AUDIO CHORD RECOGNITION WITH RECURRENT NEURAL NETWORKS

Nicolas Boulanger-Lewandowski, Yoshua Bengio and Pascal Vincent
Dept. IRO, Université de Montréal
Montréal, Québec, Canada H3C 3J7
{boulanni, bengioy, vincentp}@iro.umontreal.ca

ABSTRACT

In this paper, we present an audio chord recognition system based on a recurrent neural network. The audio features are obtained from a deep neural network optimized with a combination of chromagram targets and chord information, and aggregated over different time scales. Contrarily to other existing approaches, our system incorporates acoustic and musicological models under a single training objective. We devise an efficient algorithm to search for the global mode of the output distribution while taking long-term dependencies into account. The resulting method is competitive with state-of-the-art approaches on the MIREX dataset in the major/minor prediction task.

1. INTRODUCTION

Automatic recognition of chords from audio music is an active area of research in music information retrieval [16, 21]. Existing approaches are commonly based on two fundamental modules: (1) an *acoustic* model that focuses on the discriminative aspect of the audio signal, and (2) a musicological, or *language* model that attempts to describe the temporal dependencies associated with the sequence of chord labels, e.g. harmonic progression and temporal continuity. In this paper, we design a chord recognition system that combines the acoustic and language models under a unified training objective using the sequence transduction framework [7, 12]. More precisely, we introduce a probabilistic model based on a recurrent neural network that is able to learn realistic output distributions given the input, that can be trained automatically from examples of audio sequences and time-aligned chord labels.

Following recent advances in training deep neural networks [1] and its successful application to chord recognition [19], music annotation and auto-tagging [15], polyphonic music transcription [24] and speech recognition [17], we will exploit the power of deep architectures to extract features from the audio signals. This pre-processing step will ensure we feed the most discriminative features possible to our transduction network. A popular enhancement that we also employ consists in the use of multiscale aggregated features to describe context information [4, 10, 14]. We also exploit prior information [13] in the form of pitch class targets derived from chord labels, known to be a useful intermediate representation for chord recognition (e.g. [9]).

Recurrent neural networks (RNN) [26] are powerful dynamical systems that incorporate an internal memory, or *hidden state*, represented by a self-connected layer of neurons. This property makes them well suited to model temporal sequences, such as frames in a magnitude spectrogram or chord labels in a harmonic progression, by being trained to predict the output at the next time step given the previous ones. RNNs are completely general in that in principle they can describe arbitrarily complex long-term temporal dependencies, which made them very successful in music applications [5–7, 11, 23]. While RNN-based musical language models significantly surpass popular alternatives like hidden Markov models (HMM) [6] and offer a principled way to combine the acoustic and language models [7], existing inference procedures are time-consuming and suffer from various problems that make it difficult to obtain accurate predictions. In this paper, we propose an inference method similar to Viterbi decoding that preserves the predictive power of the probabilistic model, and that is both more efficient and accurate than alternatives.

The remainder of this paper is organized as follows. In Section 2, we present our feature extraction pipeline based on deep learning. In Sections 3 and 4 we introduce the recurrent neural network model and the proposed inference procedure. We describe our experiments and evaluate our method in Section 5.

2. LEARNING DEEP AUDIO FEATURES

2.1 Overview

The overall feature extraction pipeline is depicted in Figure 1. The magnitude spectrogram is first computed by the short-term Fourier transform using a 500 ms sliding Blackman window truncated at 4 kHz with hop size 64 ms and zero-padded to produce a high-resolution feature vector of length 1400 at each time step, L^2 normalized and square root compressed to reduce the dynamic range. Due to the following pre-processing steps, we found that a mel scale conversion was unnecessary at this point. We apply

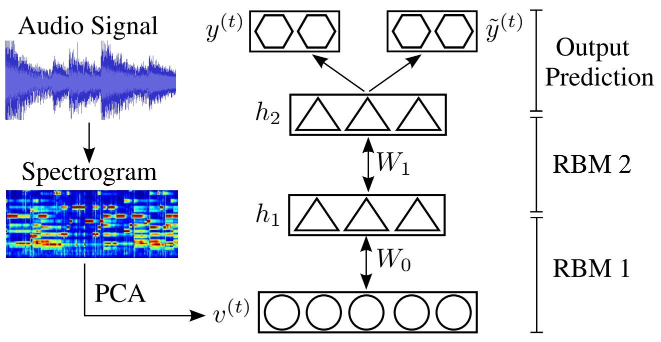

Figure 1. Pre-processing pipeline to learn deep audio features with intermediate targets $z^{(t)}, \tilde{z}^{(t)}$. Single arrows represent a deterministic function, double-ended arrows represent the hidden-visible connections of an RBM.

PCA whitening to retain 99% of the training data variance, yielding roughly 30–35% dimensionality reduction. The resulting whitened vectors $v^{(t)}$ (one at each time step) are used as input to our DBN.

2.2 Deep belief networks

The idea of deep learning is to automatically construct increasingly complex abstractions based on lower-level concepts. For example, predicting a chord label from an audio excerpt might understandably prerequire estimating active pitches, which in turn might depend on detecting peaks in the spectrogram. This hierarchy of factors is not unique to music but also appears in vision, natural language and other domains [1].

Due to the highly non-linear functions involved, deep networks are difficult to train directly by stochastic gradient descent. A successful strategy to reduce these difficulties consists in pre-training each layer successively in an unsupervised way to model the previous layer expectation. In this work, we use restricted Boltzmann machines (RBM) [27] to model the joint distribution of the previous layer's units in a deep belief network (DBN) [18] (not to be confused with a dynamic Bayesian network).

The observed vector $v^{(t)} \equiv h_0$ (input at time step t) is transformed into the hidden vector h_1, which is then fixed to obtain the hidden vector h_2, and so on in a greedy way. Layers compute their representation as:

$$h_{l+1} = \sigma(W_l h_l + b_l) \tag{1}$$

for layer l, $0 \leq l < D$ where D is the depth of the network, $\sigma(x) \equiv (1+e^{-x})^{-1}$ is the element-wise logistic sigmoid function and W_l, b_l are respectively the weight and bias parameters for layer l. The whole network is finally fine-tuned with respect to a supervised criterion such as the cross-entropy cost:

$$L(v^{(t)}, z^{(t)}) = -\sum_{j=1}^{N} z_j^{(t)} \log y_j^{(t)} + (1-z_j^{(t)}) \log(1-y_j^{(t)}) \tag{2}$$

where $y^{(t)} \equiv h_D$ is the prediction obtained at the topmost layer and $z^{(t)} \in \{0,1\}^N$ is a binary vector serving as a target at time step t. Note that in the general multi-label framework, the target $z^{(t)}$ can have multiple active elements at a given time step.

2.3 Exploiting prior information

During fine-tuning, it is possible to utilize prior information to guide optimization of the network by providing different variables, or *intermediate targets*, to be predicted at different stages of training [13]. Intermediate targets are lower-level factors that the network should learn first in order to succeed at more complex tasks. For example, chord recognition is much easier if the active pitch classes, or *chromagram targets*, are known. Note that it is straightforward to transform chord labels $z^{(t)}$ into chromagram targets $\tilde{z}^{(t)}$ and vice versa using music theory. Our strategy to encourage the network to learn this prior information is to conduct fine-tuning with respect to $\tilde{z}^{(t)}$ in a first phase then with respect to $z^{(t)}$ in a second phase, with all parameters W_l, b_l except for the last layer preserved between phases.

While a DBN trained with target $z^{(t)}$ can readily predict chord labels, we will rather use the last hidden layer $h_{D-1}^{(t)}$ as input $x^{(t)}$ to our RNN in order to take temporal information into account.

2.4 Context

We can further help the DBN to utilize temporal information by directly supplementing it with tap delays and context information. The retained strategy is to provide the network with aggregated features $\bar{x}, \tilde{x}$ [4] computed over windows of varying sizes L [14] and offsets τ relative to the current time step t:

$$\bar{x}^{(t)} = \Big\{ \sum_{\Delta t=-\lfloor L/2 \rfloor}^{\lfloor (L-1)/2 \rfloor} x^{(t-\tau+\Delta t)}, \forall (L,\tau) \Big\} \tag{3}$$

$$\tilde{x}^{(t)} = \Big\{ \sum_{\Delta t=-\lfloor L/2 \rfloor}^{\lfloor (L-1)/2 \rfloor} (x^{(t-\tau+\Delta t)} - \bar{x}_{L,\tau}^{(t)})^2, \forall (L,\tau) \Big\} \tag{4}$$

for mean and variance pooling, where the sums are taken element-wise and the resulting vectors concatenated, and L, τ are taken from a predefined list that optionally contains the original input ($L = 1, \tau = 0$). This strategy is applicable to frame-level classifiers such as the last layer of a DBN, and will enable fair comparisons with temporal models.

3. RECURRENT NEURAL NETWORKS

3.1 Definition

The RNN formally defines the conditional distribution of the output z given the input x:

$$P(z|x) = \prod_{t=1}^{T} P(z^{(t)}|\mathcal{A}^{(t)}) \tag{5}$$

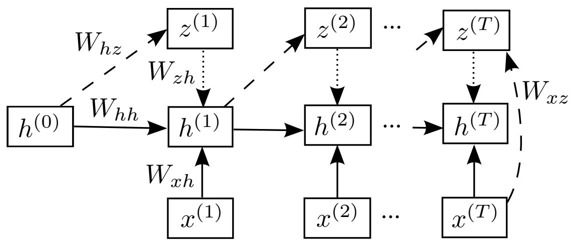

Figure 2. Graphical structure of the RNN. Single arrows represent a deterministic function, dotted arrows represent optional connections for temporal smoothing, dashed arrows represent a prediction. The $x \to z$ connections have been omitted for clarity at each time step except the last.

where $\mathcal{A}^{(t)} \equiv \{x, z^{(\tau)} | \tau < t\}$ is the sequence history at time t, $x \equiv \{x^{(t)}\}$ and $z \equiv \{z^{(t)} \in C\}$ are respectively the input and output sequences (both are given during supervised training), C is the dictionary of possible chord labels ($|C| = N$), and $P(z^{(t)}|\mathcal{A}^{(t)})$ is the conditional probability of observing $z^{(t)}$ according to the model, defined below in equation (9).

A single-layer RNN with hidden units $h^{(t)}$ is defined by its recurrence relation:

$$h^{(t)} = \sigma(W_{zh}z^{(t)} + W_{hh}h^{(t-1)} + W_{xh}x^{(t)} + b_h) \quad (6)$$

where the indices of weight matrices and bias vectors have obvious meanings. Its graphical structure is illustrated in Figure 2.

The prediction $y^{(t)}$ is obtained from the hidden units at the previous time step $h^{(t-1)}$ and the current observation $x^{(t)}$:

$$y^{(t)} = s(W_{hz}h^{(t-1)} + W_{xz}x^{(t)} + b_z) \quad (7)$$

where $s(a)$ is the softmax function of an activation vector a:

$$(s(a))_j \equiv \frac{\exp(a_j)}{\sum_{j'=1}^{N} \exp(a_{j'})}, \quad (8)$$

and should be as close as possible to the target vector $z^{(t)}$. In recognition problems with several classes, such as chord recognition, the target is a one-hot vector and the likelihood of an observation is given by the dot product:

$$P(z^{(t)}|\mathcal{A}^{(t)}) = z^{(t)} \cdot y^{(t)}. \quad (9)$$

3.2 Training

The RNN model can be trained by maximum likelihood with the following cost (replacing eq. 2):

$$L(x, z) = -\sum_{t=1}^{T} \log(z^{(t)} \cdot y^{(t)}) \quad (10)$$

where the gradient with respect to the model parameters is obtained by backpropagation through time (BPTT) [26].

While in principle a properly trained RNN can describe arbitrarily complex temporal dependencies at multiple time scales, in practice gradient-based training suffers from various pathologies [3]. Several strategies can be used to help reduce these difficulties including gradient clipping, leaky integration, sparsity and Nesterov momentum [2].

It may seem strange that the $z^{(t)}$ variable acts both as a target to the prediction $y^{(t)}$ and as an input to the RNN. How will these labels be obtained to drive the network during testing? In the transduction framework [7, 12], the objective is to infer the sequence $\{z^{(t)*}\}$ with maximal probability given the input. The search for a global optimum is a difficult problem addressed in the next section. Note that the connections $z \to h$ are responsible for temporal smoothing by forcing the predictions $y^{(t)}$ to be consistent with the previous decisions $\{z^{(\tau)}|\tau < t\}$. The special case $W_{zh} = 0$ gives rise to a recognition network without temporal smoothing.

A potential difficulty with this training scenario stems from the fact that since z is known during training, the model might (understandably) assign more weight to the symbolic information than the acoustic information. This form of *teacher forcing* during training could have dangerous consequences at test time, where the model is autonomous and may not be able to recover from past mistakes. The extent of this condition can be partly controlled by adding the regularization terms $\alpha(|W_{xz}|^2 + |W_{xh}|^2) + \beta(|W_{hz}|^2 + |W_{hh}|^2)$ to the objective function, where the hyperparameters α and β are weighting coefficients. It is trivial to revise the stochastic gradient descent updates to take those penalties into account.

4. INFERENCE

A distinctive feature of our architecture are the (optional) connections $z \to h$ that implicitly tie $z^{(t)}$ to its history $\mathcal{A}^{(t)}$ and encourage coherence between successive output frames, and temporal smoothing in particular. At test time, predicting one time step $z^{(t)}$ requires the knowledge of the previous decisions on $z^{(\tau)}$ (for $\tau < t$) which are yet uncertain (not chosen optimally), and proceeding in a greedy chronological manner does not necessarily yield configurations that maximize the likelihood of the complete sequence. We rather favor a global search approach analogous to the Viterbi algorithm for discrete-state HMMs.

4.1 Viterbi decoding

The simplest form of temporal smoothing is to use an HMM on top of a frame-level classifier. The HMM is a directed graphical model defined by its conditional independence relations:

$$P(x^{(t)}|\{x^{(\tau)}, \tau \neq t\}, z) = P(x^{(t)}|z^{(t)}) \quad (11)$$

$$P(z^{(t)}|\{z^{(\tau)}, \tau < t\}) = P(z^{(t)}|z^{(t-1)}) \quad (12)$$

where the emission probability can be formulated using Bayes' rule [17]:

$$P(x^{(t)}|z^{(t)}) \propto \frac{P(z^{(t)}|x^{(t)})}{P(z^{(t)})} \quad (13)$$

where $P(z^{(t)}|x^{(t)})$ is the output of the classifier and constant terms given x have been removed. Since the resulting joint distribution

$$P(z^{(t)}, x^{(t)}|\{z^{(\tau)}, \tau < t\}) \propto \frac{P(z^{(t)}|x^{(t)})}{P(z^{(t)})} P(z^{(t)}|z^{(t-1)}) \tag{14}$$

depends only on $z^{(t-1)}$, it is easy to derive a recurrence relation to optimize z^* by dynamic programming, giving rise to the well-known Viterbi algorithm.

4.2 Beam search

An established algorithm for sequence transduction with RNNs is beam search (Algorithm 1) [7, 12]. Beam search is a breadth-first tree search where only the w most promising paths (or nodes) at depth t are kept for future examination. In our case, a node at depth t corresponds to a subsequence of length t, and all descendants of that node are assumed to share the same sequence history $\mathcal{A}^{(t+1)}$; consequently, only $z^{(t)}$ is allowed to change among siblings. This structure facilitates identifying the most promising paths by their cumulative log-likelihood. Note that $w = 1$ reduces to a greedy search, and $w = N^T$ corresponds to an exhaustive breadth-first search.

Algorithm 1 BEAM SEARCH

Find the most likely sequence $\{z^{(t)} \in C|1 \leq t \leq T\}$ given x with beam width $w \leq N^T$.

```
1: $q \leftarrow$ priority queue
2: $q$.insert(0, $\{\}$)
3: for $t = 1 \ldots T$ do
4:     $q' \leftarrow$ priority queue of capacity $w$ *
5:     for $z$ in $C$ do
6:         for $l, s$ in $q$ do
7:             $q'$.insert($l + \log P(z^{(t)} = z|x, s), \{s, z\}$)
8:     $q \leftarrow q'$
9: return $q$.max()
```

*A priority queue of fixed capacity w maintains (at most) the w *highest* values at all times.

4.3 Dynamic programming

A pathological condition that sometimes occurs with beam search is the exponential duplication of highly likely quasi-identical paths differing only at a few time steps, that quickly saturate beam width with essentially useless variations. In that context, we propose a natural extension to beam search that makes a better use of the available width w and results in better performance. The idea is to make a trade-off between an RNN for which $z^{(t)}$ fully depends on $\mathcal{A}^{(t)}$ but exact inference is intractable, and an HMM for which $z^{(t)}$ explicitly depends only on $z^{(t-1)}$ but exact inference is in $O(TN^2)$.

We hypothesize that it is sufficient to consider only the most promising path out of all partial paths with identical $z^{(t)}$ when making a decision at time t. Under this assumption, any subsequence $\{z^{(t)*}|t \leq T'\}$ of the global optimum $\{z^{(t)*}\}$ ending at time $T' < T$ must also be optimal under the constraint $z^{(T')} = z^{(T')*}$. Note that relaxing this last constraint would lead to a greedy solution. Setting $T' = T - 1$ leads to the dynamic programming-like (DP) solution of keeping track of the N most likely paths arriving at each possible label $j \in C$ with the recurrence relation:

$$l_j^{(t)} = l_{k_j^{(t)}}^{(t-1)} + P(z^{(t)} = j|x, s_{k_j^{(t)}}^{(t-1)}) \tag{15}$$

$$s_j^{(t)} = \{s_{k_j^{(t)}}^{(t-1)}, j\} \tag{16}$$

$$\text{with } k_j^{(t)} \equiv \operatorname*{argmax}_{k=1}^{N} \left[l_k^{(t-1)} + P(z^{(t)} = j|x, s_k^{(t-1)})\right] \tag{17}$$

and initial conditions $l_j^{(0)} = 0, s_j^{(0)} = \{\}$, where the variables $l_j^{(t)}, s_j^{(t)}$ represent respectively the maximal cumulative log-likelihood and the associated partial output sequence ending with label j at time t (Algorithm 2). It is also possible to keep only the $w \leq N$ most promising paths to mimic an *effective beam width* and to make the algorithm very similar to beam search.

Algorithm 2 DYNAMIC PROGRAMMING INFERENCE

Find the most likely sequence $\{z^{(t)} \in C|1 \leq t \leq T\}$ given x with effective width $w \leq N$.

```
1: $q \leftarrow$ priority queue
2: $q$.insert(0, $\{\}$)
3: for $t = 1 \ldots T$ do
4:     $q' \leftarrow$ priority queue of capacity $w$
5:     for $z$ in $C$ do
6:         $l, s \leftarrow \text{argmax}_{(l,s)\in q} \left[l + \log P(z^{(t)} = z|x, s)\right]$
7:         $q'$.insert($l + \log P(z^{(t)} = z|x, s), \{s, z\}$)
8:     $q \leftarrow q'$
9: return $q$.max()
```

It should not be misconstrued that the algorithm is limited to “local” or greedy decisions for two reasons: (1) the complete sequence history $\mathcal{A}^{(t)}$ is relevant for the prediction $y^{(t)}$ at time t, and (2) a decision $z^{(t)*}$ at time t can be affected by an observation $x^{(t+\delta t)}$ arbitrarily far in the future via *backtracking*, analogously to Viterbi decoding. Note also that the algorithm obviously does not guarantee a globally optimal solution z^*, but is referred to as DP due to its strong similarity to the Viterbi recurrence relations.

5. EXPERIMENTS

5.1 Setup

This section describes experiments conducted on the dataset used in the MIREX audio chord estimation task [1] . Ground truth time-aligned chord symbols were mapped to the *major/minor* and *full chord* dictionaries comprising respectively 25 and 121 chord labels:

- $C_{\text{majmin}} \equiv \{\text{N}\} \cup \{\text{maj, min}\} \times S$,
- $C_{\text{full}} \equiv \{\text{N}\} \cup \{\text{maj, min, maj/3, maj/5, maj6, maj7, min7, 7, dim, aug}\} \times S$,

[1] `http://www.music-ir.org/mirex/wiki/2012:Audio_Chord_Estimation`

where S represents the 12 pitch classes and 'N' is the *no-chord* label [16, 21]. This allows us to evaluate our algorithm at different precision levels. Evaluation at the major/minor level is based on chord overlap ratio (OR) and weighted average OR (WAOR), standard denominations for the average frame-level accuracy [22, 25].

Results are reported using 3-fold cross-validation. For each of the 3 partitions, 25% of the training sequences are randomly selected and held out for validation. The hyperparameters of each model are selected over predetermined search grids to maximize validation accuracy and we report the final performance on the test set. In all experiments, we use 2 hidden layers of 200 units for the DBN, 100 hidden units for the RNN, and 8 pooling windows with $1 \leq L \leq 120$ s during pre-processing.

In order to compare our method against MIREX pre-trained systems, we also train and test our model on the whole dataset. It should be noted that this scenario is strongly prone to overfitting: from a machine learning perspective, it is trivial to design a non-parametric model performing at 100% accuracy. The objective is to contrast our results to previously published data, to analyze our models trained with equivalent features, and to provide an upper bound on the performance of the system.

5.2 Results

In Table 1, we present the cross-validation accuracies obtained on the MIREX dataset at the major/minor level using a DBN fine-tuned with chord labels z (DBN-1) and with chromagram intermediate targets $\tilde{z}$ and chord labels z (DBN-2), in addition to an RNN with DP inference. The DBN predictions are either not post-processed, smoothed with a Gaussian kernel ($\sigma = 760$ ms) or decoded with an HMM. The HPA [25] and DHMM [9] state-of-the-art methods are also provided for comparison.

Model	Smoothing	OR	WAOR
	None	65.8%	65.2%
DBN-1	Kernel	75.2%	74.6%
	HMM	74.3%	74.2%
	None	68.0%	67.3%
DBN-2	Kernel	78.1%	77.6%
	HMM	77.3%	77.2%
RNN	DP	**80.6%**	80.4%
HPA [25]	HMM	79.4%	78.8%
DHMM [9]	HMM	N/A	**84.2%**†

Table 1. Cross-validation accuracies obtained on the MIREX dataset using a DBN fine-tuned with chord labels z (DBN-1) and with chromagram intermediate targets $\tilde{z}$ and chord labels z (DBN-2), an RNN with DP inference, and the HPA [25] and DHMM [9] state-of-the-art methods. †4-fold cross-validation result taken from [9].

It is clear that optimizing the DBN with chromagram intermediate targets ultimately increases the accuracy of the classifier, and that the RNN outperforms the simpler models in both OR and WAOR. We also observe that kernel smoothing (a simple form of low-pass filtering) surprisingly outperforms the more sophisticated HMM approach. As argued previously [8], the relatively poor performance of the HMM may be due to the context information added to the input $x^{(t)}$ in equations (3-4). When the input includes information from neighboring frames, the independence property (11) breaks down, making it difficult to combine the classifier with the language model in equation (14). Intuitively, multiplying the predictions $P(z^{(t)}|x^{(t)})$ and $P(z^{(t)}|z^{(t-1)})$ to estimate the joint distribution will count certain factors twice since both models have been trained separately. The RNN addresses this issue by directly predicting the probability $P(z^{(t)}|\mathcal{A}^{(t)})$ needed during inference.

We now present a comparison between pre-trained models in the MIREX major/minor task (Table 2), where the superiority of the RNN to the DBN-2 is apparent. The RNN also outperforms competing approaches, demonstrating a high flexibility in describing temporal dependencies. Similar results can be observed at the full chord level with 121 labels (not shown).

Method	OR	WAOR
Chordino [22]	80.2%	79.5%
GMM + HMM [20]	82.9%	81.6%
HPA [25]	83.5%	82.7%
Proposed (DBN-2)	89.5%	89.8%
Proposed (RNN)	**93.5%**	**93.6%**

Table 2. Chord recognition performance (training error) of different methods pre-trained on the MIREX dataset.

To illustrate the computational advantage of DP inference over beam search, we plot the WAOR as a function of beam width w for both algorithms. Figure 3 shows that maximal accuracy is reached with a much lower width for DP ($w^* \simeq 10$) than for beam search ($w^* > 500$). The former can be run in 10 minutes on a single processor while the latter requires 38 hours for the whole dataset. While the time complexity of our algorithm is $O(TNw)$ versus $O(TNw \log w)$ for beam search, the performance gain can be mainly attributed to the possibility of significantly reducing w while preserving high accuracy. This is due to an efficient pruning of similar paths ending at $z^{(t)}$, presumably because the hypothesis stated in Section 4.3 holds well in practice.

6. CONCLUSION

We presented a comprehensive system for automatic chord recognition from audio music, that is competitive with existing state-of-the-art approaches. Our RNN model can learn basic musical properties such as temporal continuity, harmony and temporal dynamics, and efficiently search for the most musically plausible chord sequences when the audio signal is ambiguous, noisy or weakly discriminative. Our DP algorithm enables real-time decoding in live situations and would also be applicable to speech recognition.

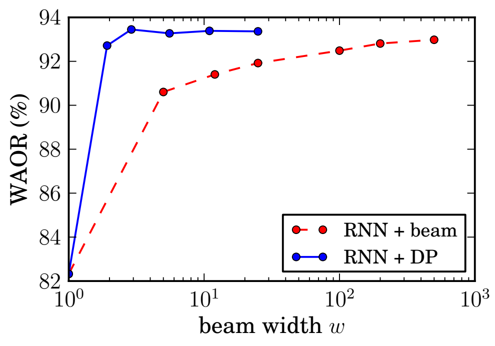

Figure 3. WAOR obtained on the MIREX dataset with the beam search and dynamic programming algorithms as a function of the (effective) beam width w.

7. REFERENCES

[1] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

[2] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu. Advances in optimizing recurrent networks. In *ICASSP*, 2013.

[3] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. on Neural Networks*, 5(2):157–166, 1994.

[4] J. Bergstra, N. Casagrande, D. Erhan, D. Eck, and B. Kégl. Aggregate features and adaboost for music classification. *Machine Learning*, 65(2-3):473–484, 2006.

[5] S. Böck and M. Schedl. Polyphonic piano note transcription with recurrent neural networks. In *ICASSP*, pages 121–124, 2012.

[6] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *ICML 29*, 2012.

[7] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. High-dimensional sequence transduction. In *ICASSP*, 2013.

[8] P. Brown. *The acoustic-modeling problem in automatic speech recognition*. PhD thesis, Carnegie-Mellon University, 1987.

[9] R. Chen, W. Shen, A. Srinivasamurthy, and P. Chordia. Chord recognition using duration-explicit hidden Markov models. In *ISMIR*, 2012.

[10] G. E. Dahl, D. Yu, L. Deng, and A. Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30–42, 2012.

[11] D. Eck and J. Schmidhuber. Finding temporal structure in music: Blues improvisation with LSTM recurrent networks. In *NNSP*, pages 747–756, 2002.

[12] A. Graves. Sequence transduction with recurrent neural networks. In *ICML 29*, 2012.

[13] Ç. Gülçehre and Y. Bengio. Knowledge matters: Importance of prior information for optimization. *ICLR*, 2013.

[14] P. Hamel, Y. Bengio, and D. Eck. Building musically-relevant audio features through multiple timescale representations. In *ISMIR*, 2012.

[15] P. Hamel and D. Eck. Learning Features from Music Audio with Deep Belief Networks. In *ISMIR*, pages 339–344, 2010.

[16] C. Harte. *Towards automatic extraction of harmony information from music signals*. PhD thesis, University of London, 2010.

[17] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition. *Signal Processing Magazine*, 29(6):82–97, 2012.

[18] G. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

[19] E. J. Humphrey and J. P. Bello. Rethinking automatic chord recognition with convolutional neural networks. In *ICMLA 11*, volume 2, pages 357–362, 2012.

[20] M. Khadkevich and M. Omologo. Time-frequency reassigned features for automatic chord recognition. In *ICASSP*, pages 181–184. IEEE, 2011.

[21] M. Mauch. *Automatic chord transcription from audio using computational models of musical context*. PhD thesis, University of London, 2010.

[22] M. Mauch and S. Dixon. Approximate note transcription for the improved identification of difficult chords. In *ISMIR*, pages 135–140, 2010.

[23] M. C. Mozer. Neural network music composition by prediction. *Connection Science*, 6(2):247–280, 1994.

[24] J. Nam, J. Ngiam, H. Lee, and M. Slaney. A classification-based polyphonic piano transcription approach using learned feature representations. In *ISMIR*, 2011.

[25] Y. Ni, M. McVicar, R. Santos-Rodríguez, and T. De Bie. An end-to-end machine learning system for harmonic analysis of music. *Audio, Speech, and Language Processing*, 20(6):1771–1783, 2012.

[26] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In *Parallel Dist. Proc.*, pages 318–362. MIT Press, 1986.

[27] P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In *Parallel Dist. Proc.*, pages 194–281. MIT Press, 1986.