Mid-level Features for Audio Chord Estimation using Stacked Denoising Autoencoders

Nikolay Glazyrin

Ural Federal University,
Ekaterinburg, Russian Federation
nglazyrin@gmail.com

Abstract. Deep neural networks composed of several pre-trained layers have been successfully applied to various tasks related to audio processing. Stacked denoising autoencoders represent one type of such networks. They are discussed in this paper in application to audio feature extraction for audio chord estimation task. The features obtained from audio spectrogram with the help of autoencoders can be used instead of conventional chroma vectors to estimate the actual chords in the audio recording.

Keywords: Audio chord estimation, autoencoder, recurrent network, deep learning.

1 Introduction

One of important areas of music information retrieval is the estimation of elements related to musical concepts. The concepts such as note, chord, melody, key can be determined from musical score, but their extraction from an audio signal is a very difficult task even for the human.

The ultimate goal of estimating automatically all musical concepts of a given musical audio, or obtaining its transcription, can hardly be solved. Instead, many researchers work on different aspects of this task, such as audio melody extraction, beat detection or audio chord estimation. The system that can estimate chords in a given audio recording will be of great value for those who want to play a favourite song on a guitar: a rare song may be missing in the online sources of guitar chords, or its chords may be incorrect or may describe a different version of the song. For a musicologist who analyzes the harmony of a song such a system may provide a good starting point by recognizing most part of chords and letting him only fix recognition errors in difficult cases. The information about the sequence of chords in a musical audio may also be used by other algorithms. For instance, an index can be built that allows for searching the song by a chord sequence.

More formally, the task of audio chord estimation consists in following. Given a digital musical audio recording one needs to determine the sequence of chords that were played in this recording with their boundaries: for each chord the name and the times of its beginning and end must be specified. Chord names are usually chosen from a fixed set of 12 major and 12 minor chords plus a N symbol for a missing chord (e.g. when no pitched musical instrument is playing or when there is silence).

During chord estimation various properties of music are exploited, such as the presence of rhythm, the presence of harmonic frequencies of pitched instruments, repeated sections, polyphony. Sometimes audio recordings contain noises, unpitched musical instruments, a voice singing out-of-tune, which make it harder to recognize the chord.

A common approach to this task can be divided into 3 steps. At first, the audio recording is pre-processed (beat detection is often performed here) and transformed to the time-frequency domain (using fast Fourier transform or constant-Q transform [2]), resulting in a so-called spectrogram. It shows how the spectrum of this recording is changing in time. Each spectrogram column represents the spectrum of a short fragment of the original recording. If beat detection was performed, fragment boundaries correspond to beat locations.

Then a series of transforms is applied to the spectrogram to take into account music properties mentioned above and obtain the sequence of feature vectors (e.g. pitch class profile vectors [7], chroma DCT-reduced log-pitch [16]). Most of them are generally called *chroma* vectors, because they have 12 components, one component per pitch class (which unites all frequencies that correspond to notes with the same name from different octaves). Each feature vector is of smaller dimensionality than the corresponding spectrogram column.

Finally, the sequence of feature vectors is converted to the sequence of chord symbols. Chord boundaries appear naturally, because of initial separation of the recording into a set of fragments with fixed boundaries. Probabilistic models such as hidden Markov models (e.g. [12]) and dynamic Bayesian networks [14] are often used here because of their ability to model series of sequential events and include various factors (bass note, musical key). The target sequence of chord labels can be obtained from the sequence of observed feature vectors by application of the Viterbi algorithm to the model.

Simpler algorithms that only compare the feature vectors with predefined template vectors for chords [18] perform a little worse, but do not require training and therefore cannot be overfitted to concrete chord sequences or music style. We believe that a good chord recognition quality can be achieved without probabilistic models, using better features with simpler classifier instead. Promising results in this direction were obtained in [10] with the help of deep convolutional neural network. In this paper we investigate stacked denoising autoencoders (including recurrent ones), which were applied successuflly to automatic speech recognition [15]. With the introduction of recurrent connections it is possible to model the dependency on the chord playing at the previous fragment of a recording.

The remainder of the paper is organized as follows. Section 2 introduces autoencoders and their modifications. In section 3 the experimental setup and evaluation methods are described. In section 4 the results are presented and discussed. Section 5 provides the directions for future work.

2 Autoencoders and recurrent networks

2.1 Theoretical background

The definitions here are given following [21].

An *autoencoder* (also often called autoassociator) can be represented as a pair of transforms:

$$y = f_\theta(x) = s(Wx + b) \tag{1}$$

$$z = g_{\theta'}(y) = s(W'y + b') \tag{2}$$

Here x is an input vector, z is the reconstructed output vector, y is the internal representation of x, $\theta = \{W, b\}$ and $\theta' = \{W', b'\}$ are parameters (with a typical restriction $W' = W^T$), s is a non-linear activation function, natural choice for it is the sigmoid function or the hyperbolic tangent function. In (2) a linear function s is sometimes used. A convenient representation for an autoencoder is a neural network with one hidden layer.

During autoencoder training the cost function $L(X, Z(X)))$ is minimized, where X is a set of possible inputs. A natural choice in case when X is a set of spectrogram columns is the squared error objective function $L(x, z) = ||x - z||^2$.

To avoid learning the identity mapping, the internal representation often has less dimensions than the input vector. Alternative way is to make the internal representation having more dimensions than the input vector, and introduce the sparsity restriction on the hidden layer, so that most of its activations are close to zero. In this case the internal representation becomes a sparse representation of the input. If we denote as $f_\theta^j(x)$ the activation of hidden unit j for an input x, then we can define the average activation of this hidden unit over the whole training set of size m:

$$\hat{\rho}_j = \frac{1}{m} \sum_{i=1}^{m} f_\theta^j \left(x^{(i)} \right) \tag{3}$$

We would like to enforce $\hat{\rho}_j = \rho$, where ρ is a sparsity parameter, typically close to zero, we choose $\rho = 0.05$. To achieve this, an extra penalty term L_ρ is added to the cost function L. Many choices for this term are possible, we apply the one proposed in [17]:

$$L_\rho = \beta \left[\sum_{j=1}^{h} \left(\rho \log \frac{\rho}{\hat{\rho}_j} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_j} \right) \right] \tag{4}$$

where h is the number of units in the hidden layer.

A *denoising autoencoder* is trained to reconstruct the input vector x from its corrupted version $\tilde{x}$. It was shown (see [1]) that internal representations obtained from a denoising autoencoder are more stable and robust and capture better the internal structure of the input distribution. Potential noise in the audio spectrum can be simply modelled as additive isotropic Gaussian noise $\tilde{x}|x \sim \mathcal{N}(x, \sigma^2 I)$. Other noise types can be also considered.

Denoising autoencoders can be stacked by connecting the hidden layer of one autoencoder to the input of another autoencoder. The layers are pre-trained one by one in an unsupervised manner. The values from the hidden layer of the innermost autoencoder can be used as feature vectors.

Recurrent denoising autoencoder can be constructed from the conventional denoising autoencoder by adding a recurrent connection from its hidden layer to itself (which results essentially in the Elman network [6]). Then the output of this layer is computed as

$$y(x_t) = s(Wx_t + b + Uy(x_{t-1})) \tag{5}$$

2.2 Adaptation to audio chord estimation

The output of the hidden layer depends both on current input and on the hidden representation of previous input vector. This allows modeling of dependence on the previously sounding chord. In case of stacked denoising autoencoder recurrent connection can be added at any level or even at all layers.

A shortcoming of autoencoders is the lack of an evident interpretation of those values. We could train a multiclass classifier directly on these feature vectors. But having 25 different classes a lot of examples may be needed to train the classifier properly. The task can be instead reduced to a regression task in a known feature space. For chord estimation algorithms it is more common to work with chroma features in 12-dimensional space. To convert the feature vector to a chroma vector, a logistic regression layer with 12 outputs is added on top of the autoencoders.

The whole network is then fine-tuned in a supervised manner using minibatch stochastic gradient descent with chord templates as target vectors. Binary chord templates were used for training, and a null vector was used as the template for no chord. Binary templates have 1 on the positions that correspond to notes of a chord, and 0 on the other positions. Then the no chord was detected when the element of the resulting vector with maximum absolute value is no greater than Δ – the parameter, which was adjusted empirically. In fact, autoencoders are used to pre-train the network.

Rotation of inputs or chroma vectors is a common trick when learning model parameters in chord estimation algorithms. We propose here to use rotation both when training a stacked denoising autoencoder and when testing it on unknown inputs. The input vector spanning 6 octaves can be passed through a sliding rectangle window 5 octaves wide. Each time when the window moves a semitone up, the root note of the chord moves a semitone down (with possible loss of data in the lower octave). During the training this yields in 12x growth of training examples with evenly spaced root notes. To compensate for the difference in total numbers of major and minor chords within the training set, we restricted this difference to be no more than 100 (which turns to 1200 due to rotation) in the generated set of examples for neural network training. On the other hand, chroma vectors for a new spectrogram can be calculated 12 times with 12 different roots, then rotated to the same root and averaged. This can potentially result in better recognition quality. Another option here is to use input vectors that span 5 octaves and do cyclic rotation for them. This is not natural, because highest spectral components are moved before lowest ones, and we did not use this option in the experiments.

3 Experimental setup

3.1 Pre-processing and spectrogram

A thorough description of the spectrogram calculation can be found in [8]. We will further assume that the spectrogram spans 6 octaves: from C2 (65.41 Hz) to B7 (3951 Hz) and has 1 row per note. Therefore each spectrogram column consists of 72 values that represent the intensity of the corresonding note on a given sound fragment. A logarithmic transformation is applied to each value to mimic the human perception of sound intensity: each value v is replaced with $\log_{10}(1000v + 1)$, as proposed in [16]. Each spectrogram column was normalized to fit its values into $[0, 1]$ before passing it to the neural network.

3.2 Feature vectors and neural networks

12-dimensional chroma feature vectors are obtained from the neural network trained as described above. We experimented with various network layouts. In all cases the network has 60 inputs and 12 outputs, but the number of hidden layers and their size varied across expermients. Hidden layers are pre-trained as autoencoders. Following options were considered:

– the described network with 1, 2 and 3 hidden layers and no recurrent connections (SDA);
– the described network with 1, 2 and 3 hidden layers and recurrent connection from the innermost hidden layer to itself (RSDA);
– conventional chroma features calculated without neural network: pitch class profile (PCP) [7], chroma-log-pitch (CLP) [11] and chroma DCT-reduced log pitch (CRP) [16].

3.3 Post-processing

At first, each feature vector is replaced with a linear combination of 15% of most similar vectors, where each vector's weight is its similarity to the source feature vector. Similarity is calculated as Euclidean distance between 2 vectors. This allows to take into account the repetition of music phrases. Similar technique was employed in [14], but they only preserved the diagonals in self-similarity matrix which are parallel to the main diagonal. We do not put such restriction on this matrix.

Additionally, two heuristics were implemented to correct chord detection errors of certain types. First heuristic consists in searching for all intervals where multiple chords of same root but different type are arranged next to each other, and then replacing each interval with a single chord label. This chord is chosen as the nearest one to the sum of all feature vectors within the interval. Another heuristic was introduced to fix chord sequences like (A, B, C), where 3 successive beats are marked with 3 different chords. Each similar sequence is replaced with the one of (A, A, C), (A, C, C), (B, B, C), (A, B, B) for which the sum the of distances from successive feature vectors to corresponding chord labels is minimal.

3.4 Evaluation

The experiments were conducted on the extensively used *Isophonics* dataset [13] consisting of 218 songs by *The Beatles*, *Queen* and *Zweieck*. This dataset was randomly divided into 2 groups of same size, which were treated by turns as train and test sets.

Given the reference sequence of chord labels and the estimated sequence of chord labels for a recording the evaluation was performed as follows. At first, the recording is separated into segments using all chord boundaries from both reference and estimated sequences. Then on each segment a "Mirex2010" score was calculated as described in [19]. Its value s equals to 1 on a segment when estimated and reference chord on this segment have at least 3 common notes (or at least 2 if the reference chord is an augmented or diminished chord) or are both no chord symbols. Otherwise $s = 0$. Then the *overlap ratio* is calculated for this recording as:

$$OR = \frac{\sum_{i=1}^{N_{segm}} s_i \ell_i}{\sum_{i=1}^{N_{segm}} \ell_i} \tag{6}$$

where s_i is the score on ith segment, ℓ_i is the length of ith segment, N_{segm} is the total number of segments in the recording. *Weighted average overlap ratio* is defined for the whole dataset as:

$$WAOR = \frac{\sum_{i=1}^{N_{tracks}} OR_i \cdot L_i}{\sum_{i=1}^{N_{tracks}} L_i} \tag{7}$$

where OR_i is the overlap ratio for ith recording, L_i is the total length of ith recording and N_{tracks} is the size of the dataset (218 recordings).

The values of weighted average overlap ratio calculated during the MIREX Audio Chord Estimation 2012 contest[1] are between 0.7159 and 0.8273. Average segmentation defined in [14] is not commonly used, but we also provide its values here. It characterizes the chord boundaries estimation quality.

We also performed statistical significance tests to determine, if there is a significant difference between algorithm variations. Friedman's ANOVA with post-hoc Tukey HSD is used in MIREX Audio Chord Estimation and other MIREX contests [5]. In this work the R [20] implementation [9] was employed using the code by Tal Galili[2].

[1] `http://nema.lis.illinois.edu/nema_out/mirex2012/results/ace/mrx/summary.html`

[2] `http://www.r-statistics.com/2010/02/post-hoc-analysis-for-friedmans-test-r-code)`

4 Results

The number of inputs in the network is relatively small. Therefore we only experimented with 1, 2 and 3 hidden layers. Learning rate for autoencoder pre-training and network learning were set to 0.03 and 0.01 respectively; both pre-training and network learning were run for 15 epochs. Batch size for batch gradient descent was set to 5. Standard deviance for isotropic Gaussian noise was set to 0.2, and the noise was applied to each spectral component with probability $p = 0.7$.

The first experiment is targeted at discovering the best performing network layout. Same layouts were used with and without recurrent connections. The spectrum was calculated from 6 octaves using sliding 5 octaves wide window. Both logarithmic transformation and post-processing were applied. The results are summarized in Table 1 (in parentheses are the sizes of hidden layers).

Algorithm	Overlap	Segmentation
SDA (48)	0.7104	0.7562
SDA (48, 40)	0.7154	0.7594
SDA (120, 60)	**0.7183**	0.7596
SDA (48, 40, 36)	0.7087	0.7561
RSDA (48)	0.7047	0.7553
RSDA (48, 40)	0.7102	0.7585
RSDA (120, 60)	0.7170	**0.7615**
RSDA (48, 40, 36)	0.7100	0.7579

Table 1. Chord recognition quality achieved with different neural network layouts.

All network layouts lead to very similar results. Sparse variants seem to perform a bit better. But only the differences between RSDA(48) and SDA(120, 60), RSDA(48) and RSDA(120, 60), RSDA(48) and SDA(48, 40), RSDA(48) and RSDA(48, 40), RSDA(48, 40, 36) and SDA(48, 40) were statistically significant. Sparse version SDA(120, 60) provides best average overlap value and therefore it was chosen for further experiments. Best results were obtained using networks with 2 hidden layers. Probably the input vectors have too little dimensions to provide any advantage for deeper models. We may increase the dimensionality by concatenating spectra of 3 or more sequential fragments, but short experiments in this direction did not lead to any improvement.

It can be noted that the difference between SDA and corresponding RSDA decreases when the number of units in hidden layers grows, and RSDA (48, 40, 36) even outperforms SDA (48, 40, 36).

The process of CLP features computation also involves a logarithmic transformation of the spectrum. CRP features additionally employ one more non-linear transformation – discrete cosine transform, followed by suppression of first coefficients and inverse discrete cosine transform. To understand if non-linear transformations performed by the network units can lead to better results, we trained

SDA(120, 60) network using spectral data without logarithmic transformation. We then measured the recognition quality using PCP, CLP and CRP features (which are obtained without neural network) calculated from spectrograms spanning 5 octaves on the same dataset. The results are shown in table 2.

Algorithm	Overlap	Segmentation
SDA (120, 60)	0.7183	0.7596
RSDA (120, 60)	0.7170	0.7615
SDA (120, 60) (no log)	0.6814	0.7522
RSDA (120, 60) (no log)	0.6753	0.7504
PCP	0.6830	0.7314
CLP	0.7148	0.7367
CRP	0.7382	0.7597

Table 2. The effect of non-linear transformations.

Both SDA and RSDA being learned on spectrum without logarithmic transformation show the same recognition quality as PCP features, there is no significant difference between them. With such transformation SDA and RSDA perform slightly better than corresponding CLP features, but again with no significant difference. But it seems impossible for this network to learn a transformation similar to the one that is used during CRP features computation. CRP features provide consistently better results compared to SDA/RSDA.

Algorithm	Overlap	Segmentation
SDA (96, 48)	0.7197	0.7630
SDA (120, 60)	0.7183	0.7596
CRP (4)	0.7458	0.7679
CRP (5)	0.7382	0.7597

Table 3. The effect of the input size.

Table 3 summarizes the effect of the number of octaves in spectrum on the chord recognition quality. In previous experiments all neural networks had 60 input units, and so has the SDA (120, 60) in this table. SDA (96, 48) has 48 inputs and was trained in the same way as SDA (120, 60), but using 4 octaves wide window sliding over the spectrum spanning 5 octaves. The number in parentheses for CRP features also specifies the number of octaves used to calculate them. Both neural network and CRP perform slightly better when using 4 octaves instead of 5, but not significantly.

It was already noticed in our previous work [8] and in other works (e.g. [14]) that smoothing of the sequence of feature vectors has a positive effect on

Algorithm	Overlap	Segmentation
SDA (120, 60)	0.7183	0.7596
SDA (120, 60) (without heuristics)	0.7041	0.7346
CRP	0.7382	0.7597
CRP (without heuristics)	0.7302	0.7627

Table 4. The effect of the post processing.

the chord recognition quality. So we only evaluate the effect of the heuristics introduced in section 3.3. From table 4 it can be seen that these heuristics also have a positive effect, but it is only significant for SDA (120, 60), not for CRP.

5 Conclusions and future work

In this paper we have investigated how the introduction of stacked denoising autoencoder can help to produce features that can be used for audio chord estimation. It was found that different layout and the presence of a recurrent connection layer do not influence significantly to the resulting chord recognition quality. The feautres obtained when first denoising autoencoder was sparse were the best ones. But we could not achieve the same chord recognition quality as with CRP chroma features. We have also implemented heuristics that can improve the result and can be combined with any features.

In the future we plan to investigate whether aggregating different types of features can lead to better chord recognition quality. Another option to consider is the usage of bagging technique with neural networks.

References

1. Bengio, Y., Courville, A. C., Vincent, P.: Unsupervised Feature Learning and Deep Learning: A Review and New Perspectives. *http://arxiv.org/abs/1206.5538* (2012).
2. Brown, J.C.: Calculation of a constant q spectral transform. Journal of The Acoustical Society of America, Vol. 89, No. 1, pp. 425–434 (1991).
3. Davies, M. E. P., Plumbley, M. D.: Context-Dependent Beat Tracking of Musical Audio. Trans. Audio, Speech and Lang. Proc., Vol. 15, No. 3, pp. 1009–1020 (2007).
4. Dixon, S.: Evaluation of the Audio Beat Tracking System BeatRoot. Journal of New Music Research, Vol. 36, No. 1, pp. 39–50 (2007).
5. Downie, Stephen J.: The music information retrieval evaluation exchange (2005-2007): A window into music information retrieval research. Acoustical Science and Technology, Vol. 29, No. 4, pp. 247–255 (2008).
6. Jeffrey L. Elman: Finding structure in time. Cognitive Science, Vol. 36, No. 2, pp. 179–211 (1990).
7. Fujishima, T.: Realtime Chord Recognition of Musical Sound: a System Using Common Lisp Music. Proceedings of ICMC, 1999, pp. 464–467 (1999).
8. Glazyrin, N.: On the task of digital audio chord recognition (*in russian*). Izvestia of Irkutsk state university, “Mathematics” series, 2013, Vol. 6, no 2., pp. 217 (2013).

9. Torsten Hothorn, Kurt Hornik, Mark A. van de Wiel, Achim Zeileis: Implementing a Class of Permutation Tests: The coin Package. Journal of Statistical Software, Vol. 28, no. 8, pp. 1-23 (2008).
10. E. J. Humphrey, T. Cho, and J. P. Bello: Learning a robust tonnetz-space transform for automatic chord recognition. Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-12), pp. 453–456 (2012).
11. N. Jiang, P. Grosche, V. Konz, and M. Müller: "Analyzing Chroma Feature Types for Automated Chord Recognition," *Proceedings of the AES 42nd International Conference: Semantic Audio*, pp. 285–294, 2011.
12. Khadkevich, M., Omologo, M.: Use of Hidden Markov Models and Factored Language Models for Automatic Chord Recognition. Proceedings of the 10th International Society for Music Information Retrieval Conference, Kobe, Japan, pp. 561–566 (2009).
13. Mauch, M. and Cannam, C. and Davies, M. and Dixon, S. and Harte, C. and Kolozali, S. and Tidhar, D. and Sandler, M.: OMRAS2 Metadata Project 2009. Late-breaking session at the 10th International Conference on Music Information Retrieval, Kobe, Japan (2009).
14. Mauch, M.: Automatic Chord Transcription from Audio Using Computational Models of Musical Context. PhD Thesis, University of London (2010).
15. Maas, A., Le, Q., O'Neil, T., Vinyals, O., Nguyen, P., Ng, A.: Recurrent Neural Networks for Noise Reduction in Robust ASR. Proceedings of INTERSPEECH (2012).
16. Müller, M., Ewert, S., Kreuzer, S.: Making chroma features more robust to timbre changes. Proceedings of the 2009 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 1877–1880 (2009).
17. Andrew Ng: CS294A Lecture Notes. Sparse autoencoder. `http://www.stanford.edu/class/cs294a/sparseAutoencoder.pdf`
18. Oudre, L., Grenier, Y., Févotte, C.: Template-Based Chord Recognition : Influence of the Chord Types. Proceedings of the 10th International Society for Music Information Retrieval Conference, pp. 153–158 (2009).
19. Johan Pauwels, Geoffroy Peeters: valuating automatically estimated chord sequences. Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-13), (2013).
20. R Core Team: R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria, (2012).
21. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol P.-A.: Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. The Journal of Machine Learning Research, Vol. 11, pp. 3371-3408 (2010).
22. Zhu, Y., Kankanhalli, M. S., Gao, S.: Music Key Detection for Musical Audio. Proceedings of 11th International Multimedia Modelling Conference (MMM '05), pp. 30–37 (2005).