of the Sound and Music Computing Conference (SMC)*, pages 1–8, 2010.

[6] Takuya Fujishima. Realtime chord recognition of musical sound: A system using common lisp music. In *Proceedings of the International Computer Music Conference (ICMC)*, volume 1999, pages 464–467, 1999.

[7] Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. Rwc music database: Popular, classical and jazz music databases. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, volume 2, pages 287–288, 2002.

[8] Frantisek Grezl, Martin Karafiát, Stanislav Kontár, and J Cernocky. Probabilistic and bottle-neck features for lvcsr of meetings. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 4, pages IV–757. IEEE, 2007.

[9] Philippe Hamel and Douglas Eck. Learning features from music audio with deep belief networks. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 339–344. Utrecht, The Netherlands, 2010.

[10] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97, 2012.

[11] Geoffrey Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[12] Eric J Humphrey and Juan Pablo Bello. Rethinking automatic chord recognition with convolutional neural networks. In *Proceedings of the International Conference on Machine Learning and Applications (ICMLA)*, volume 2, pages 357–362. IEEE, 2012.

[13] Eric J Humphrey, Juan Pablo Bello, and Yann LeCun. Moving beyond feature design: Deep architectures and automatic feature learning in music informatics. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 403–408, 2012.

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

[15] Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361:310, 1995.

[16] Honglak Lee, Peter Pham, Yan Largman, and Andrew Y Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in neural information processing systems*, 2009.

[17] Kyogu Lee and Malcolm Slaney. Acoustic chord transcription and key extraction from audio using key-dependent HMMs trained on synthesized audio. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(2):291–301, 2008.

[18] Thomas M Martinetz, Stanislav G Berkovich, and Klaus J Schulten. Neural-gas' network for vector quantization and its application to time-series prediction. *Neural Networks, IEEE Transactions on*, 4(4):558–569, 1993.

[19] Matthias Mauch, Chris Cannam, Matthew Davies, Simon Dixon, Christopher Harte, Sefki Kolozali, Dan Tidhar, and Mark Sandler. Omras2 metadata project 2009. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, 2009.

[20] Matthias Mauch and Simon Dixon. Approximate note transcription for the improved identification of difficult chords. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 135–140, 2010.

[21] Yizhao Ni, Matt McVicar, Raul Santos-Rodriguez, and Tijl De Bie. Using hyper-genre training to explore genre information for automatic chord estimation. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 109–114, 2012.

[22] Mohammad Norouzi, Mani Ranjbar, and Greg Mori. Stacks of convolutional restricted boltzmann machines for shift-invariant feature learning. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2735–2742. IEEE, 2009.

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

[24] Christopher Poultney, Sumit Chopra, Yann L Cun, et al. Efficient learning of sparse representations with an energy-based model. In *Advances in neural information processing systems*, pages 1137–1144, 2006.

[25] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The kaldi speech recognition toolkit. In *Proceedings of the Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, December 2011. IEEE Catalog No.: CFP11SRW-USB.

[26] Alexander Sheh and Daniel PW Ellis. Chord segmentation and recognition using em-trained hidden markov models. *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 185–191, 2003.

[27] J Sola and J Sevilla. Importance of input data normalization for the application of neural networks to complex industrial problems. *Nuclear Science, IEEE Transactions on*, 44(3):1464–1468, 1997.

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