Université de Montréal

Modeling High-Dimensional Audio Sequences with Recurrent Neural Networks

par Nicolas Boulanger-Lewandowski

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Thèse présentée à la Faculté des arts et des sciences
en vue de l'obtention du grade de Philosophiæ Doctor (Ph.D.)
en informatique

Avril, 2014

Résumé

Cette thèse étudie des modèles de séquences de haute dimension basés sur des réseaux de neurones récurrents (RNN) et leur application à la musique et à la parole. Bien qu'en principe les RNN puissent représenter les dépendances à long terme et la dynamique temporelle complexe propres aux séquences d'intérêt comme la vidéo, l'audio et la langue naturelle, ceux-ci n'ont pas été utilisés à leur plein potentiel depuis leur introduction par Rumelhart et al. (1986a) en raison de la difficulté de les entraîner efficacement par descente de gradient. Récemment, l'application fructueuse de l'optimisation Hessian-free et d'autres techniques d'entraînement avancées ont entraîné la recrudescence de leur utilisation dans plusieurs systèmes de l'état de l'art. Le travail de cette thèse prend part à ce développement.

L'idée centrale consiste à exploiter la flexibilité des RNN pour apprendre une description probabiliste de séquences de *symboles*, c'est-à-dire une information de haut niveau associée aux signaux observés, qui en retour pourra servir d'à priori pour améliorer la précision de la recherche d'information. Par exemple, en modélisant l'évolution de groupes de notes dans la musique polyphonique, d'accords dans une progression harmonique, de phonèmes dans un énoncé oral ou encore de sources individuelles dans un mélange audio, nous pouvons améliorer significativement les méthodes de transcription polyphonique, de reconnaissance d'accords, de reconnaissance de la parole et de séparation de sources audio respectivement. L'application pratique de nos modèles à ces tâches est détaillée dans les quatre derniers articles présentés dans cette thèse.

Dans le premier article, nous remplaçons la couche de sortie d'un RNN par des machines de Boltzmann restreintes conditionnelles pour décrire des distributions de sortie multimodales beaucoup plus riches. Dans le deuxième article, nous évaluons et proposons des méthodes avancées pour entraîner les RNN. Dans les quatre derniers articles, nous examinons différentes façons de combiner nos modèles symboliques à des réseaux profonds et à la factorisation matricielle non-négative, notamment par des produits d'experts, des architectures entrée/sortie et des cadres génératifs généralisant les modèles de Markov cachés. Nous proposons et analysons également des méthodes d'inférence efficaces pour ces modèles, telles la recherche vorace chronologique, la recherche en faisceau à haute dimension, la recherche en faisceau élagué et la descente de gradient. Finalement, nous abordons les questions de l'étiquette biaisée, du maître imposant, du lissage temporel, de la régularisation et du pré-entraînement.

Mots-clés: apprentissage automatique, réseaux de neurones récurrents, recherche d'information musicale, modèles séquentiels, transcription polyphonique, reconnaissance de la parole, factorisation matricielle non-négative.

Summary

This thesis studies models of high-dimensional sequences based on recurrent neural networks (RNNs) and their application to music and speech. While in principle RNNs can represent the long-term dependencies and complex temporal dynamics present in real-world sequences such as video, audio and natural language, they have not been used to their full potential since their introduction by Rumelhart et al. (1986a) due to the difficulty to train them efficiently by gradient-based optimization. In recent years, the successful application of Hessian-free optimization and other advanced training techniques motivated an increase of their use in many state-of-the-art systems. The work of this thesis is part of this development.

The main idea is to exploit the power of RNNs to learn a probabilistic description of sequences of *symbols*, i.e. high-level information associated with observed signals, that in turn can be used as a prior to improve the accuracy of information retrieval. For example, by modeling the evolution of note patterns in polyphonic music, chords in a harmonic progression, phones in a spoken utterance, or individual sources in an audio mixture, we can improve significantly the accuracy of polyphonic transcription, chord recognition, speech recognition and audio source separation respectively. The practical application of our models to these tasks is detailed in the last four articles presented in this thesis.

In the first article, we replace the output layer of an RNN with conditional restricted Boltzmann machines to describe much richer multimodal output distributions. In the second article, we review and develop advanced techniques to train RNNs. In the last four articles, we explore various ways to combine our symbolic models with deep networks and non-negative matrix factorization algorithms, namely using products of experts, input/output architectures, and generative frameworks that generalize hidden Markov models. We also propose and analyze efficient inference procedures for those models, such as greedy chronological search, high-dimensional beam search, dynamic programming-like pruned beam search and gradient descent. Finally, we explore issues such as label bias, teacher forcing, temporal smoothing, regularization and pre-training.

Keywords: machine learning, recurrent neural networks, music information retrieval, sequential models, polyphonic transcription, speech recognition, non-negative matrix factorization.

Contents

List of Figures

List of Tables

List of Abbreviations

ASR	Automatic speech recognition
BPTT	Backpropagation through time
BRNN	Bidirectional recurrent neural network
CD	Contrastive divergence
CE	Cross entropy
CG	Conjugate gradient
CRBM	Conditional restricted Boltzmann machine
CRF	Conditional random field
DBN	Deep belief network
DBN	Dynamic Bayesian network
DNN	Deep neural network
DP	Dynamic programming
EM	Expectation-maximization
ESN	Echo state network
GMM	Gaussian mixture model
HF	Hessian-free
HMM	Hidden Markov model
I/O	Input/output
KL	Kullback-Leibler divergence
LR	Logistic regression
LSTM	Long short term memory
MEMM	Maximum entropy Markov model
MFFE	Multiple fundamental frequency estimation
MIR	Music information retrieval
MLP	Multilayer perceptron
MRF	Markov random field

MSE	Mean squared error
NADE	Neural autoregressive distribution estimator
NAG	Nesterov accelerated gradient
NLL	Negative log-likelihood
NMF	Non-negative matrix factorization
OR	Overlap ratio
PCA	Principal component analysis
RBM	Restricted Boltzmann machine
RTRBM	Recurrent temporal restricted Boltzmann machine
RNN	Recurrent neural network
SAR	Sources to artifacts ratio
SDR	Signal to distortion ratio
SGD	Stochastic gradient descent
SIR	Source to interference ratio
STFT	Short-term Fourier transform
SVM	Support vector machine
TRBM	Temporal restricted Boltzmann machine
WAOR	Weighted average overlap ratio

Acknowledgments

I extend my sincere gratitude to my thesis co-advisors Yoshua Bengio and Pascal Vincent, professors at the department of Computer Science and Operations Research. In the last few years, I could benefit from their vast scientific knowledge, expert guidance and generous availability that allowed me to carry out a captivating project. I would also like to thank all members of the LISA and GAMME labs who provided a stimulating and creative environment for research.

Thanks to Frédéric Bastien for his constant availability and technical support, and to Theano developers for making this very useful software library a reality. In particular, thanks to Razvan Pascanu and Ian Goodfellow for their work on the scan op and the R-operator, without whom this work would not have been possible.

I am indebted to Douglas Eck, now Research Scientist at Google, for getting me first interested in machine learning and music information retrieval, and acting as a mentor at the beginning of my PhD and during my internship at Google; to Jasha Droppo and the people at Microsoft Research with whom I collaborated for providing me with the opportunity to apply some of my ideas to speech recognition; and to Gautham Mysore and Matthew Hoffman at Adobe Research for allowing me to actualize some long-standing ideas related to non-negative matrix factorization and source separation.

I would also like to thank NSERC for awarding me the prestigious Alexander Graham Bell Canada Doctoral Scholarship, and the Canada Research Chairs for funding. Their support allowed me to spend time on my project uninterrupted and attend conferences across the world.

I am thankful to my master's thesis advisor Alain Rochefort, professor at the Engineering Physics department of Polytechnique Montréal, who showed me the ropes of academic research and scientific paper writing.

Finally, I am grateful to my parents Solange and Jacques for introducing me to the natural sciences at a young age and for encouraging me to pursue higher education, and to whom I attribute much of my success in this adventure.

1 Introduction

This thesis focuses on advancing the state of the art in sequence modeling, and thereby improving several applications in the area of polyphonic music and speech, namely polyphonic music generation and transcription, audio chord recognition, speech recognition and audio source separation. Modeling real-world sequences often involves capturing long-term dependencies between the high-dimensional objects that compose such sequences. This problem is in general too difficult to tackle by manually engineering rules to process the data in each possible scenario and we instead follow a machine learning approach.

1.1 Modeling high-dimensional sequences

Modeling sequences is an important area of machine learning since many naturally occurring phenomena such as music, speech, or human motion are inherently sequential. This section outlines some properties of such sequences that are particularly challenging to model.

Complex sequences are *non-local* in that the impact of a factor localized in time can be delayed by an arbitrarily long time-lag. For example, musical patterns or themes appearing at the beginning of a piece are often repeated towards the end; similarly, the meaning of a particular sentence in a text often depends on references introduced much earlier. Recurrent neural networks (RNNs) (Rumelhart et al., 1986a) incorporate an internal memory that can, in principle, summarize the entire sequence history. This property makes them well suited to represent long-term dependencies, but it is nevertheless a challenge to train them efficiently by gradient-based optimization (Bengio et al., 1994). It was recently shown that several training strategies could help reduce these difficulties, motivating their use as sequential models in this thesis. RNNs can also be used to generate realistic sequences in different styles (Sutskever et al., 2011; Graves, 2013).

Many sequences of interest are over *high-dimensional* objects, such as images in video, short-term spectra in audio music, tuples of notes in musical scores, or words in text. In these cases, predicting the value at the next time step given the observed values at the previous time steps is complicated by the fact that the *conditional distribution* of that value given the previous time steps is very often multimodal. For the case of polyphonic music, it is obvious that the occurrence of a particular note at a particular time modifies considerably the probability with which other notes may occur at the same time. In other words, notes appear together in correlated patterns that cannot be conveniently described by a typical RNN architecture designed for the multi-label classification task, for example, because simply predicting the expected value of each unit at the next time step would produce an incoherent blend of the different modes. The other extreme of enumerating all configurations of the variable to predict in a multiclass classification framework would be very expensive. We would strongly prefer our models of such sequences to predict realistic multimodal conditional distributions of the next time step. This motivates using energy-based models which allow us to express the negative log-likelihood of a given configuration by an arbitrary energy function, among which the restricted Boltzmann machine (RBM) (Smolensky, 1986) has become notorious. One of our contributions will be to develop an RNN variant that employs multimodal conditional RBM distributions (Chapters 3 and 4).

In the practical applications tackled in this thesis, it is often useful to impose high-level constraints on the output sequence, in the same way that natural language models encourage the coherence of a transcribed segment during speech recognition (Rabiner, 1989). Naturally, our sequence models will be ideal candidates to describe such constraints, as outlined in the next section.

1.2 Generalized language models

Many applications involve the transformation, or *transduction*, of an input sequence into an output sequence. This output sequence can be a string of high-dimensional symbols that provide an abstract description of the observed signal, such as chord labels in polyphonic music or words in speech. Those annotations themselves often exhibit recurrent patterns that adhere to certain probabilistic rules. For example, it is well known that chord progressions favor smooth transitions, that musical scores follow harmonic and rhythmic principles, that words in text satisfy grammatical and semantic constraints, and that individual sounds in a mixture obey temporal dynamics specific to each source. We refer to these high-level descriptions as generalized language models or *symbolic* models. Note that symbolic models do not forcibly impose any rigid constraints; their probabilistic nature rather allows for the occasional exception to the rule.

Humans commonly interpret music and speech by giving importance to what they expect to hear rather than exclusively to what is present in the actual signal. Unfortunately, many computer algorithms still rely exclusively on the audio signal or employ only rudimentary temporal constraints. It has long been known that temporal priors can improve purely auditive approaches to speech recognition (Schuster, 1999b), polyphonic transcription (Cemgil, 2004; Cemgil et al., 2006; Raphael, 2002), chord recognition (de Haas et al., 2012) and audio source separation (Virtanen, 2007). However, combining these two sources of information is not trivial, with the result that temporal smoothing with an HMM is often the only post-processing involved in the state of the art (e.g. Nam et al., 2011; Dahl et al., 2013).

In this thesis, we replace the tedious process of handcrafting symbolic rules with powerful sequential models that can be learned directly from training data. In addition to being more readily adaptable to the styles of different corpora, this approach can also exploit the vastly available unlabeled data in either acoustic or symbolic form to refine each model independently. This will ultimately allow us to simulate the synergy between two regions of the brain (e.g. a sensory region reacting to external stimuli and an analytical region focusing on high-level interpretation would correspond to the acoustic and symbolic models respectively) while learning to process sequential events.

1.3 Applications

We shall now present the set of music and speech related applications of sequence modeling that we will consider improving and that are the focus of the articles presented in this thesis.

1.3.1 Polyphonic music generation and related applications

Learning realistic probabilistic models of polyphonic music has many applications, the most obvious one being to automatically generate music (Mozer, 1994). Those models can be used to morph the style of background music in video games depending on the context (Wooller and Brown, 2005), or form the backbone of an automatic accompaniment system or melody improviser for aspiring musicians (Davies, 2007). By training different models on datasets having distinct characteristics such as their genre, it is possible to build a Bayes classifier to predict musical genre by comparing the likelihood of unseen pieces according to each model. Inferring the tonality of a piece is also possible by finding the transposition that maximizes its likelihood under a model trained on fixed tonality pieces (Aljanaki, 2011; Krumhansl and Kessler, 1982). Finally, polyphonic music models can improve music information retrieval (MIR) algorithms such as polyphonic transcription or onset detection by serving as a musicological language prior describing the missing information (Cemgil, 2004).

In this work, we consider sequences of symbolic music, i.e. represented by the explicit timing, pitch, velocity and instrumental information typically contained in a score or a MIDI file rather than more complex, acoustically rich audio signals. Musical models mostly focus on the basic components of western music, harmony and rhythm, and are trained to predict the pattern of notes (simultaneities) to be played together in the next time interval, given the previous ones. Two elements characterize the qualitative performance of a model: temporal dependencies and chord conditional distributions. While most existing models output only monophonic notes along with predefined chords or other reduced-dimensionality representation (e.g. Mozer, 1994; Eck and Schmidhuber, 2002; Paiement et al., 2009), we aim to model unconstrained polyphonic music in the piano-roll representation, i.e. as a binary matrix specifying precisely which notes occur at each time step. Despite ignoring dynamics and other score annotations, this task represents a well-defined framework to improve machine learning algorithms and is directly applicable to polyphonic transcription.

1.3.2 Polyphonic music transcription

The objective of polyphonic transcription, or multiple fundamental frequency estimation (MFFE), is to determine the underlying notes of a polyphonic audio signal without access to its score. For some authors, MFFE strictly involves recognizing audible note pitches at regular intervals (usually 10 ms) and polyphonic transcription additionally requires positioning correctly the note onsets and offsets. In this thesis, we employ both terms indiscriminately and report the common frame-level evaluation metrics of precision, recall, F-measure and accuracy (Bay et al., 2009):

$$\text{Precision} = \frac{\sum_{t=1}^{T} TP(t)}{\sum_{t=1}^{T} TP(t) + FP(t)} \tag{1.1}$$

$$\text{Recall} = \frac{\sum_{t=1}^{T} TP(t)}{\sum_{t=1}^{T} TP(t) + FN(t)} \tag{1.2}$$

$$\text{F-measure} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{1.3}$$

$$\text{Accuracy} = \frac{\sum_{t=1}^{T} TP(t)}{\sum_{t=1}^{T} TP(t) + FP(t) + FN(t)} \tag{1.4}$$

where $TP(t), FP(t), FN(t)$ denote respectively the number of true positives, false positves and false negatives at time step t.

Polyphonic transcription is a particular case of the audio source separation problem that becomes very hard when the polyphony (number of simultaneous notes) is higher than 4; state-of-the-art systems obtain around 65% accuracy in that case. Most existing transcription algorithms are frame-based and rely exclusively on the audio signal, even though some approaches employ rudimentary musicological constraints (e.g. Li and Wang, 2007). Generally, transcription algorithms primarily follow either a signal processing or a machine learning approach.

Signal processing approaches are usually based on the short term Fourier transform (STFT) with sliding analysis windows of length $\simeq$ 100 ms and zero-padded to 2–8 times their original size, that yield a so-called *spectrogram*, or time-frequency representation. At each time step, a number of heuristics are performed such as peak detection, peak classification, noise level estimation, iterative sinusoidal peak elimination (Abe and Smith, 2005), and evaluation of polyphonic salience functions, in order to form a set of pitch candidates of which all combinations are ranked via handcrafted multi-criteria score functions (Yeh, 2008). Overall, very few hyperparameters are tuned to instrumental sound corpora.

Machine learning approaches are based on training acoustic models in a supervised way on datasets of musical pieces and their score. Those acoustic models usually feed columns of the magnitude spectrogram to a multi-label classifier like the support vector machine (SVM) (Poliner and Ellis, 2007, 2005), a multilayer perceptron (MLP) (Marolt, 2004), or a non-negative matrix factorization (NMF) feature extractor (Plumbley et al., 2006; Abdallah and Plumbley, 2006; Smaragdis and Brown, 2003; Lee et al., 2010; Cont, 2006; Dessein et al., 2010). Learning algorithms are naturally more adaptable to new domain distributions (e.g. different instruments or styles) than handcrafted heuristics, provided that labeled data is available in the new domain. This last condition is unfortunately difficult to fulfill in practice due to the high cost of producing expert annotations, with the result that most available data is synthesized or obtained from automated means (Yeh et al., 2007). Furthermore, while learning-based transcription usually performs extremely well inside a single domain (Marolt, 2004), generalization to new variations as small as a different physical instrument of the same type is notoriously poor (Poliner and Ellis, 2007). Connectionist approaches also suffer from catastrophic forgetting whenever training examples from all domains are not continuously available during training (Goodfellow et al., 2014). Due to the large quantity of unlabeled data, it would be a tremendous advantage for new algorithms to be developed and evaluated in the self-taught framework (Raina et al., 2007), i.e. under both the semi-supervised and the transfer learning paradigms.

1.3.3 Audio chord recognition

Automatic recognition of chords from audio music is another active area of research in MIR (Mauch, 2010; Harte, 2010). The objective of chord recognition is to produce a time-aligned sequence of *chord labels* taken from predefined dictionaries C that describe the harmonic structure of the piece. Popular dictionaries include the major/minor task where inversions and deviations from the basic triads are neglected, and the full chord task that comprises 11 distinct chord types:

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

where $S \equiv \{\text{A, A\#, B, C, C\#, D, D\#, E, F, F\#, G, G\#}\}$ represents the 12 pitch classes and 'N' is the no-chord label (Harte, 2010; Mauch, 2010). This allows us to evaluate chord recognition at different precision levels. Evaluation at the major/minor level is based on chord overlap ratio (OR) and weighted average overlap ratio (WAOR), standard denominations for the average frame-level accuracy (Ni et al., 2012; Mauch and Dixon, 2010). At the full chord level, other metrics can be defined depending on whether we require an exact match between the target and the prediction, we allow inversions or we compare at the dyad level (Mauch and Dixon, 2010).

Audio chord recognition is related to polyphonic transcription in that estimating active pitches is a reasonable prerequisite; however in chord recognition, some octave errors, harmonic errors, missed notes, insertions or substitutions are tolerated. Chord annotations also tend to be more stable in time, i.e. they are usually not affected by transients and temporary spurious notes, which makes temporal models even more important in this context.

Many chord recognition systems are based on harmonic pitch class profiles (HPCP) (Fujishima, 1999), or *chroma* features, that compress a truncated spectrum into an octave-independent histogram of 12 bins (or a multiple if more resolution is desired) that describe the relative intensity of each of the 12 pitch classes. The chromagram, a matrix of time-dependent chroma features, is a useful representation for tonality and harmony based classification.

1.3.4 Speech recognition

Automatic speech recognition (ASR) is a widely studied problem in computer science that involves extracting phonemes, words or higher-level meaning from speech audio (Baker et al., 2009). This difficult problem is complicated by a strong speaker dependence on pronunciation, large vocabulary sizes, continuous and spontaneous speech, noisy conditions and application-specific constraints. In many practical applications, both accuracy and efficiency of decoding matter. Contrarily to polyphonic transcription or chord recognition, the output sequences need not necessarily be aligned in time; a concatenated string of contiguous segments often suffices.

Traditional models of ASR are based on hidden Markov models (HMMs) with Gaussian mixture model (GMM) emissions. Recently, deep learning methods have been very successful at replacing the GMM acoustic model in state-of-the-art systems (Dahl et al., 2012, 2013; Graves et al., 2013). It is important for learned acoustic models to take into account the *context dependency* of phonemes, i.e. the fact that the same phoneme can have drastically different realizations depending on the surrounding ones. This usually requires feeding large input context windows to a frame-level classifier and, with simple phonetic models like an HMM, defining distinct auxiliary phonemes, or *phones*, for each possible relevant context.

1.3.5 Audio source separation

Source separation is the problem of extracting individual channels, or *sources*, from a mixture signal. This problem naturally occurs in different modalities where the component signals combine *additively* such as images of unoccluded or transparent objects, electromagnetic waves, and more commonly music and speech audio, especially for denoising or to isolate specific parts (e.g. the cocktail party problem) (Benaroya et al., 2006).

Non-negative matrix factorization (NMF) (Lee and Seung, 1999) of the mixture spectrogram is an effective method for source separation because it can discover a basis of recurring interpretable patterns for each source that combine additively to reconstitute the observations. NMF assumes that each observed spectrogram frame is representable as a non-negative linear combination of the isolated sources, an approximation that depends on the interference between overlapping harmonic partials in a polyphonic mix but that is nevertheless reasonable (Yeh and Röbel, 2009). It also requires that basis spectra be linearly independent and appear in all possible combinations in the data. In addition to purely minimizing the NMF reconstruction error, it is useful to exploit prior knowledge about:

1. **the basis spectra**, such as sparsity (Cont, 2006), harmonicity (Vincent et al., 2010) or relevance with respect to a discriminative criterion (Boulanger-

Lewandowski et al., 2012a); and

2. **their time-varying encodings**, such as continuity (Virtanen, 2007) or other temporal behavior (e.g. Nam et al., 2012; Ozerov et al., 2009; Nakano et al., 2010; Mohammadiha and Leijon, 2013; Mysore et al., 2010).

In contrast to blind source separation (Cardoso, 1998) that uses very little prior knowledge about the sources, the supervised and semi-supervised NMF paradigms allow the use of training data to model those properties and hence facilitate the separation.

1.4 Overview

The remainder of this thesis is organized as follows.

In Chapter 2, we provide some background on the machine learning methods used in the rest of the thesis. We cover topics such as density estimation, sequential models, non-negative matrix factorization, deep learning and optimization.

In the first article (Chapters 3 and 4), we introduce the RNN-RBM, a model that replaces the output layer of an RNN with conditional restricted Boltzmann machines, and we perform extensive experiments on symbolic sequences of polyphonic music.

In the second article (Chapters 5 and 6), we review, analyze and combine advanced techniques to train RNNs, including a novel formulation of Nesterov momentum. We carry out experiments on symbolic polyphonic music and text data.

In the third article (Chapters 7 and 8), we introduce the input/output RNN-RBM that merges a symbolic and acoustic model under a joint training objective, and we devise an efficient inference algorithm called high-dimensional beam search. We apply our method to polyphonic music transcription.

In the fourth article (Chapters 9 and 10), we develop an RNN-based system for audio chord recognition. We propose a two-pass fine-tuning method to exploit the information contained in chords labels in the form of intermediate chromagram targets and we develop a dynamic programming-like beam search pruning technique that improves efficiency and accuracy of inference.

In the fifth article (Chapters 11 and 12), we introduce a hybrid architecture that generalizes the HMM to combine an RNN symbolic model with a frame-level acoustic classifier in a way that circumvents the label bias problem. We derive training, inference and alignment procedures and we study the role of phone sequence modeling in speech recognition.

In the sixth article (Chapters 13 and 14), we introduce a generative architecture that can reconstruct audio signals by incorporating an RNN prior on the NMF activities. We apply our method to separate voice and accompaniment tracks in a dataset of karaoke recordings (MIR-1K).

2 Background

In this chapter, we briefly present the machine learning methods that the rest of the thesis builds upon. We cover topics such as density estimation (Section 2.1), sequential models (Section 2.2), non-negative matrix factorization (Section 2.3), deep learning (Section 2.4) and optimization (Section 2.5).

2.1 Density estimators

In this section, we review two important generic density estimators: the restricted Boltzmann machine (RBM) and its tractable variant NADE. Those models allow to estimate the joint probability distribution, or multivariate *density*, of vectors v of size N observed in the training data. Those vectors are usually binary, i.e. $v \in \{0,1\}^N$, but extensions of both models dealing with real-valued vectors $v \in \mathbb{R}^N$ are also described.

2.1.1 Restricted Boltzmann machines

An RBM is an energy-based model where the joint probability of a given simultaneous configuration of visible vector v (inputs) and hidden vector h is:

$$P(v,h) = \exp(-b_v^{\mathrm{T}} v - b_h^{\mathrm{T}} h - h^{\mathrm{T}} W v)/Z \tag{2.1}$$

where b_v, b_h and W are the parameters of the model and Z is a normalization factor called the partition function. The classic RBM involves binary hidden units h_i and binary visible units v_j, but there are many other options in this regard (Welling et al., 2005). Note that computing Z is usually intractable. When the value of the vector v is given, the hidden units h_i are conditionally independent of one another, and vice-versa:

$$P(h_i = 1|v) = \sigma(b_h + Wv)_i \tag{2.2}$$

$$P(v_j = 1|h) = \sigma(b_v + W^{\mathrm{T}}h)_j \tag{2.3}$$

where $\sigma(x) \equiv (1 + e^{-x})^{-1}$ is the element-wise logistic sigmoid function. The marginalized probability of v is related to the free-energy $F(v)$ by $P(v) \equiv e^{-F(v)}/Z$:

$$F(v) = -b_v^{\mathrm{T}}v - \sum_i \log(1 + e^{b_h+Wv})_i \tag{2.4}$$

Inference in RBMs consists of sampling the h_i given v (or the v_j given h) according to their conditional Bernoulli distribution (equation 2.2). Sampling v from the RBM can be performed efficiently by block Gibbs sampling, i.e. by performing k alternating steps of sampling $h|v$ and $v|h$. Computing the gradient of the negative log-likelihood given a dataset of inputs $\{v^{(l)}\}$ involves two opposing terms, called the positive and negative phase:

$$\frac{\partial(-\log P(v^{(l)}))}{\partial\Theta} = \frac{\partial F(v^{(l)})}{\partial\Theta} - \frac{\partial(-\log Z)}{\partial\Theta} \tag{2.5}$$

where $\Theta \equiv \{b_v, b_h, W\}$ are the model parameters. Although Z is an intractable sum over all possible v configurations, its gradient can be estimated by a single sample $v^{(l)*}$ obtained from a k-step Gibbs chain starting at $v^{(l)}$:

$$\frac{\partial(-\log P(v^{(l)}))}{\partial\Theta} \simeq \frac{\partial F(v^{(l)})}{\partial\Theta} - \frac{\partial F(v^{(l)*})}{\partial\Theta}. \tag{2.6}$$

The resulting algorithm, dubbed k-step “contrastive divergence” (CD$_k$) (Hinton, 2002), works surprisingly well with chain lengths of $k = 1$ but higher k result in better log-likelihood, at the expense of more computation (proportional to k).

Gaussian RBMs

Instead of modeling the inputs as bits or as bit probabilities (which works well for discrete inputs such as the musical scores or the pixel intensities of our bouncing balls video), we can model them as Gaussian values, conditioned on the hidden units’ configurations. The simplest way to achieve this is to use a Gaussian

RBM (Welling et al., 2005), which simply adds a quadratic penalty term $||v||^2/2$ to the energy function. Equations (2.3) and (2.4) become:

$$P'(v|h) = \mathcal{N}(v; b_v + W^{\mathrm{T}}h, I) \tag{2.7}$$

$$F'(v) = -||v||^2/2 + F(v) \tag{2.8}$$

where $\mathcal{N}(v; \mu, \Sigma)$ is the density of v under the multivariate normal distribution of mean μ and variance Σ.

2.1.2 Neural autoregressive distribution estimator

The neural autoregressive distribution estimator (NADE) (Larochelle and Murray, 2011) is a tractable model inspired by the RBM and specializing (with tying constraints) an earlier model for the joint distribution of high-dimensional variables (Bengio and Bengio, 2000). NADE is similar to a fully visible sigmoid belief network in that the conditional probability distribution of a visible unit v_j is expressed as a nonlinear function of the vector $v_{<j} \equiv \{v_k, \forall k < j\}$:

$$P(v_j = 1|v_{<j}) = \sigma(V_{j,:}h_j + (b_v)_j) \tag{2.9}$$

$$h_j = \sigma(W_{:,<j}v_{<j} + b_h) \tag{2.10}$$

where $\sigma(x) \equiv (1 + e^{-x})^{-1}$ is the logistic sigmoid function, and V is an additional matrix parameter that can be set to $W^\top$, but in practice tying those weights is neither necessary nor beneficial.

In the following discussion, one can substitute RBMs with NADEs by replacing equation (2.6) with the exact gradient of the negative log-likelihood cost $C \equiv$

$-\log P(v)$:

$$\begin{aligned}
\frac{\partial C}{\partial (b_v)_j} &= P(v_j = 1 | v_{<j}) - v_j && (2.11)\\
\frac{\partial C}{\partial b_h} &= \sum_{k=1}^{N} \frac{\partial C}{\partial (b_v)_k} V_{k,:} h_k (1 - h_k) && (2.12)\\
\frac{\partial C}{\partial W_{:,j}} &= v_j \sum_{k=j+1}^{N} \frac{\partial C}{\partial (b_v)_k} V_{k,:} h_k (1 - h_k) && (2.13)\\
\frac{\partial C}{\partial V_{j,:}} &= \frac{\partial C}{\partial (b_v)_j} h_j && (2.14)
\end{aligned}$$

In addition to the possibility of using second-order methods for training, a tractable distribution estimator is necessary to compare the probabilities of different output sequences during inference, as explained in Chapter 8.

Real-valued NADE

Analogously to the Gaussian RBM, the real-value NADE (RNADE) (Uría et al., 2013) was recently introduced to estimate the multivariate density of real-valued vectors. The one-dimensional conditional distribution of the real-valued variable v_j given $v_{<j}$ (keeping the same notation as previously) is obtained by replacing (2.9) with a mixture of K Gaussian distributions:

$$P(v_j | v_{<j}) = \sum_{k=1}^{K} \frac{\alpha_{jk}}{\sigma_{jk}\sqrt{2\pi}} \exp\left[-\frac{(v_j - \mu_{jk})^2}{2\sigma_{jk}^2}\right] \tag{2.15}$$

where α_j, μ_j and σ_j are vectors denoting respectively the K mixing fractions, component means and standard deviations for the j-th unit. These parameters are obtained by a function of the hidden vector h_j:

$$\begin{aligned}
\alpha_j &= s\left(V_j^{\alpha T} h_j + b_j^{\alpha}\right) && (2.16)\\
\mu_j &= V_j^{\mu T} h_j + b_j^{\mu} && (2.17)\\
\sigma_j &= \exp\left(V_j^{\sigma T} h_j + b_j^{\sigma}\right) && (2.18)
\end{aligned}$$

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

$$(s(a))_k \equiv \frac{\exp(a_k)}{\sum_{k'=1}^{K} \exp(a_{k'})}. \tag{2.19}$$

Training an RNADE can be done using gradient descent after heuristically scaling the learning rate associated with each component mean (Uría et al., 2013).

2.2 Sequential models

In this section, we review important probabilistic sequential models, i.e. graphical models that assign a probability $P(z)$ to a sequence of T symbols $z \equiv \{z^{(t)}, 1 \leq t \leq T\}$. The symbols $z^{(t)}$ are usually vectors of length N and can be real-valued, binary, or one-hot (binary with unit norm). In the latter case, they can alternatively be represented by a single integer $1 \leq z^{(t)} \leq N$, depending on the context. Some models instead capture the conditional probability $P(z|x)$ of z given an input sequence $x \equiv \{x^{(t)}, 1 \leq t \leq T\}$, or *observations*. Many of the models presented in this section, including the RNN, exploit a common factorization of the joint probability distribution of z:

$$\begin{aligned} P(z) &= \prod_{t=1}^{T} P(z^{(t)}|\mathcal{A}^{(t)}) && (2.20) \\ \text{or: } P(z|x) &= \prod_{t=1}^{T} P(z^{(t)}|\mathcal{A}^{(t)}, x) && (2.21) \end{aligned}$$

where $\mathcal{A}^{(t)} \equiv \{z^{(\tau)}, \tau < t\}$ is the *sequence history* at time step t, i.e. the value of the previously emitted output symbols. It is important to remember that the output sequence z is a random variable in this framework, even when the model predictions are deterministic.

The models presented here are often so general that they can be applied to a wide variety of natural phenomena like music, speech, human motion, etc. We will highlight the advantages of each model for modeling musical sequences whenever appropriate.

2.2.1 Markov chains

A Markov chain of order k, or $(k+1)$-gram, is a stochastic process where the probability of observing the discrete state $z^{(t)}, 1 \leq z^{(t)} \leq N$ at time t depends only on the states at the previous k time steps:

$$P(z^{(t)}|\mathcal{A}^{(t)}) = P(z^{(t)}|\{z^{(\tau)}, t-k \leq \tau < t\}) \tag{2.22}$$

The actual probabilities are explicitly maintained in a transition table of N^k values.

More commonly applied to natural language modeling with state $z^{(t)}$ representing a word from the dictionary or a character from the alphabet, Markov chains are also well suited for monophonic music (Pickens, 2000) by letting $z^{(t)}$ represent the active pitch in the equal temperament at time t. We can also model the evolution of predefined chords (Pickens et al., 2002), which is still a relatively low-dimensional representation. Modeling polyphonic music with k-grams is harder due to the exponential number of possible note combinations.

An obvious limitation of this model is that the finite state transition probabilities depend only on a short sequence history, which prevents the model from exploiting non-local temporal dependencies, such as the overall context of the piece. Some approaches attempt to discover repeated patterns in a given piece before running the k-gram in order to alleviate this issue (Conklin, 2003; Paiement et al., 2007).

2.2.2 Hidden Markov models

A hidden Markov model (HMM) is a generative stochastic process where the observation $x^{(t)}$ at time t is conditioned on the corresponding hidden state $z^{(t)}$, which itself evolves according to a Markov chain (eq. 2.22) usually of order $k = 1$. The *generative* qualifier indicates that x is also a random variable. An HMM is a directed graphical model defined by its conditional independence relations:

$$\begin{aligned} P(x^{(t)}|\{x^{(\tau)}, \tau \neq t\}, z) &= P(x^{(t)}|z^{(t)}) && (2.23) \\ P(z^{(t)}|\mathcal{A}^{(t)}) &= P(z^{(t)}|\{z^{(\tau)}, t-k \leq \tau < t\}). && (2.24) \end{aligned}$$

Since the resulting joint distribution

$$P(z^{(t)}, x^{(t)} | \mathcal{A}^{(t)}) = P(x^{(t)} | z^{(t)}) P(z^{(t)} | \{z^{(\tau)}, t - k \leq \tau < t\}) \qquad (2.25)$$

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

The emission probability in equation (2.23) is often parametrized via a Gaussian mixture model (GMM, eq. 2.15), or formulated as a function of a classifier using Bayes' rule:

$$P(x^{(t)} | z^{(t)}) = \frac{P(z^{(t)} | x^{(t)}) P(x^{(t)})}{P(z^{(t)})} \qquad (2.26)$$

where $P(z^{(t)} | x^{(t)})$ is the output of the classifier. The latter case of stacking an HMM on top of a frame-level classifier corresponds to a simple form of temporal smoothing.

Despite their limitations, HMMs are popular models for polyphonic music transcription. The common strategy is to use separate HMMs with $N = 2$ states to transcribe each possible pitch independently. An input/output HMM (Bengio and Frasconi, 1996), in which the state transitions depend on an auxiliary input sequence, can also be useful to model melody lines in a given harmonic context (Paiement et al., 2009).

2.2.3 Dynamic Bayesian networks

Dynamic Bayesian networks (DBNs) (Murphy, 2002) are directed graphical models that exploit the factorization (2.20) to characterize the general evolution of a distributed, discrete-continuous mixed state from which the observations are emitted as in equation (2.23). Many configurations and parametrizations of states are possible in a DBN, giving rise to a number of particular cases, such as the HMM, the input/output HMM, the factorial HMM, and others.

A carefully constructed DBN incorporating multiple musicological sub-modules describing harmony, duration, voice and polyphony has recently been used to model polyphonic music in symbolic form (Raczynski et al., 2013).

2.2.4 Maximum entropy Markov models

Maximum entropy Markov models (MEMMs) (McCallum et al., 2000) are also directed graphical models that employ the conditional factorization of equation (2.21) in which the input x is not considered a random variable. This model additionally imposes Markovian assumptions and predictions in the form of a maximum entropy classifier:

$$\begin{aligned} P(z^{(t)}|\mathcal{A}^{(t)}, x) &= P(z^{(t)}|z^{(t-1)}, x^{(t)}) && (2.27) \\ &= s\big(W\phi(z^{(t-1)}, x^{(t)}) + b\big) && (2.28) \end{aligned}$$

where $s(\cdot)$ is the softmax non-linearity function (eq. 2.19), W, b are the weight matrix and bias vector, and ϕ is a feature vector that depends on $z^{(t-1)}$ and $x^{(t)}$. Training an MEMM via maximum likelihood is straightforward and similar to training a logistic regression model.

An advantage of the MEMM is the possibility to include in the feature vector ϕ a range of domain-specific discriminative features correlated with non-local observations, i.e. the dependence on $x^{(t)}$ alone in (2.27) is not a strict requirement. As argued previously (Brown, 1987), a similar procedure for HMM would violate the independence property (2.23) and make it difficult to combine the emission probability with the language model in equation (2.25). Intuitively, multiplying those predictions together to estimate the joint distribution in an HMM will count certain factors twice since both models have been trained separately. Note that this violation does not necessarily translate in a bad performance in practice. The MEMM nevertheless addresses this issue by predicting the relevant probability $P(z^{(t)}|\mathcal{A}^{(t)}, x)$ directly.

The label bias problem

In output sequences with low-entropy conditional distributions $P(z^{(t)}|\mathcal{A}^{(t)})$, a severe drawback of MEMM-like models is the *label bias* problem. Low-entropy conditional distributions can occur with frequently repeated output symbols, i.e. where $z^{(t)} = z^{(t+1)}$ is highly likely. In this case, the maximum entropy classifier will understandably be strongly biased toward the previous label while mostly ignoring the observations. This "conditional class imbalance" is related to the following issues:

1. The **probability flow problem**, in which the likelihood of all possible successors of an unlikely partial sequence $\{z^{(t)}, 1 \leq t \leq T' < T\}$ must still sum to one and cannot be influenced by future observations that would contradict the current state (Lafferty et al., 2001). Note that multiplying the probability distributions in an HMM without renormalizing (eq. 2.25) allows a proper weighting between the symbolic and acoustic predictors.
2. The **teacher forcing problem**, in which the model is trained in perfect conditions with correct sequence histories $\mathcal{A}^{(t)}$, but does not necessarily learn to recover from past mistakes at test time, nor to accurately describe the likelihood of sequences with incoherent histories.

Several tricks will be described later in this thesis to partially control the label bias problem in RNNs. The safest strategy to avoid it entirely is probably to use a generative model like a DBN or conditional random fields, described next.

2.2.5 Random fields

Random fields (RFs) are undirected graphical models between vectors of random variables that can be observed or latent depending on the context. Contrarily to Bayesian networks described previously, RFs are better suited to naturally represented cyclic rather than causal dependencies. A Markov random field (MRF) is a special case where each variable is directly connected only to its nearest neighbors.

This method has been used to model polyphonic music in a piano-roll representation (Lavrenko and Pickens, 2003), where the symbolic sequence is seen as a bidimensional random field in which the presence of each note depends on the presence of past notes and current notes of lower pitch according to learned patterns. This structure allows to describe within-frame note correlations and short-term temporal evolution; longer-term dependencies remain elusive due to the limited range of the learned patterns and the impossibility to remember information about the sequence history.

2.2.6 Conditional random fields

Conditional random fields (CRFs) (Lafferty et al., 2001) are RFs conditioned on an observed sequence x; this model was specifically designed to overcome the label bias problem when estimating the conditional density of the output $P(z|x)$. Linear chain CRFs, i.e. ones exhibiting Markovian assumptions, are often used in practice to replace HMMs in what is commonly referred to as “full-sequence training” in the speech recognition community (e.g. Mohamed et al., 2010).

The undirected connections between output variables $z^{(t)}$ of the random field define a probability distribution that is only *globally normalized* (LeCun et al., 1998) and thus avoids the probability flow problem mentioned earlier. This allows the current observation to properly influence the distribution of the current output label, even in the case of frequently reoccuring output symbols $z^{(t)} = z^{(t+1)}$. However, it should be noted that gains achieved with full-sequence training compared to an HMM baseline are typically low, e.g. around 0.3% in phone recognition on TIMIT (Mohamed et al., 2010).

2.2.7 Recurrent neural networks

Recurrent neural networks (RNNs) (Rumelhart et al., 1986a) are characterized by their internal memory, or hidden layer, defined by the recurrence relation:

$$h^{(t)} = f(W_{zh}z^{(t)} + W_{hh}h^{(t-1)} + W_{xh}x^{(t)} + b_h) \tag{2.29}$$

where W_{uv} is a weight matrix connecting $u \rightarrow v$, b_h is a bias vector, $f(\cdot)$ is an element-wise non-linearity function, and $h^{(0)}$ is an additional model parameter. Popular choices for f include the logistic sigmoid function $f(a) = (1 + e^{-a})^{-1}$, the hyperbolic tangent $f(a) = \tanh(a)$ and the rectifier non-linearity $f(a) = \max(a, 0)$ (Nair and Hinton, 2010; Glorot et al., 2011a). Note that rectifiers should be used in conjunction with an L1 penalty on the hidden units of an RNN to avoid gradient explosion.

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

$$y^{(t)} = o(W_{hz}h^{(t-1)} + W_{xz}x^{(t)} + b_z) \tag{2.30}$$

where $o(a)$ is the output non-linearity function of an activation vector a and should be as close as possible to the target vector $z^{(t)}$. The prediction $y^{(t)}$ serves to define

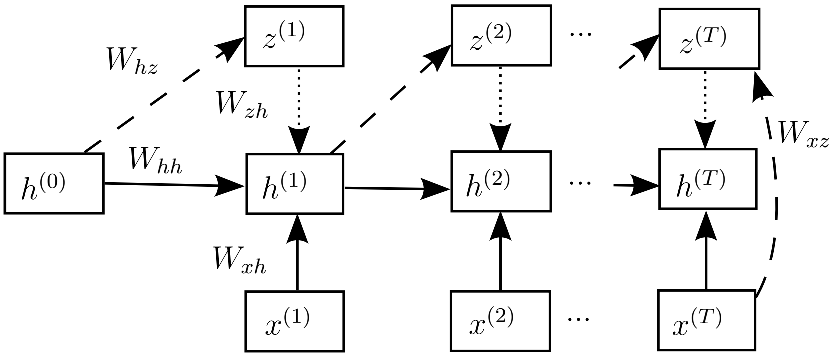


Figure 2.1: Graphical structure of an input/output RNN. Single arrows represent a deterministic function, dotted arrows represent optional connections for temporal smoothing, dashed arrows represent a prediction. The $x \to z$ connections have been omitted for clarity at each time step except the last.

the conditional distribution of $z^{(t)}$ given $\mathcal{A}^{(t)}$ and x used in equation (2.21):

$$\log P(z^{(t)}|\mathcal{A}^{(t)}, x) = -\sum_{j=1}^{N} z_j \log y_j + (1 - z_j)\log(1 - y_j) \qquad (2.31)$$

$$\text{or: } \log P(z^{(t)}|\mathcal{A}^{(t)}, x) = -\sum_{j=1}^{N} z_j \log y_j \qquad (2.32)$$

for multi-label (many-of-N) and multiclass (one-of-N) classification respectively, but other conditional distributions are possible. The RNN graphical structure is depicted in Figure 2.1.

RNNs are commonly trained to predict the next time step given the previous ones and the input, using backpropagation through time (BPTT) (Rumelhart et al., 1986a). Since gradient-based training suffers from various pathologies (Bengio et al., 1994), several strategies will be discussed later in this thesis to help reduce these difficulties, particularly in Chapter 6.

Here are some common extensions and particular cases of the RNN:

1. **Long short term memory** (LSTM) cells (Hochreiter and Schmidhuber, 1997) can increase the range of the captured temporal dependencies by using multiplicative gates to the input (other locations are possible), combined with unit-norm self-connections that impose a constant error flow. This can

be achieved by replacing (2.29) with:

$$h^{(t)} = h^{(t-1)} + f(W_{zh}z^{(t)} + W_{hh}h^{(t-1)} + W_{xh}x^{(t)} + b_h) \circ \sigma(W_{xg}x^{(t)} + b_g), \quad (2.33)$$

where W_{xg}, b_g are the weight and bias input gate parameters and $\circ$ denotes element-wise multiplication. This approach was successful in a number of long-term memorization tasks that were previously "effectively impossible" for stochastic gradient descent.

2. **Temporal smoothing connections** (dotted arrows in Figure 2.1) are the optional connections $z \to h$ that implicitly tie $z^{(t)}$ to its history and encourage coherence between successive output frames, and temporal smoothing in particular. Without temporal smoothing ($W_{zh} = 0$), the $z^{(t)}, 1 \leq t \leq T$ are conditionally independent given x and inference can simply be carried out separately at each time step t.

3. **Input/output connections** are the optional connections $x \to h$ and $x \to z$ that make the RNN model the conditional output distribution given the input $P(z|x)$ for a transduction task. Merely modeling the output distribution $P(z)$ can be achieved by setting $W_{xh} = W_{xz} = 0$.

4. **Time-delay connections (taps)** (El Hihi and Bengio, 1996) can be added in equations (2.29) and (2.30) to supplement the recurrence and the predictions with a direct access to predecessors $x^{(t-\tau)}$, $h^{(t-\tau)}$ and $z^{(t-\tau)}$ for fixed time lags $\tau \geq 1$. This can help the RNN discover temporal dependencies spanning a range τ times longer at the expense of more computation.

5. **Equivalent definitions of the RNN**, e.g. in which $z^{(t-1)} \to h^{(t)} \to z^{(t)}$, can be derived by the change of variable $h'^{(t)} = h^{(t-1)}$, which intuitively corresponds to shifting the hidden layer by one time step to the left in Figure 2.1 while keeping all arrows attached. Alternative formulations should not introduce cyclic dependencies between the $z^{(t)}$.

6. **Bidirectional RNNs** (BRNNs) (Schuster and Paliwal, 1997) are composed of two RNNs with separate hidden units that recurse respectively forward and backward in time; the two hidden layers at time t then predict a properly normalized joint distribution of $z^{(t)}$. Both networks can fully depend on the input x but only one of them can incorporate temporal smoothing connections

to avoid cyclic dependencies in the output random variables $z^{(t)}$. Stacking a CRF on top of a bidirectional RNN can avoid this problem (Yao et al., 2013).

Polyphonic music models based on RNNs typically output only monophonic notes along with predefined chords or other reduced-dimensionality representation (Mozer, 1994; Eck and Schmidhuber, 2002; Franklin, 2006) via equation (2.30). Another possibility to model sequences in the piano-roll representation is to predict independent note probabilities (Martens and Sutskever, 2011), i.e. for which the conditional output distribution $P(z^{(t)}|\mathcal{A}^{(t)}, x)$ factorizes:

$$P(z^{(t)}|\mathcal{A}^{(t)}, x) = \prod_{i=1}^{N} P(z_i^{(t)}|\mathcal{A}^{(t)}, x) \tag{2.34}$$

which is a strong assumption in harmonic music. LSTM cells were also successful at capturing longer-term structure in symbolic music when used in conjunction with time-delay connections aligned on the rhythmic structure (Eck and Lapalme, 2008).

2.2.8 Hierarchical models

Many natural sequences exhibit a multilevel or *hierarchical* structure in which the occurrence of lower-level patterns can itself be described by a higher-level model. For example, musical pieces can often be divided into parts (e.g. verse and chorus), which in turn can be divided into phrases, measures and notes.

The simplest way to incorporate prior knowledge about the hierarchical organization of temporal dependencies is to provide time-delay bypass connections to the hidden units of an RNN as described earlier (El Hihi and Bengio, 1996). The time delays can optionally be aligned with the known temporal structure or follow a geometrical spacing. Another option is to stack multiple interconnected hidden layers in a deep RNN (Schmidhuber, 1992), which is a natural architecture to model hierarchical dependencies (Hermans and Schrauwen, 2013). It is also possible to constrain different subsets of recurrent hidden units to vary in time at different frequencies (El Hihi and Bengio, 1996; Jaeger et al., 2007; Siewert and Wustlich, 2007), the rationale for this approach being that high-level phenomena should vary more slowly than low-level ones. A graphical model with an explicit hierarchical structure has also been designed to model polyphonic music (Paiement et al., 2005).

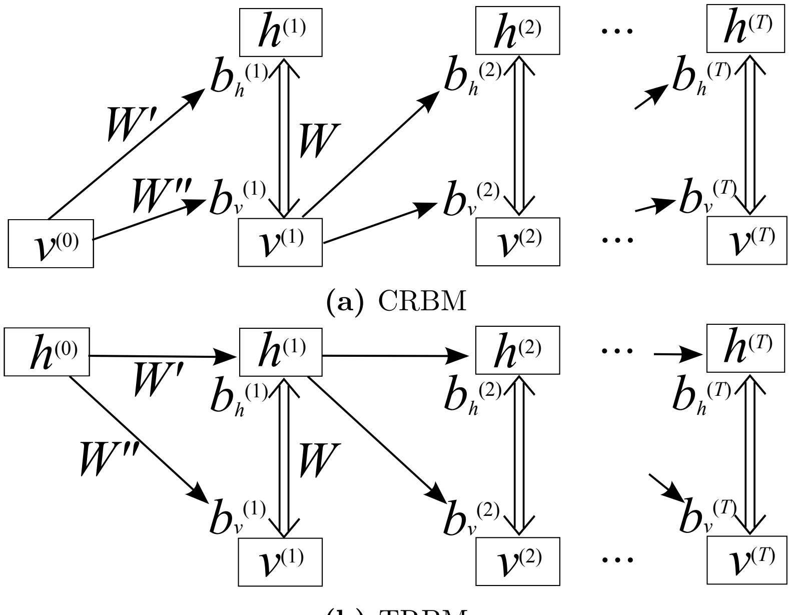
2.2.9 Temporal RBMs

In this section, we wish to exploit the ability of RBMs to represent a complicated distribution for each time step, with parameters that depend on the previous ones, an idea first put forward with conditional RBMs (Taylor et al., 2007). In this model (Figure 2.2a), the biases $b_v^{(t)}, b_h^{(t)}$ of the time-varying RBM describing the conditional distribution $P(v^{(t)}|\mathcal{A}^{(t)})$ as per equation (2.4) depend on the sequence history as a linear function of the previous outputs (for simplicity, only $v^{(t-1)}$ here):

$$b_h^{(t)} = W'v^{(t-1)} + b_h \tag{2.35}$$

$$b_v^{(t)} = W''v^{(t-1)} + b_v \tag{2.36}$$

where $W', W'', W, b_h, b_v, v, v^{(0)}$ are the resulting model parameters. Note that $v^{(t)}$ is the visible layer of the t-th RBM and also represents of the output random variable $z^{(t)}$ in the directed graphical model; the factorization (2.20) applies. Because CRBMs are Markov processes, they cannot represent long-term dependencies.



(a) CRBM

(b) TRBM

Figure 2.2: Comparison of the graphical structures of (a) the CRBM and (b) the TRBM. Single arrows represent a deterministic function, double arrows represent the stochastic hidden-visible connections of an RBM. The RBM biases $b_h^{(t)}$, $b_v^{(t)}$ are a linear function of either $v^{(t-1)}$ or $h^{(t-1)}$.

An extension of conditional RBMs is the temporal RBM (TRBM) (Sutskever and Hinton, 2007) in which the time-varying RBMs are rather conditioned on the past values of the hidden units $h^{(\tau)}, \tau < t$ (only $h^{(t-1)}$ here) as shown in Figure 2.2b. This model is complicated by the fact that h are latent random variables and requires an heuristic training procedure. The recurrent temporal RBM (RTRBM) (Sutskever et al., 2008) is a similar model that allows for exact inference and efficient training by contrastive divergence (CD). The trick is to express the RBM parameters as a function of the *mean-field* value $\hat{h}^{(t)}$ of $h^{(t)}$, i.e. replacing equations (2.35) and (2.36) with:

$$b_h^{(t)} = W'\hat{h}^{(t-1)} + b_h \tag{2.37}$$

$$b_v^{(t)} = W''\hat{h}^{(t-1)} + b_v \tag{2.38}$$

which makes exact inference of $h^{(t)}$ very easy and improves the efficiency of training (Sutskever et al., 2008). Despite its simplicity, this model successfully accounts for several interesting sequences, such as videos of balls bouncing in a box and motion capture data.

Note that the mean-field value of $h^{(t)}$ is deterministic given $\mathcal{A}^{(t+1)}$:

$$\hat{h}^{(t)} = \sigma(Wv^{(t)} + b_h^{(t)}) = \sigma(Wv^{(t)} + W'\hat{h}^{(t-1)} + b_h) \tag{2.39}$$

which is exactly the defining equation of an RNN with hidden units $\hat{h}^{(t)}$ and a sigmoid non-linearity (eq. 2.29). A similar architecture based on the echo state network (ESN) was also recently developed (Schrauwen and Buesing, 2009).

2.3 Non-negative matrix factorization

Non-negative matrix factorization (NMF) is an unsupervised technique to discover parts-based representations underlying non-negative data (Lee and Seung, 1999), i.e. a set of characteristic components that can be combined additively to reconstitute the observations. When applied to the magnitude spectrogram of a polyphonic audio signal, NMF can discover a basis of interpretable recurring sound events and their associated time-varying encodings, or *activities*, that together optimally reconstruct the original spectrogram.

The activities extracted by NMF have proven useful as features for a wide variety of tasks, including polyphonic transcription (Abdallah and Plumbley, 2006; Smaragdis and Brown, 2003; Dessein et al., 2010) and audio source separation (e.g. Virtanen, 2007). Sparsity, temporal and spectral priors have proven useful to enhance the accuracy of multiple pitch estimation (Cont, 2006; Vincent et al., 2010; Fitzgerald et al., 2005). Ordinary NMF is an unsupervised technique, but some supervised variants exploit the availability of ground truth annotations to increase the relevance of the extracted features with respect to a discriminative task (Boulanger-Lewandowski et al., 2012a). A temporal description of the NMF activity matrices can also serve as a useful prior during the decomposition, as discussed in Chapter 14.

An advantage of the NMF decomposition is its inherent ability to infer the time-varying activities from a complex signal in a way similar to the well-known matching pursuit algorithm (Mallat and Zhang, 1993). This mechanism gives first priority to the most salient spectral feature before subtracting the "explained away" part and iteratively repeating this procedure with the residual spectrum. A similar technique is employed in some polyphonic transcription algorithms (Yeh, 2008).

Algorithms for NMF

The NMF method aims to discover an approximate factorization of an input matrix X:

$$\overset{N\times T}{X} \simeq \overset{N\times T}{\Lambda} \equiv \overset{N\times K}{W} \cdot \overset{K\times T}{H} \tag{2.40}$$

where X is the observed magnitude spectrogram with time and frequency dimensions T and N respectively, Λ is the reconstructed spectrogram, W is a dictionary matrix of K basis spectra and H is the activity matrix. Non-negativity constraints $W_{nk} \geq 0, H_{kt} \geq 0$ apply on both matrices. NMF seeks to minimize the *reconstruction error*, a distortion measure between the observed spectrogram X and the reconstruction Λ. A popular choice is the Euclidean distance:

$$C_{LS} \equiv ||X - \Lambda||^2 \tag{2.41}$$

for which we will provide training algorithms although they can be easily generalized to other distortion measures in the β-divergence family (Kompass, 2007).

Minimizing C_{LS} can be achieved by alternating multiplicative updates to H and W (Lee and Seung, 2001):

$$H \leftarrow H \circ \frac{W^T X}{W^T \Lambda} \tag{2.42}$$

$$W \leftarrow W \circ \frac{X H^T}{\Lambda H^T} \tag{2.43}$$

where the $\circ$ operator denotes element-wise multiplication, and division is also element-wise. These updates are guaranteed to decrease the reconstruction error assuming a local minimum is not already reached. While the objective is convex in either W or H separately, it is non-convex in W and H together and thus finding the global minimum is intractable in general.

If we wish to describe the concatenated spectrogram of a large dataset in terms of a single dictionary ($T \gg N$, $T \gg K$), it is more efficient to apply the multiplicative updates to W in mini-batches of X. The corresponding activity mini-batches H should then be either kept in memory between training epochs or reinitialized for each new mini-batch by applying equation (2.42) until convergence, before updates to W can be performed.

Sparsity constraints

In a polyphonic signal with relatively few sound events occurring at any given instant, it is reasonable to assume that active elements H_{ij} should be limited to a small subset of the available basis spectra. To encourage this behavior, a sparsity penalty C_S can be added to the total SNMF objective (Hoyer, 2002):

$$C_S = \lambda |H| \tag{2.44}$$

where $|\cdot|$ denotes the L_1 norm and λ specifies the relative importance of sparsity. In order to eliminate underdetermination associated with the invariance of WH under the transformation $W \to WD$, $H \to D^{-1}H$, where D is a diagonal matrix, we impose the constraint that the basis spectra have unit norm. Equation (2.42) becomes:

$$H \leftarrow H \circ \frac{W^T X}{W^T \Lambda + \lambda} \tag{2.45}$$

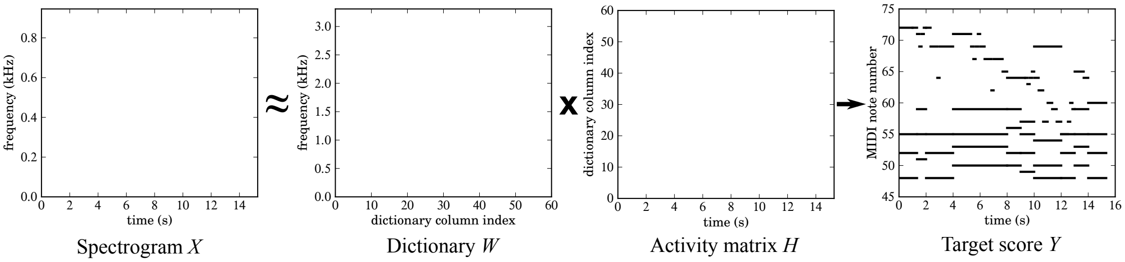


Figure 2.3: Illustration of the sparse NMF decomposition ($\lambda = 0.01$, $\mu = 10^{-5}$) of an excerpt of Drigo's *Serenade*. Using a dictionary W pretrained on a polyphonic piano dataset, the spectrogram X is transformed into an activity matrix H approximating the piano-roll transcription Y. The columns of W were sorted by increasing estimated pitch for visualization.

and the multiplicative update to W (equation 2.43) is replaced by projected gradient descent (Lin, 2007):

$$W \leftarrow W - \mu(\Lambda - X)H^T \tag{2.46}$$

$$W_{nk} \leftarrow \max(W_{nk}, 0), W_{:k} \leftarrow \frac{W_{:k}}{|W_{:k}|} \tag{2.47}$$

where $W_{:k}$ is the k-th column of W, μ is the learning rate and $1 \leq k \leq K$.

NMF for polyphonic transcription

The ability of NMF to extract fundamental note events from a polyphonic mixture makes it an obvious stepping stone for multiple pitch estimation. In the ideal scenario, the dictionary W contains the spectrum profiles of individual notes composing the mix and the activity matrix H approximately corresponds to the ground-truth score. An example of the sparse NMF decomposition of an excerpt of Drigo's *Serenade* using a dictionary pretrained on a simple polyphonic piano dataset is illustrated in Figure 2.3. The dictionary contains mostly monophonic basis spectra that were sorted by increasing estimated pitch for visualization. We also observe a clear similarity between the activity matrix and the target score in a piano-roll representation Y.

There are many options to exploit the NMF decomposition to perform actual multiple pitch estimation. The *dictionary inspection* approach (Abdallah and Plumbley, 2006; Smaragdis and Brown, 2003) consists in estimating the pitch (or lack thereof) of each column of W, which can be done automatically using harmonic combs (Vincent et al., 2010), and to transcribe all pitches for which the associated H_{kt} activities exceed a threshold η:

$$Y_{lt} = 1 \Leftrightarrow \sum_{k|L(k)=l} H_{kt} \geq \eta \tag{2.48}$$

$$\text{or: } Y_{lt} = 1 \Leftrightarrow \max_{k|L(k)=l} H_{kt} \geq \eta \tag{2.49}$$

where $L(k)$ is the estimated pitch label (index) of the k-th basis spectrum. For this method, a new factorization can be performed adaptively for each new piece to analyze, or the dictionary can be pretrained from an extended corpus and kept fixed during testing. Dictionaries can also be constructed from the concatenation of isolated note spectra (Cont, 2006; Dessein et al., 2010).

Another option is to predict each column of Y from the corresponding column of H using a general-purpose multi-label classifier or a set of binary classifiers, one for each label (note) in the designated range. This obviously requires the use of a fixed dictionary and the availability of annotated pieces to train the classifiers. An effective polyphonic transcription system can be built using pre-trained dictionaries and linear support vector machines (SVM) (Poliner and Ellis, 2007) following this principle.

Note that the simple interpretation of the activity matrix as an approximate transcription usually deteriorates when we increase instrumental diversity, pitch range or polyphony. In this case, the supervised methods that we developed in (Boulanger-Lewandowski et al., 2012a) and Chapter 14 can help produce more relevant features with respect to a discriminative task.

2.4 Deep neural 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 (Bengio, 2009). Feature learning with deep neural networks was very successful in a number of audio applications (Mohamed et al., 2009; Hamel and Eck, 2010; Nam et al., 2011; Hinton et al., 2012; Humphrey and Bello, 2012) and is also known to generalize well across different domains (Glorot et al., 2011b; Bengio et al., 2011; Hamel et al., 2013).

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. For example, we can use RBMs (Smolensky, 1986) to model the joint distribution of the previous layer's units in a deep belief network (DBN) (Hinton et al., 2006) (not to be confused with a dynamic Bayesian network), or denoising autoencoders in a similar greedy fashion (Vincent et al., 2008). This pre-training technique usually leads to better generalization than with random initialization (Erhan et al., 2009). Another option is to perform approximate model averaging after randomly dropping out (i.e., setting to 0) some fraction of the hidden units at each layer (Hinton et al., 2012). Deep neural networks can be conveniently constructed and trained using the Theano numerical computing library (Bergstra et al., 2010; Bastien et al., 2012).

Computing representations

The observed vector $x \equiv h_0$ 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:

$$h_{l+1} = f(W_l h_l + b_l) \tag{2.50}$$

for layer l, $0 \leq l < D$ where D is the depth of the network, $f(\cdot)$ is an element-wise non-linearity function and W_l, b_l are respectively the weight and bias parameters for layer l. Popular choices for f include the logistic sigmoid function $f(a) = (1 + e^{-a})^{-1}$, the hyperbolic tangent $f(a) = \tanh(a)$ and the rectifier non-linearity $f(a) = \max(a, 0)$ (Nair and Hinton, 2010; Glorot et al., 2011a). The non-linearity function of the last layer (output) is selected appropriately for the discriminative task at hand. The whole network is finally fine-tuned with respect to a supervised criterion such as the cross-entropy cost:

$$L(v,z) = -\sum_{j=1}^{N} z_j \log y_j + (1-z_j)\log(1-y_j) \quad (2.51)$$

$$\text{or: } L(v,z) = -\sum_{j=1}^{N} z_j \log y_j \quad (2.52)$$

for multi-label (many-of-N) and multiclass (one-of-N) classification respectively, where $y \equiv h_D$ is the prediction obtained at the topmost layer and $z \in \{0,1\}^N$ is a binary vector serving as a target. Note that the target z can have multiple active elements for multi-label classification, but only one for multiclass classification.

Application to sequence labeling

When assigning class labels $z^{(t)}$ to individual frames of an input signal $x^{(t)}$, such as the columns of a magnitude spectrogram, a popular enhancement consists in the use of multiscale aggregated features and time-delay connections to describe context information (Bergstra et al., 2006; Hamel et al., 2012; Dahl et al., 2012). The retained strategy is to provide the network with aggregated features $\bar{x}, \tilde{x}$ (Bergstra et al., 2006) computed over windows of varying sizes L (Hamel et al., 2012) and offsets τ relative to the current time step t:

$$\bar{x}^{(t)} = \Big\{ \sum_{\Delta t=-\lfloor L/2 \rfloor}^{\lfloor (L-1)/2 \rfloor} x^{(t-\tau+\Delta t)}, \forall(L,\tau) \Big\} \quad (2.53)$$

$$\tilde{x}^{(t)} = \Big\{ \sum_{\Delta t=-\lfloor L/2 \rfloor}^{\lfloor (L-1)/2 \rfloor} (x^{(t-\tau+\Delta t)} - \bar{x}^{(t)}_{L,\tau})^2, \forall(L,\tau) \Big\} \quad (2.54)$$

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 a DNN, and enables fair comparisons with temporal models.

2.5 Hessian-free optimization

Hessian-free (HF) optimization is a second-order optimization method that received considerable attention following its successful application to training deep autoencoders (Martens, 2010) and RNNs (Martens and Sutskever, 2011), alleviating the problem of learning long-term dependencies in the latter case. The excellent performance obtained by RNNs on synthetic pathological problems (Hochreiter and Schmidhuber, 1997), text generation (Sutskever et al., 2011) and music sequence modeling (Martens and Sutskever, 2011) motivates its use in this thesis.

The method is derived from the Newton method which seeks to minimize the objective function $f(\theta) : \mathbb{R}^n \to \mathbb{R}$ by approximating it near the current point θ by a quadratic form:

$$f(\theta + \delta_\theta) \simeq f(\theta) + \nabla f(\theta)^{\mathrm{T}} \delta_\theta + \frac{1}{2} \delta_\theta^{\mathrm{T}} B \delta_\theta \tag{2.55}$$

where B approximates the Hessian matrix H. At each training iteration, we minimize the approximation of $f(\theta + \delta_\theta) + \lambda ||\delta_\theta||^2$ where a quadratic damping term prevents straying too far from the current point where the approximation is potentially incorrect.

Hessian-free optimization differs from the Newton method in that the Hessian matrix is never explicitly calculated. Instead, the quadratic form (2.55) is minimized by the successive application of matrix-vector products Bv in the conjugate gradient (CG) algorithm (Shewchuk, 1994), which can be efficiently computed by applying the R operator (Pearlmutter, 1994).

An important modification consists in using for B not H directly, but the Gauss-Newton matrix (Schraudolph, 2002), a positive-definite approximation to the Hessian obtained by sectioning the computational graph of $f(\theta)$ in two parts $f(\theta) \to f(g(\theta))$ where $f(g)$ is convex:

$$\nabla^2_{\theta\theta} f = H \simeq B = \nabla_\theta g^{\mathrm{T}} \nabla^2_{gg} f \nabla_\theta g. \tag{2.56}$$

Our Theano-based implementation (Bergstra et al., 2010) of the HF optimizer that includes all the details explained in (Martens, 2010; Martens and Sutskever, 2011) is available online.[1]

[1] `https://github.com/boulanni/theano-hf`

3 Prologue to First Article

3.1 Article Details

Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription
Nicolas Boulanger-Lewandowski, Yoshua Bengio and Pascal Vincent
Published in *Proceedings of the 29th International Conference on Machine Learning (ICML)* in 2012.

3.2 Context

RNNs are promising candidates to model sequential phenomena due to their ability in principle to represent long-term dependencies and complex temporal behaviors. It was recently shown that Hessian-free optimization could help reduce the vanishing and exploding gradient problems (Martens and Sutskever, 2011; Bengio et al., 1994) and train RNNs more effectively.

In realistic high-dimensional sequences, predicting the next time step is often complicated by the multimodality of the conditional output distribution. That property is especially important in polyphonic music where notes appear together in strongly correlated patterns. The idea of using RBMs to estimate the conditional distributions was first put forward with the so-called temporal RBM (Taylor et al., 2007; Sutskever and Hinton, 2007) and later with the recurrent temporal RBM (RTRBM) (Sutskever et al., 2008). It is also possible to use Gaussian mixtures to model those conditional distributions (Schuster, 1999a).

Most existing models of polyphonic music output only monophonic notes along with predefined chords or other reduced-dimensionality representation (e.g. Mozer, 1994; Eck and Schmidhuber, 2002; Paiement et al., 2009), which makes it difficult to interpret results and compare machine learning algorithms.

Most existing polyphonic transcription algorithms are frame-based and rely exclusively on the audio signal. It has long been known that musicological models can improve purely auditive approaches to music information retrieval (Cemgil, 2004). However, combining these two sources of information is not trivial, with the result that temporal smoothing with an HMM is often the only post-processing involved in state-of-the-art transcription (Nam et al., 2011).

3.3 Contributions

This paper introduces the RNN-RBM, a generalization of the RTRBM that allows more freedom to describe the temporal dependencies involved in high-dimensional sequences. Proposed improvements include a separate layer of recurrent hidden units, the use of pre-training techniques and Hessian-free optimization, and the possibility to substitute conditional RBMs with NADEs.

With extensive experiments carried over four large datasets of symbolic music, we demonstrate that the RNN-RBM consistently outperforms the RTRBM and many other traditional models of polyphonic music in both log-likelihood and frame-level accuracy.

Finally, we present a polyphonic transcription algorithm based on a product of experts between an arbitrary acoustic classifier and our symbolic model. Our method improves transcription accuracy much more than the popular HMM approach.

3.4 Recent Developments

Brakel et al. (2013) recently developed the recurrent energy-based model (REBM) in the context of time-series imputation. The REBM has an architecture similar to the RNN-RBM but is trained to explicitly minimize the reconstruction error of deterministic mean-field predictions with respect to ground-truth missing values.

The four polyphonic music datasets prepared for this paper and the associated task of sequence prediction in the piano-roll representation were later taken as benchmarks by other authors attempting to improve learning in RNNs (Pascanu et al., 2012, 2013, 2014; Bayer et al., 2014). Our architecture still holds the state of the art on the four datasets in both log-likelihood and accuracy by a significant margin. In our opinion, the conditional distribution esimators of the RNN-RBM are an essential component in the realistic modeling of high-dimensional sequences that cannot fully be accounted for simply by increasing the flexibility of the recurrence relation or the efficiency of optimization.

The transcription algorithm presented in this paper is based on a product of experts and a greedy chronological search. In Chapter 8, we extend this approach with a comprehensive input/output architecture that combines the acoustic and symbolic models, and a global inference procedure based on high-dimensional beam search.

4 Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription

We investigate the problem of modeling symbolic sequences of polyphonic music in a completely general piano-roll representation. We introduce a probabilistic model based on distribution estimators conditioned on a recurrent neural network that is able to discover temporal dependencies in high-dimensional sequences. Our approach outperforms many traditional models of polyphonic music on a variety of realistic datasets. We show how our musical language model can serve as a symbolic prior to improve the accuracy of polyphonic transcription.

4.1 Introduction

Modeling sequences is an important area of machine learning since many naturally occurring phenomena such as music, speech, or human motion are inherently sequential. Complex sequences are *non-local* in that the impact of a factor localized in time can be delayed by an arbitrarily long time-lag. For example, musical patterns or themes appearing at the beginning of a piece are often repeated towards the end. Recurrent neural networks (RNN) (Rumelhart et al., 1986a) incorporate an internal memory that can, in principle, summarize the entire sequence history. This property makes them well suited to represent long-term dependencies, but it is nevertheless a challenge to train them efficiently by gradient-based optimization (Bengio et al., 1994). It was recently shown that training RNNs via Hessian-free (HF) optimization could help reduce these difficulties (Martens and Sutskever, 2011).

Many sequences of interest are over high-dimensional objects, such as images in video, short-term spectra in audio music, tuples of notes in musical scores, or words in text. In these cases, simply predicting the expected value at the next time step given the observed values of the previous time steps is not satisfying. With such high-dimensional objects at each time step, the conditional distribution is very often multi-modal, and we would strongly prefer our models of such sequences to predict the *conditional distribution* of the next time step given previous time steps. For the case of polyphonic music, it is obvious that the occurrence of a particular note at a particular time modifies considerably the probability with which other notes may occur at the same time. In other words, notes appear together in correlated patterns, or *simultaneities*, that cannot be conveniently described by a typical RNN architecture designed for the multiclass classification task, for example, because enumerating all configurations of the variable to predict would be very expensive. This difficulty motivates energy-based models which allow us to express the negative log-likelihood of a given configuration by an arbitrary energy function, among which the restricted Boltzmann machine (RBM) (Smolensky, 1986) has become notorious.

In this context, we wish to exploit the ability of RBMs to represent a complicated distribution for each time step, with parameters that depend on the previous ones, an idea first put forward with the so-called temporal RBM (Taylor et al., 2007; Sutskever and Hinton, 2007) which is trained via a heuristic procedure. Combining the desirable characteristics of RNNs and RBMs has proven to be non-trivial. The recurrent temporal RBM (RTRBM) (Sutskever et al., 2008) is a similar model that allows for exact inference and efficient training by contrastive divergence (CD). Despite its simplicity, this model successfully accounts for several interesting sequences. A similar architecture based on the echo state network was also recently developed (Schrauwen and Buesing, 2009). In this work, we demonstrate that the RTRBM outperforms many traditional models of polyphonic music, and we introduce a generalization of the RTRBM, called the RNN-RBM, that allows more freedom to describe the temporal dependencies involved.

More precisely, we will consider sequences of *symbolic* music, i.e. represented by the explicit timing, pitch, velocity and instrumental information typically contained in a score or a MIDI file rather than more complex, acoustically rich audio signals. Musical models mostly focus on the basic components of western music, harmony and rhythm, and are trained to predict the pattern of notes (simultaneities) to be played together in the next time interval, given the previous ones. Two elements characterize the qualitative performance of a model: temporal dependencies and chord conditional distributions. While most existing models output only monophonic notes along with predefined chords or other reduced-dimensionality representation (e.g. Mozer, 1994; Eck and Schmidhuber, 2002; Paiement et al., 2009), we aim to model unconstrained polyphonic music in the piano-roll representation, i.e. as a binary matrix specifying precisely which notes occur at each time step. Despite ignoring dynamics and other score annotations, this task represents a well-defined framework to improve machine learning algorithms and is directly applicable to polyphonic transcription.

The objective of polyphonic transcription is to determine the underlying notes of a polyphonic audio signal without access to its score. Human experts approach this difficult problem by giving importance to what they expect to hear rather than exclusively to what is present in the actual signal. Most existing transcription algorithms are frame-based and rely exclusively on the audio signal, even though some approaches employ rudimentary musicological constraints (e.g. Li and Wang, 2007). It has long been known that, in the same way that natural language models tremendously improve the performance of speech recognition systems, *musical language models* can improve purely auditive approaches to music information retrieval (Cemgil, 2004). However, combining these two sources of information is not trivial, with the result that temporal smoothing with an HMM is often the only post-processing involved in state-of-the-art transcription (Nam et al., 2011). We will show how to enrich an arbitrary transcription algorithm (under basic assumptions) to include the advice of an expert trained on symbolic sequences. Using our hybrid approach, we can improve transcription accuracy (Bay et al., 2009) much more than the popular HMM approach.

The remainder of the paper is organized as follows. In Sections 4.2, 4.3 and 4.4 we introduce the RBM, the RTRBM and the RNN-RBM architectures. In Section 4.5 we validate our model on benchmark datasets. In Section 4.6 we present our results on musical sequences, and we detail our hybrid transcription approach in Section 4.7.

4.2 Restricted Boltzmann machines

An RBM is an energy-based model where the joint probability of a given configuration of the visible vector v (inputs) and the hidden vector h is:

$$P(v,h) = \exp(-b_v^{\mathrm{T}} v - b_h^{\mathrm{T}} h - h^{\mathrm{T}} W v)/Z \tag{4.1}$$

where b_v, b_h and W are the model parameters and Z is the usually intractable partition function. When the vector v is given, the hidden units h_i are conditionally independent of one another, and vice versa:

$$P(h_i = 1|v) = \sigma(b_h + Wv)_i \tag{4.2}$$

$$P(v_j = 1|h) = \sigma(b_v + W^{\mathrm{T}} h)_j \tag{4.3}$$

where $\sigma(x) \equiv (1 + e^{-x})^{-1}$ is the element-wise logistic sigmoid function. The marginalized probability of v is related to the free-energy $F(v)$ by $P(v) \equiv e^{-F(v)}/Z$:

$$F(v) = -b_v^{\mathrm{T}} v - \sum_i \log(1 + e^{b_h + Wv})_i \tag{4.4}$$

Inference in RBMs consists of sampling the h_i given v (or the v_j given h) according to their conditional Bernoulli distribution (eq. 4.2). Sampling v from the RBM can be performed efficiently by block Gibbs sampling, i.e. by performing k alternating steps of sampling $h|v$ and $v|h$. The gradient of the negative log-likelihood of an input vector $v^{(l)}$ involves two opposing terms, called the positive and negative phase:

$$\frac{\partial(-\log P(v^{(l)}))}{\partial\Theta} = \frac{\partial F(v^{(l)})}{\partial\Theta} - \frac{\partial(-\log Z)}{\partial\Theta} \tag{4.5}$$

where $\Theta \equiv \{b_v, b_h, W\}$. The second term can be estimated by a single sample $v^{(l)*}$ obtained from a k-step Gibbs chain starting at $v^{(l)}$:

$$\frac{\partial(-\log P(v^{(l)}))}{\partial\Theta} \simeq \frac{\partial F(v^{(l)})}{\partial\Theta} - \frac{\partial F(v^{(l)*})}{\partial\Theta}. \tag{4.6}$$

resulting in the well-known contrastive divergence (CD$_k$) algorithm (Hinton, 2002).

The neural autoregressive distribution estimator (NADE) (Larochelle and Murray, 2011) is a tractable model inspired by the RBM and specializing (with tying constraints) an earlier model for the joint distribution of high-dimensional variables (Bengio and Bengio, 2000). NADE is similar to a fully visible sigmoid belief network in that the conditional probability distribution of a visible unit v_j is expressed as a nonlinear function of $v_k, \forall k < j$. In the following discussion, one can substitute RBMs with NADEs by replacing equation (4.6) with the exact gradient defined in (Larochelle and Murray, 2011) where the biases are set to $b = v_b^{(t)}$, $c = v_h^{(t)}$. The advantages of a tractable distribution estimator will become obvious when used as part of sequential models.

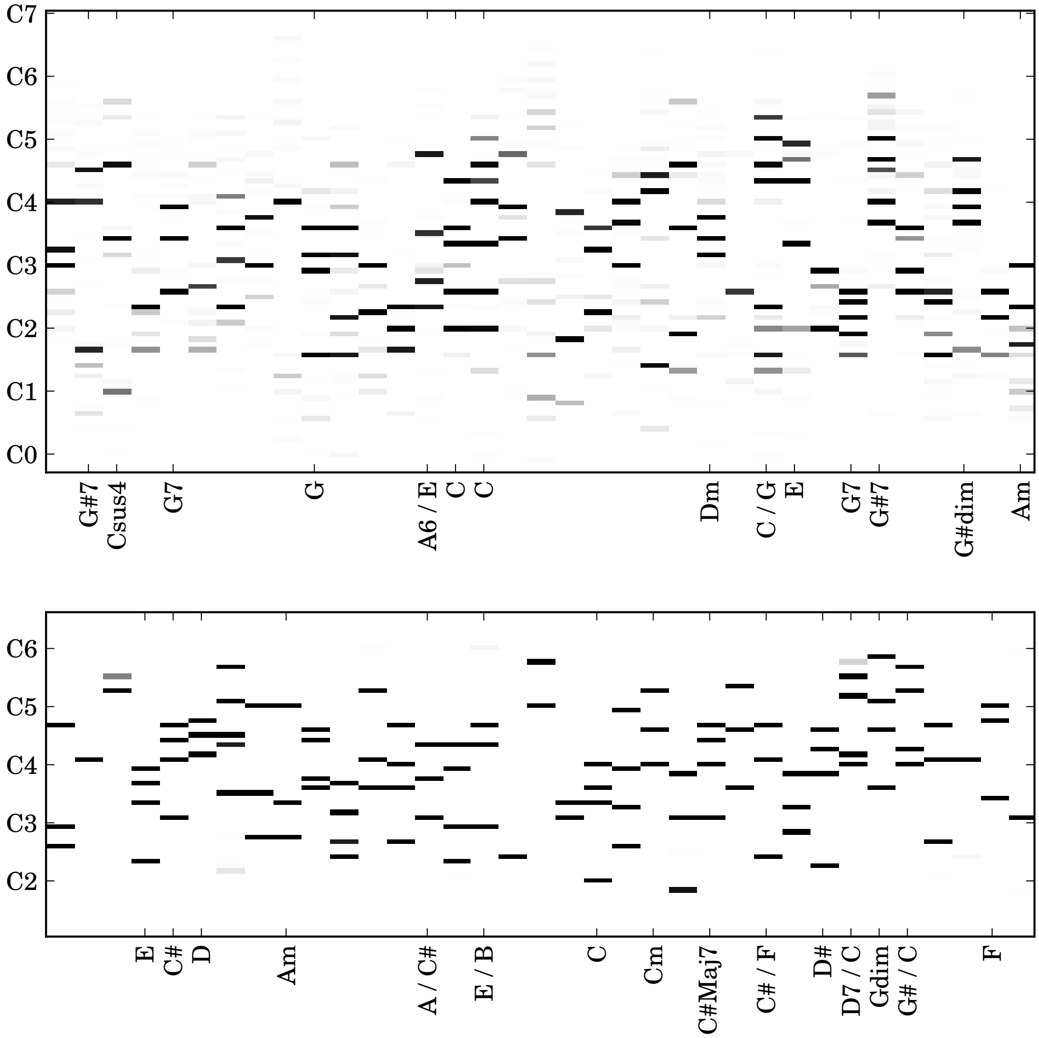


Figure 4.1: Mean-field samples of an RBM trained on the Piano-midi (top) and JSB chorales (bottom) datasets. Each column is a sample vector of notes, with a chord label where the analysis is unambiguous.

Figure 4.1 presents mean-field samples $P(v_j = 1|h^*)$, where $h^* \sim P(h)$, drawn from RBMs trained on a diverse collection of classical piano music (top) and on the four-part chorales by J. S. Bach (bottom), along with chord labels where the analysis is unambiguous. It is obvious that for the diverse collection, each sample has some room for additional melody notes with probabilities depending on the harmonic context (grey), whereas for JSB chorales, the simultaneities are taken from a more restricted pool and the samples are more clear-cut. This mechanism makes sense musically and the fact that RBMs can adapt to various styles will be useful for the following.

4.3 The RTRBM

The RTRBM (Sutskever et al., 2008) is a sequence of conditional RBMs (one at each time step) whose parameters $b_v^{(t)}, b_h^{(t)}, W^{(t)}$ are time-dependent and depend on the sequence history at time t, denoted $\mathcal{A}^{(t)} \equiv \{v^{(\tau)}, \hat{h}^{(\tau)} | \tau < t\}$ where $\hat{h}^{(t)}$ is the mean-field value of $h^{(t)}$. Its graphical structure is depicted in Figure 4.2a. The RTRBM is formally defined by its joint probability distribution:

$$P(\{v^{(t)}, h^{(t)}\}) = \prod_{t=1}^{T} P(v^{(t)}, h^{(t)} | \mathcal{A}^{(t)}) \tag{4.7}$$

where $P(v^{(t)}, h^{(t)} | \mathcal{A}^{(t)})$ is the joint probability (eq. 4.1) of the t^{th} RBM whose parameters are defined below (eq. 4.8 and 4.9).

While all the parameters of the RBMs can depend on the previous time steps, we will consider the case where only the biases depend on $\hat{h}^{(t-1)}$:

$$b_h^{(t)} = b_h + W' \hat{h}^{(t-1)} \tag{4.8}$$

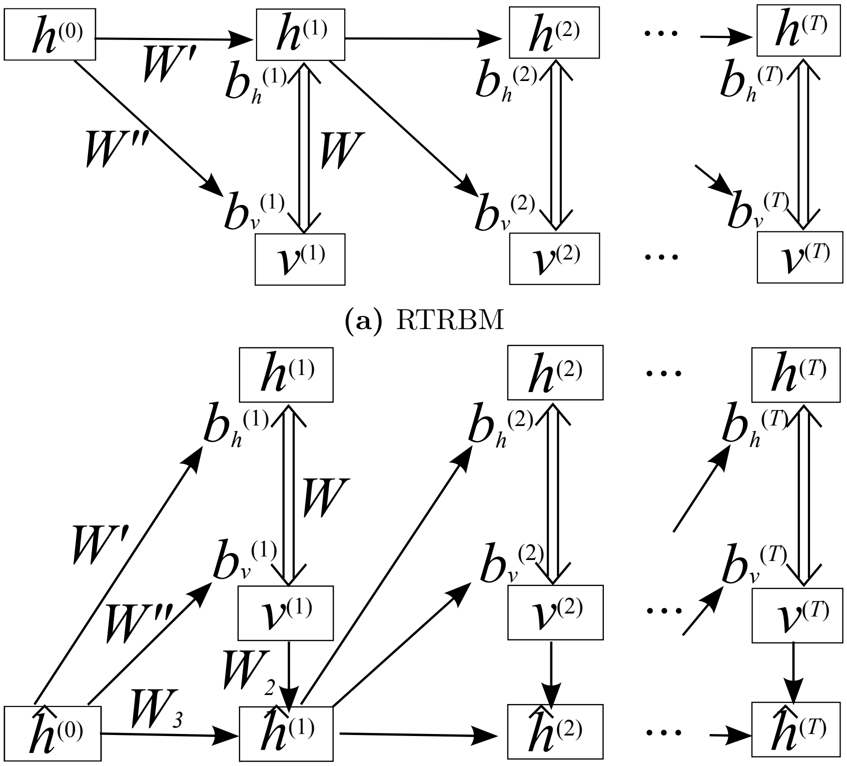
$$b_v^{(t)} = b_v + W'' \hat{h}^{(t-1)} \tag{4.9}$$

which gives the RTRBM six parameters: $W, b_v, b_h, W', W'', \hat{h}^{(0)}$. The general case is derived in a similar manner.

While the hidden units $h^{(t)}$ are binary during inference and sampling, it is the *mean-field* value $\hat{h}^{(t)}$ that is transmitted to its successors (see eq. 4.10). This important distinction makes exact inference of the $\hat{h}^{(t)}$ very easy and improves the efficiency of training (Sutskever et al., 2008):

$$\hat{h}^{(t)} = \sigma(Wv^{(t)} + b_h^{(t)}) = \sigma(Wv^{(t)} + W'\hat{h}^{(t-1)} + b_h) \tag{4.10}$$

is obtained directly from equations (4.2) and (4.8). Note that equation (4.10) is exactly the defining equation of a single-layer RNN with hidden units $\hat{h}^{(t)}$.



(a) RTRBM

(b) RNN-RBM

Figure 4.2: Comparison of the graphical structures of (a) the RTRBM and (b) the single-layer RNN-RBM. Single arrows represent a deterministic function, double arrows represent the stochastic hidden-visible connections of an RBM. The upper half of the RNN-RBM is the RBM stage while the lower half is a RNN with hidden units $\hat{h}^{(t)}$. The RBM biases $b_h^{(t)}$, $b_v^{(t)}$ are a linear function of $\hat{h}^{(t-1)}$.

4.4 The RNN-RBM

The RTRBM can be understood as a sequence of conditional RBMs whose parameters are the output of a deterministic RNN, with the constraint that the hidden units must describe the conditional distributions *and* convey temporal information. This constraint can be lifted by combining a full RNN with distinct hidden units $\hat{h}^{(t)}$ with the RTRBM graphical model as shown in Figure 4.2b. We call this model the RNN-RBM. The joint probability distribution of the RNN-RBM is also given by equation (4.7), but with $\hat{h}^{(t)}$ defined arbitrarily, here as per equation (4.11).

For simplicity, we consider the RBM parameters to be $W, b_v^{(t)}, b_h^{(t)}$ (i.e. only the biases are variable) and a single-layer RNN (bottom portion of Fig. 4.2b) whose hidden units $\hat{h}^{(t)}$ are only connected to their direct predecessor $\hat{h}^{(t-1)}$ and to $v^{(t)}$ by the relation:

$$\hat{h}^{(t)} = \sigma(W_2 v^{(t)} + W_3 \hat{h}^{(t-1)} + b_{\hat{h}}). \tag{4.11}$$

The RBM portion of the RNN-RBM (upper portion of Fig. 4.2b) is otherwise exactly the same as its RTRBM counterpart. This gives the single-layer RNN-RBM nine parameters: $W, b_v, b_h, W', W'', \hat{h}^{(0)}, W_2, W_3, b_{\hat{h}}$.

The training algorithm is slightly different than for the RTRBM since the mean-field values of the $h^{(t)}$ are now distinct from $\hat{h}^{(t)}$. An iteration of training is based on the following general scheme:

1. Propagate the current values of the hidden units $\hat{h}^{(t)}$ in the RNN portion of the graph using (4.11),
2. Calculate the RBM parameters that depend on the $\hat{h}^{(t)}$ (eq. 4.8 and 4.9) and generate the negative particles $v^{(t)*}$ using k-step block Gibbs sampling,
3. Use CD$_k$ to estimate the log-likelihood gradient (eq. 4.6) with respect to W, $b_v^{(t)}$ and $b_h^{(t)}$,
4. Propagate the estimated gradient with respect to $b_v^{(t)}, b_h^{(t)}$ backward through time (BPTT) (Rumelhart et al., 1986a) to obtain the estimated gradient with respect to the RNN parameters.

This procedure can be adapted to any RNN architecture and conditional distribution estimator assuming the RNN provides the estimator's parameters (step 2) and can be trained based on a stochastic gradient signal on those parameters (obtained in step 3). The RNN-NADE, obtained by substituting NADEs for RBMs, allows for exact gradient computation.

Note that the single-layer RNN-RBM is a generalization of the RTRBM and reduces to this simpler model by setting $W_2 = W$, $W_3 = W'$ and $b_{\hat{h}} = b_h$ in equations (4.10) and (4.11). The RTRBM was not gaining computationally from sharing these connections, hence untying them does not make it slower. In practice, the ability to distinguish between the number of hidden units h and $\hat{h}$ allows to scale RBMs to several hundred hidden units while keeping the RNNs to their (typically smaller) optimal size, improving performance.

4.4.1 Initialization strategies

Initialization strategies based on unsupervised pretraining of each layer have been shown to be important both for supervised and unsupervised training of deep architectures (Bengio, 2009). A recurrent network corresponds to a very deep architecture when unfolded in time, and indeed we find that pretraining can clearly affect the overall performance of both the RTRBM and the RNN-RBM. To ensure the quality of the learned weight matrices, we found that initializing the W, b_v and b_h parameters from a trained RBM yields less noisy filters. The hidden-to-bias weights W', W'' can then be initialized to small random values, such that the sequential model will initially behave like independent RBMs, eventually departing from that state.

In order to capture better temporal dependencies, we initialize the $W_2, W_3, b_{\hat{h}}, W'', b_v, \hat{h}^{(0)}$ parameters of the RNN-RBM from an RNN trained with the cross-entropy cost:

$$L(\{v^{(t)}\}) = \frac{1}{T} \sum_{t=1}^{T} \sum_{j=1}^{n_v} -v_j^{(t)} \log y_j^{(t)} - (1 - v_j^{(t)}) \log(1 - y_j^{(t)}) \tag{4.12}$$

where $y^{(t)} = \sigma(b_v^{(t)})$ and equations (4.9) and (4.11) hold. This deterministic objective allows the use of a second-order optimization method for pretraining of the RNN. Note that the RTRBM could use this strategy to initialize $W, W', b_v, b_h, W'', \hat{h}^{(0)}$, but in practice we have found the initialization from an RBM more important.

4.4.2 Details of the BPTT algorithm

Suppose we want to minimize the negative log-likelihood cost $C \equiv -\log P(\{v^{(t)}\})$. The gradient of C with respect to the parameters of the conditional RBMs can be estimated by CD using equations (4.4) and (4.6):

$$\frac{\partial C}{\partial b_v^{(t)}} \simeq v^{(t)*} - v^{(t)} \tag{4.13}$$

$$\frac{\partial C}{\partial W} \simeq \sum_{t=1}^{T} \sigma(Wv^{(t)*} - b_h^{(t)})v^{(t)*\mathrm{T}} - \sigma(Wv^{(t)} - b_h^{(t)})v^{(t)\mathrm{T}} \tag{4.14}$$

$$\frac{\partial C}{\partial b_h^{(t)}} \simeq \sigma(Wv^{(t)*} - b_h^{(t)}) - \sigma(Wv^{(t)} - b_h^{(t)}). \tag{4.15}$$

The gradient then back-propagates through the hidden-to-bias parameters (eq. 4.8 and 4.9):

$$\frac{\partial C}{\partial W'} = \sum_{t=1}^{T} \frac{\partial C}{\partial b_h^{(t)}} \hat{h}^{(t-1)\mathrm{T}} \tag{4.16}$$

$$\frac{\partial C}{\partial W''} = \sum_{t=1}^{T} \frac{\partial C}{\partial b_v^{(t)}} \hat{h}^{(t-1)\mathrm{T}} \tag{4.17}$$

$$\frac{\partial C}{\partial b_h} = \sum_{t=1}^{T} \frac{\partial C}{\partial b_h^{(t)}} \text{ and } \frac{\partial C}{\partial b_v} = \sum_{t=1}^{T} \frac{\partial C}{\partial b_v^{(t)}}. \tag{4.18}$$

For the single-layer RNN-RBM, the BPTT recurrence relation follows from (4.11):

$$\begin{aligned}\frac{\partial C}{\partial \hat{h}^{(t)}} &= W_3 \frac{\partial C}{\partial \hat{h}^{(t+1)}} \hat{h}^{(t+1)}(1 - \hat{h}^{(t+1)}) \\ &\quad + W' \frac{\partial C}{\partial b_h^{(t+1)}} + W'' \frac{\partial C}{\partial b_v^{(t+1)}}\end{aligned} \tag{4.19}$$

for $0 \leq t < T$ ($\hat{h}^{(0)}$ being a parameter of the model) and $\partial C / \partial \hat{h}^{(T)} = 0$. Formulas for the remaining RNN-RBM parameters are:

$$\frac{\partial C}{\partial b_{\hat{h}}} = \sum_{t=1}^{T} \frac{\partial C}{\partial \hat{h}^{(t)}} \hat{h}^{(t)}(1 - \hat{h}^{(t)}) \tag{4.20}$$

$$\frac{\partial C}{\partial W_3} = \sum_{t=1}^{T} \frac{\partial C}{\partial \hat{h}^{(t)}} \hat{h}^{(t)}(1 - \hat{h}^{(t)}) \hat{h}^{(t-1)\mathrm{T}} \tag{4.21}$$

$$\frac{\partial C}{\partial W_2} = \sum_{t=1}^{T} \frac{\partial C}{\partial \hat{h}^{(t)}} \hat{h}^{(t)}(1 - \hat{h}^{(t)}) v^{(t)\mathrm{T}}. \tag{4.22}$$

4.5 Baseline experiments

In this section, we compare the performance of the RTRBM with the RNN-RBM on two baseline datasets: bouncing balls videos and motion capture data (Sutskever et al., 2008). We use the mean frame-level squared prediction error as a basis of comparison. The prediction of the t^{th} conditional RBM is performed by 50 steps of block Gibbs sampling starting at $v^{(t-1)}$ and hoping to reconstruct $v^{(t)}$ optimally.

The bouncing ball videos dataset [1] is based on a simulation of balls bouncing in a box (Sutskever and Hinton, 2007). The generated videos are of length $T = 128$ and of resolution 15×15 pixels in the $[0, 1]$ interval, which makes binary RBMs (eq. 4.1) well suited for this task. With up to 300 hidden units and an initial learning rate of 0.01, we obtain a squared prediction error of 2.11 for the RTRBM and 0.96 for the RNN-RBM, i.e. **less than half the error**. The receptive fields (weights) of the first 48 hidden units $h^{(t)}$ (RNN-RBM) are plotted in Figure 4.3. Localized edge detectors are apparent in nearly all the learned filters.

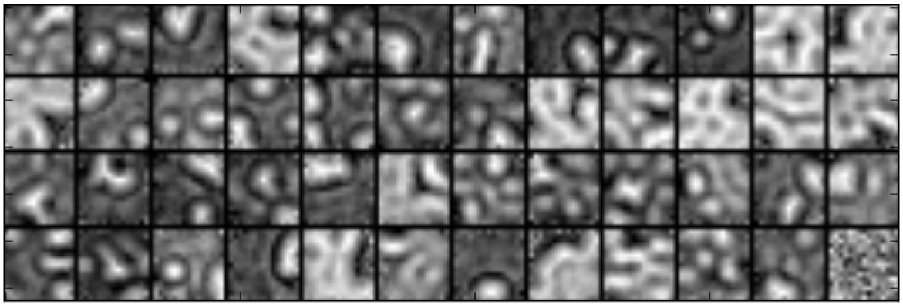


Figure 4.3: Receptive fields of 48 hidden units of an RNN-RBM trained on the bouncing balls dataset. Each square shows the input weights of a hidden unit as an image.

[1] `www.cs.utoronto.ca/~ilya/code/2008/RTRBM.tar`

The human motion capture dataset[2] is represented by a sequence of joint angles, translations and rotations of the base of the spine in an exponential-map parameterization (Hsu et al., 2005; Taylor et al., 2007). Since the data consists of 49 real values per time step, we use the Gaussian RBM variant (Welling et al., 2005) for this task. We use up to 450 hidden units and an initial learning rate of 0.001. The mean squared prediction test error is 20.1 for the RTRBM and **reduced substantially to 16.2 for the RNN-RBM**.

4.6 Modeling sequences of polyphonic music

In this section, we show results with main application of interest for this paper: probabilistic modeling of sequences of polyphonic music. We report our experiments on four datasets of varying complexity converted to our input format.

Piano-midi.de is a classical piano MIDI archive that was split according to Poliner and Ellis (2007).

Nottingham is a collection of 1200 folk tunes[3] with chords instantiated from the ABC format.

MuseData is an electronic library of orchestral and piano classical music from CCARH[4].

JSB chorales refers to the entire corpus of 382 four-part harmonized chorales by J. S. Bach with the split of Allan and Williams (2005).

Each dataset contains at least 7 hours of polyphonic music and the total duration is approximately 67 hours. The polyphony (number of simultaneous notes) varies from 0 to 15 and the average polyphony is 3.9. We use an input of 88 binary visible units that span the whole range of piano from A0 to C8 and temporally aligned on an integer fraction of the beat (quarter note). Consequently, pieces with different time signatures will not have their measures start at the same interval. Although it is not strictly necessary, learning is facilitated if the sequences are transposed in a common tonality (e.g. C major/minor) as preprocessing.

[2] `people.csail.mit.edu/ehsu/work/sig05stf`

[3] `ifdo.ca/~seymour/nottingham/nottingham.html`

[4] `www.musedata.org`

In addition to the models previously described, we evaluate the following commonly used methods:

- The simplest baseline model consists in outputting a Gaussian density centered on the previous frame $\mu = v^{(t-1)}$ and learned covariance Σ.
- N-grams simulate the evolution of note simultaneities as an $(N-1)^{\text{th}}$-order Markov chain. We use add-p or Gaussian smoothing and back-off.
- Note N-grams model each note independently by a binary N-gram, possibly with shared parameters (IID).
- An interesting model for chorales harmonisation (Allan and Williams, 2005) has been adapted to serve as a generative model. It can only be evaluated on the JSB chorales dataset.
- The 'random fields' approach of Lavrenko and Pickens (2003) is a type of fully visible sigmoid belief network with learned connectivity.
- Other common methods include Gaussian mixture models (GMM), hidden Markov models (HMM) using GMM indices as their state, and multilayer perceptrons (MLP) with the last n time steps as input.

The log-likelihood (LL) and expected frame-level accuracy (ACC) (Bay et al., 2009) of the symbolic models are presented in Table 4.1. We estimate the partition function of each conditional RBM by 100 runs of annealed importance sampling (Salakhutdinov and Murray, 2008). We make a few key observations:

- The complexity of the dataset, such as the simplistic chord accompaniment of Nottingham and the redundant style of four-part chorales by a single composer, in comparison with diverse piano and orchestral music, is clearly reflected in the obtained log-likelihoods and accuracies.
- N-gram models (optimal $N^* = 2$) perform reasonably well for simple datasets but fail in more realistic settings due to the increased data sparsity. In this case, note N-grams ($N^* \in [8, 14]$) are a better alternative albeit ignoring harmonic dependencies. This inherent trade-off in traditional polyphonic music models can be addressed robustly by the RNN-based models, that perform better on a range of datasets.
- The harmonisation model of Allan and Williams (2005), tailored to the specific style of four-part chorales, requires annotated harmonic symbols and yet performs relatively poorly compared to our best performer. Similarly to the GMM + HMM, this model is penalized by the limited history of the

Table 4.1: Log-likelihood and expected accuracy for various musical models in the symbolic prediction task. The double line separates frame-level models (above) and models with a temporal component (below).

Model	Piano-midi.de		Nottingham		MuseData		JSB chorales	
	LL	ACC %	LL	ACC %	LL	ACC %	LL	ACC %
Random	-61.00	3.35	-61.00	4.53	-61.00	3.74	-61.00	4.42
1-Gram (Add-p)	-27.64	4.85	-5.94	22.76	-19.03	6.67	-12.22	16.80
1-Gram (Gaussian)	-10.79	6.04	-5.30	21.31	-10.15	7.87	-7.56	17.41
Note 1-Gram	-11.05	5.80	-10.25	19.87	-11.51	7.72	-11.06	15.25
Note 1-Gram (IID)	-12.90	2.51	-16.24	3.56	-14.06	2.82	-15.93	3.51
GMM	-15.84	5.08	-7.87	22.62	-12.20	7.37	-11.90	15.84
RBM	-10.17	5.63	-5.25	5.81	-9.56	8.19	-7.43	4.47
NADE	-10.28	5.82	-5.48	22.67	-10.06	7.65	-7.19	17.88
Previous + Gaussian	-12.48	25.50	-8.41	55.69	-12.90	25.93	-19.00	18.36
N-Gram (Add-p)	-46.04	7.42	-6.50	63.45	-35.22	10.47	-29.98	24.20
N-Gram (Gaussian)	-12.22	10.01	-3.16	65.97	-10.59	16.15	-9.74	28.79
Note N-Gram	-7.50	26.80	-4.54	62.49	-7.91	26.35	-10.26	20.34
GMM + HMM	-15.30	7.91	-6.17	59.27	-11.17	13.93	-11.89	19.24
(Allan and Williams, 2005)	–	–	–	–	–	–	-9.24	16.32
(Lavrenko and Pickens, 2003)	-9.05	18.37	-5.44	55.34	-9.87	18.39	-8.78	22.93
MLP	-8.13	20.29	-4.38	63.46	-7.94	25.68	-8.70	30.41
RNN	-8.37	19.33	-4.46	62.93	-8.13	23.25	-8.71	28.46
RNN (HF)	-7.66	23.34	-3.89	66.64	-7.19	30.49	-8.58	29.41
RTRBM	-7.36	22.99	-2.62	75.01	-6.35	30.85	-6.35	30.17
RNN-RBM	**-7.09**	**28.92**	**-2.39**	**75.40**	-6.01	**34.02**	-6.27	**33.12**
RNN-NADE	-7.48	20.69	-2.91	64.95	-6.74	24.91	-5.83	32.11
RNN-NADE (HF)	**-7.05**	23.42	**-2.31**	71.50	**-5.60**	32.60	**-5.56**	32.50

HMM and by the difficulty to generalize to new chord voicings in a principled manner.

- In accordance with earlier results (Martens and Sutskever, 2011), the use of HF significantly helps the density estimation and prediction performance of RNNs (eq. 4.12) which would otherwise perform worse than simpler MLPs. This motivates our strategy of pretraining the RNN layer of an RNN-RBM via HF.
- In addition to the distinct recurrent hidden units $\hat{h}^{(t)}$ that convey temporal information more freely, and the fact that suitable learning rates can be specified differently for the RNN and the RBM parts, pretraining the W_2, W_3 and $b_{\hat{h}}$ parameters can have the most impact on the RNN-RBM prediction performance. Figure 4.4 clearly demonstrates the importance of pretraining and finetuning the RNN and the additional advantage of using HF.
- Although frame-level NADEs are slightly less powerful than RBMs, their desirable properties make the combined RNN-NADE model the most robust distribution estimator. We believe this is due to their tractable distribution, for two reasons. First, CD may not be ideally suited for conditional RBMs with slowly-mixing Gibbs chains (Mnih et al., 2011), a non-issue for exact-gradient models. Secondly, the joint sequential model, and not only the RNN portion, can benefit from second-order optimization as can be seen from the last two rows of Table 4.1.

We evaluate our models qualitatively by generating sample sequences, provided on the authors' website[5], and discussed here. While note correlations are obviously neglected in the simpler models (sequence 2), RBM-based models learned basic harmony rules (sequence 3), melody lines (sequences 4, 8) and local temporal coherence (sequence 5). However, long-term structure and musical meter remain elusive.

[5] www-etud.iro.umontreal.ca/~boulanni/icml2012

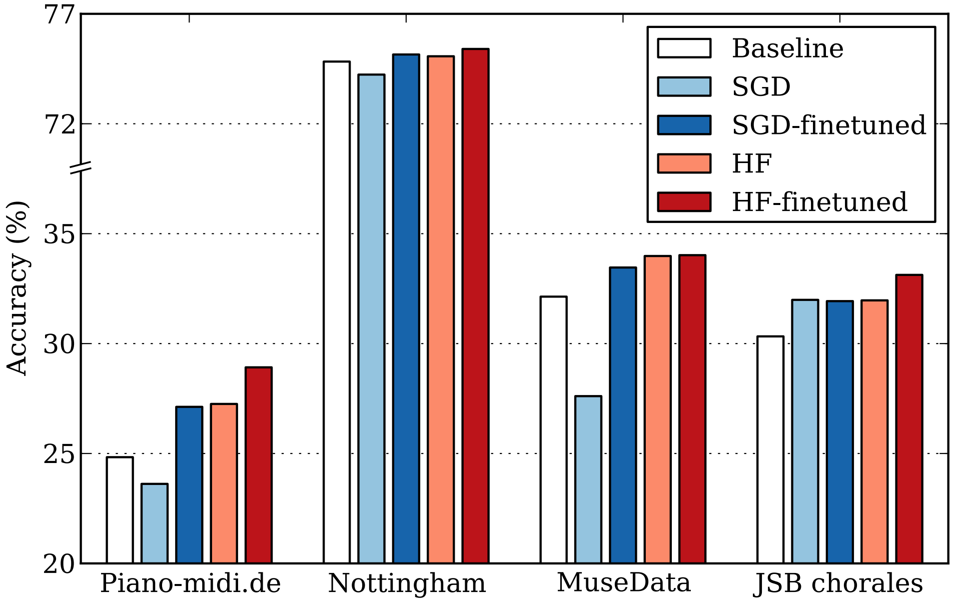


Figure 4.4: Effect of SGD and HF pretraining on the RNN-RBM symbolic prediction performance. All strategies except the baseline involve pretraining.

4.7 Polyphonic transcription

Multiple fundamental frequency (f_0) estimation, or polyphonic transcription, consists in estimating the audible note pitches in the signal at 10 ms intervals without tracking note contours. We combine our polyphonic sequence models with the acoustic model of Nam et al. (2011) in order to demonstrate a practical application of the sequence models. Their model was adapted for multiple instruments, and it can be generalized to any method that can score hypothetical combinations of f_0 for a given time frame.

At each time frame, the Nam et al. (2011) algorithm outputs independent probabilities that each note is present and reports every note with probability $p \geq 0.5$. To incorporate our symbolic model prediction $P_s(v^{(t)}|\mathcal{A}^{(t)})$, we consider the k most promising f_0 candidates ($k = 7$) from the acoustic model $P_a(v^{(t)})$ and jointly evaluate all combinations of M candidates $\forall M \leq k$ by the following cost function:

$$C = -\log P_a(v^{(t)}) - \alpha \log P_s(v^{(t)}|\tilde{\mathcal{A}}^{(t)}) \tag{4.23}$$

where $\tilde{\mathcal{A}}^{(t)}$ is the approximate sequence history constructed from the f_0 estimated so far in at least half the audio frames corresponding to each past symbolic time step[6]. This corresponds to a product of experts where the hyperparameter α is the confidence coefficient of our symbolic predictor. If our algorithm is run on audio signals without preprocessing, tempo tracking must be performed first. Since the symbolic models describe only fixed tonality pieces, a first audio-only pass is needed to transpose the estimated f_0 in the correct tonality. Once the optimal f_0 estimates have been determined, HMM smoothing can still filter out spurious results and enhance onset accuracies.

Digital audio has been generated for the four datasets and we report in Figure 4.5 the frame-level transcription accuracy of the Nam et al. (2011) algorithm, either alone, after HMM smoothing, or using our best performing model as a symbolic prior. We observe an improvement in absolute accuracy between 1.3% and 10% over the HMM approach. It can be seen easily that an HMM with emission probabilities $P_a(v^{(t)})$ is equivalent to equation (4.23) with a note 2-gram symbolic model, one time step per audio frame and $\alpha = 1$. It is therefore unsurprising that the advantage of our search algorithm decreases when the note N-gram already performs well, e.g. for Piano-midi.de (Table 4.1). However, the HMM allows for a *global* search of the most likely f_0 (the Viterbi path), whereas our algorithm requires a greedy chronological search, a limitation we are currently working to address.

4.8 Conclusions

We presented an RNN-based model that can learn harmonic and rhythmic probabilistic rules from polyphonic music scores of varying complexity, substantially better than popular methods in music information retrieval. We showed that different strategies related to the description of temporal dependencies can improve prediction accuracy of such models. While longer-term musical structure remains elusive in our unconstrained representation, our model can immediately serve as a

[6] This can create a 'snowball' effect where accurate baseline transcriptions form accurate $\tilde{\mathcal{A}}^{(t)}$ estimates, resulting in more relevant symbolic predictions $P_s(v^{(t)}|\tilde{\mathcal{A}}^{(t)})$, which in turn improve the final transcription.

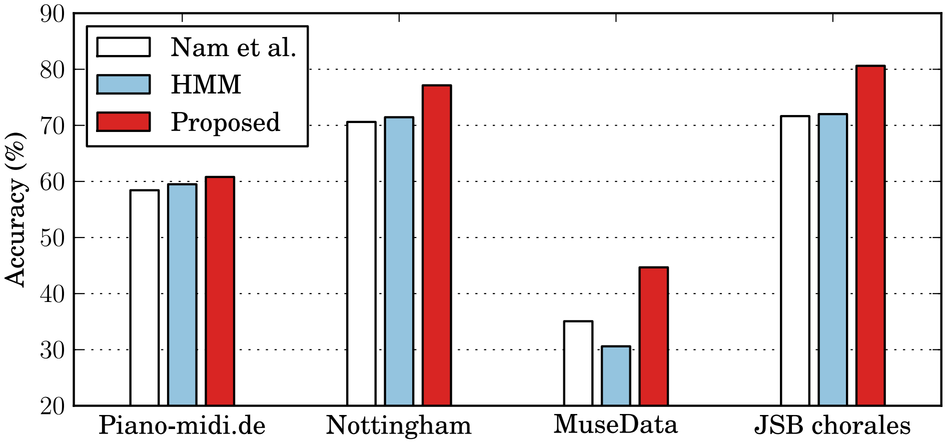


Figure 4.5: Frame-level transcription accuracy of the Nam et al. (2011) model either alone, after HMM smoothing or with our best performing model as a symbolic prior.

symbolic prior for polyphonic transcription, clearly improving the state of the art in this area.

Acknowledgments

The authors would like to thank NSERC, CIFAR and the Canada Research Chairs for funding, and Compute Canada/Calcul Québec for computing resources.

5 Prologue to Second Article

5.1 Article Details

Advances in Optimizing Recurrent Networks
Yoshua Bengio, Nicolas Boulanger-Lewandowski and Razvan Pascanu
Published in *Proceedings of the 38th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* in 2013.

5.2 Context

After it was observed that capturing long-term dependencies by gradient-based optimization was difficult (Hochreiter, 1991; Bengio et al., 1994), there has been a major reduction in research efforts in the area of RNNs in the 90's and 2000's. Several strategies have been proposed to reduce those difficulties, such as long short term memory (LSTM) networks (Hochreiter and Schmidhuber, 1997), Hessian-free optimization (Martens and Sutskever, 2011), clipped gradients (Mikolov, 2012; Pascanu et al., 2012, 2013), leaky units (El Hihi and Bengio, 1996; Jaeger et al., 2007; Siewert and Wustlich, 2007), conditional output distribution estimators (Chapter 4), sparser gradients (Bengio, 2009), and Nesterov momentum (Nesterov, 1983; Sutskever, 2012). There is now a revival of interest in these learning algorithms and their use in state-of-the-art systems (Mikolov et al., 2011; Sutskever, 2012).

5.3 Contributions

This paper studies the issues giving rise to the optimization difficulties in RNNs and discusses, reviews, and combines several techniques that have been proposed in order to improve training. Experiments are carried over datasets of symbolic polyphonic music and text at both character and word level. We find that these techniques generally help generalization performance as well as training performance, which suggests they help to improve the optimization of the training criterion. We also find that although these techniques can be applied in the online setting similarly to stochastic gradient descent (SGD), they allow to compete with second-order methods such as Hessian-Free optimization. Finally, we propose a simplified formulation of Nesterov momentum from the point of view of regular momentum and we offer an alternative interpretation of the method.

All three co-authors provided a similar contribution to this paper. I conducted experiments with the RNN-RBM and RNN-NADE and discussed the results obtained on polyphonic music data. I also developed the theory of Section 6.3.5.

6 Advances in Optimizing Recurrent Networks

After more than a decade-long period of relatively little research activity in the area of recurrent neural networks, several new developments will be reviewed here that have allowed substantial progress both in understanding and in technical solutions towards more efficient training of recurrent networks. These advances have been motivated by and related to the optimization issues surrounding deep learning. Although recurrent networks are extremely powerful in what they can in principle represent in terms of modeling sequences, their training is plagued by two aspects of the same issue regarding the learning of long-term dependencies. Experiments reported here evaluate the use of clipping gradients, spanning longer time ranges with leaky integration, advanced momentum techniques, using more powerful output probability models, and encouraging sparser gradients to help symmetry breaking and credit assignment. The experiments are performed on text and music data and show off the combined effects of these techniques in generally improving both training and test error.

6.1 Introduction

Machine learning algorithms for capturing statistical structure in sequential data face a fundamental problem (Hochreiter, 1991; Bengio et al., 1994), called the *difficulty of learning long-term dependencies*. If the operations performed when forming a fixed-size summary of relevant past observations (for the purpose of predicting some future observations) are linear, this summary must exponentially forget past events that are further away, to maintain stability. On the other hand, if they are non-linear, then this non-linearity is composed many times, yielding a highly non-linear relationship between past events and future events. Learning such non-linear relationships turns out to be difficult, for reasons that are discussed here, along with recent proposals for reducing this difficulty.

Recurrent neural networks (Rumelhart et al., 1986b) can represent such non-linear maps (F, below) that iteratively build a relevant summary of past observations. In their simplest form, recurrent neural networks (RNNs) form a deterministic *state* variable h_t as a function of the present input observation x_t and the past value(s) of the state variable, e.g., $h_t = F_\theta(h_{t-1}, x_t)$, where θ are tunable parameters that control what will be remembered about the past sequence and what will be discarded. Depending on the type of problem at hand, a loss function $L(h_t, y_t)$ is defined, with y_t an observed random variable at time t and $C_t = L(h_t, y_t)$ the cost at time t. The generalization objective is to minimize the expected future cost, and the training objective involves the average of C_t over observed sequences. In principle, RNNs can be trained by gradient-based optimization procedures (using the back-propagation algorithm (Rumelhart et al., 1986b) to compute a gradient), but it was observed early on (Hochreiter, 1991; Bengio et al., 1994) that capturing dependencies that span a long interval was difficult, making the task of optimizing θ to minimize the average of C_t's almost impossible for some tasks when the span of the dependencies of interest increases sufficiently. More precisely, using a local numerical optimization such as stochastic gradient descent or second order methods (which gradually improve the solution), the proportion of trials (differing only from their random initialization) falling into the basin of attraction of a good enough solution quickly becomes very small as the temporal span of dependencies is increased (beyond tens or hundreds of steps, depending of the task).

These difficulties are probably responsible for the major reduction in research efforts in the area of RNNs in the 90's and 2000's. However, a revival of interest in these learning algorithms is taking place, in particular thanks to (Martens and Sutskever, 2011) and (Mikolov et al., 2011). This paper studies the issues giving rise to these difficulties and discusses, reviews, and combines several techniques that have been proposed in order to improve training of RNNs, following up on a recent thesis devoted to the subject (Sutskever, 2012). We find that these techniques generally help generalization performance as well as training performance, which suggest they help to improve the optimization of the training criterion. We also find that although these techniques can be applied in the online setting, i.e., as add-ons to stochastic gradient descent (SGD), they allow to compete with batch (or large minibatch) second-order methods such as Hessian-Free optimization, recently found to greatly help training of RNNs (Martens and Sutskever, 2011).

6.2 Learning Long-Term Dependencies and the Optimization Difficulty with Deep Learning

There has been several breakthroughs in recent years in the algorithms and results obtained with so-called *deep learning* algorithms (see (Bengio, 2009) and (Bengio et al., 2012) for reviews). Deep learning algorithms discover multiple levels of representation, typically as deep neural networks or graphical models organized with many levels of representation-carrying latent variables. Very little work on deep architectures occurred before the major advances of 2006 (Hinton et al., 2006; Bengio et al., 2006; Ranzato et al., 2007), probably because of *optimization difficulties* due to the high level of non-linearity in deeper networks (whose output is the composition of the non-linearity at each layer). Some experiments (Erhan et al., 2010) showed the presence of an extremely large number of apparent local minima of the training criterion, with no two different initializations going to the same *function* (i.e. eliminating the effect of permutations and other symmetries of parametrization giving rise to the same function). Furthermore, qualitatively different initialization (e.g., using unsupervised learning) could yield models in completely different regions of function space. An unresolved question is whether these difficulties are actually due to local minima or to ill-conditioning (which makes gradient descent converge so slowly as to appear stuck in a local minimum). Some ill-conditioning has clearly been shown to be involved, especially for the difficult problem of training deep auto-encoders, through comparisons (Martens, 2010) of stochastic gradient descent and Hessian-free optimization (a second order optimization method). These optimization questions become particularly important when trying to train very large networks on very large datasets (Le et al., 2012), where one realizes that a major challenge for deep learning is the *underfitting* issue. Of course one can trivially overfit by increasing capacity in the wrong places (e.g. in the output layer), but what we are trying to achieve is learning of more powerful representations in order to also get good generalization.

6.2 Learning Long-Term Dependencies and the Optimization Difficulty with Deep Learning

The same questions can be asked for RNNs. When the computations performed by a RNN are unfolded through time, one clearly sees a deep neural network *with shared weights* (across the 'layers', each corresponding to a different time step), and with a cost function that may depends on the output of intermediate layers. Hessian-free optimization has been successfully used to considerably extend the span of temporal dependencies that a RNN can learn (Martens and Sutskever, 2011), suggesting that ill-conditioning effects are also at play in the difficulties of training RNN.

An important aspect of these difficulties is that the gradient can be decomposed (Bengio et al., 1994; Pascanu et al., 2012) into terms that involve *products of Jacobians* $\frac{\partial h_t}{\partial h_{t-1}}$ over subsequences linking an event at time t_1 and one at time t_2: $\frac{\partial h_{t_2}}{\partial h_{t_1}} = \prod_{\tau=t_1+1}^{t_2} \frac{\partial h_\tau}{\partial h_{\tau-1}}$. As $t_2 - t_1$ increases, the products of $t_2 - t_1$ of these Jacobian matrices tend to either vanish (when the leading eigenvalues of $\frac{\partial h_t}{\partial h_{t-1}}$ are less than 1) or explode (when the leading eigenvalues of $\frac{\partial h_t}{\partial h_{t-1}}$ are greater than 1 [1]). This is problematic because the total gradient due to a loss C_{t_2} at time t_2 is a sum whose terms correspond to the effects at different time spans, which are weighted by $\frac{\partial h_{t_2}}{\partial h_{t_1}}$ for different t_1's:

$$\frac{\partial C_{t_2}}{\partial \theta} = \sum_{t_1 \leq t_2} \frac{\partial C_{t_2}}{\partial h_{t_2}} \frac{\partial h_{t_2}}{\partial h_{t_1}} \frac{\partial h_{t_1}}{\partial \theta^{(t_1)}}$$

where $\frac{\partial h_{t_1}}{\partial \theta^{(t_1)}}$ is the derivative of h_{t_1} with respect to the instantiation of the parameters θ at step t_1, i.e., that directly come into the computation of h_{t_1} in F. When the $\frac{\partial h_{t_2}}{\partial h_{t_1}}$ tend to vanish for increasing $t_2 - t_1$, the long-term term effects become exponentially smaller in magnitude than the shorter-term ones, making it very difficult to capture them. On the other hand, when $\frac{\partial h_{t_2}}{\partial h_{t_1}}$ "explode" (becomes large), gradient descent updates can be destructive (move to poor configuration of parameters). It is not that the gradient is wrong, it is that gradient descent makes small but finite steps $\Delta\theta$ yielding a ΔC, whereas the gradient measures the effect of ΔC when $\Delta\theta \to 0$. A much deeper discussion of this issue can be found in (Pascanu et al., 2012), along with a point of view inspired by dynamical systems theory and by the geometrical aspect of the problem, having to do with the shape of the training criterion as a function of θ near those regions of exploding gradient. In particular, it

[1] Note that this is not a sufficient condition, but a necessary one. Further more one usually wants to operate in the regime where the leading eigenvalue is larger than 1 but the gradients do not explode.

is argued that the strong non-linearity occurring where gradients explode is shaped like a cliff where not just the first but also the second derivative becomes large *in the direction orthogonal to the cliff*. Similarly, flatness of the cost function occurs simultaneously on the first and second derivatives. Hence dividing the gradient by the second derivative in each direction (i.e., pre-multiplying by the inverse of some proxy for the Hessian matrix) could in principle reduce the exploding and vanishing gradient effects, as argued in (Martens and Sutskever, 2011).

6.3 Advances in Training Recurrent Networks

6.3.1 Clipped Gradient

To address the exploding gradient effect, (Mikolov, 2012; Pascanu et al., 2012) recently proposed to *clip gradients above a given threshold*. Under the hypothesis that the explosion occurs in very small regions (the cliffs in cost function mentioned above), most of the time this will have no effect, but it will avoid aberrant parameter changes in those cliff regions, while guaranteeing that the resulting updates are still in a descent direction. The specific form of clipping used here was proposed in (Pascanu et al., 2012) and is discussed there at much greater length: when the norm of the gradient vector g for a given sequence is above a $threshold$, the update is done in the direction $threshold\frac{g}{||g||}$. As argued in (Pascanu et al., 2012), this very simple method implements a very simple form of second order optimization in the sense that the second derivative is also proportionally large in those exploding gradient regions.

6.3.2 Spanning Longer Time Ranges with Leaky Integration

An old idea to reduce the effect of vanishing gradients is to introduce shorter paths between t_1 and t_2, either via connections with longer time delays (Lin et al., 1995) or inertia (slow-changing units) in some of the hidden units (El Hihi and Bengio, 1996; Jaeger et al., 2007), or both (Sutskever and Hinton, 2010). Long-Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997), which were shown to be able to handle much longer range dependencies, also benefit from a linearly self-connected memory unit with a near 1 self-weight which allows signals (and gradients) to propagate over long time spans.

A different interpretation to this slow-changing units is that they behave like low-pass filter and hence they can be used to focus certain units on different frequency regions of the data. The analogy can be brought one step further by introducing band-pass filter units (Siewert and Wustlich, 2007) or by using domain specific knowledge to decide on what frequency bands different units should focus. (Mikolov and Zweig, 2012) shows that adding low frequency information as an additional input to a recurrent network helps improving the performance of the model.

In the experiments performed here, a subset of the units were forced to change slowly by using the following "leaky integration" state-to-state map: $h_{t,i} = \alpha_i h_{t-1,i} + (1 - \alpha_i) F_i(h_{t-1}, x_t)$. The standard RNN corresponds to $\alpha_i = 0$, while here different values of α_i were randomly sampled from $(0.02, 0.2)$, allowing some units to react quickly while others are forced to change slowly, but also propagate signals and gradients further in time. Note that because $\alpha < 1$, the vanishing effect is still present (and gradients can still explode via F), but the *time-scale* of the vanishing effect can be expanded.

6.3.3 Combining Recurrent Nets with a Powerful Output Probability Model

One way to reduce the underfitting of RNNs is to introduce multiplicative interactions in the parametrization of F, as was done successfully in (Martens and Sutskever, 2011). When the output predictions are multivariate, another approach is to capture the high-order dependencies between the output variables using a powerful output probability model such as a Restricted Boltzmann Machine (RBM) (Sutskever et al., 2008; Boulanger-Lewandowski et al., 2012b) or a deterministic variant of it called NADE (Larochelle and Murray, 2011; Boulanger-Lewandowski et al., 2012b). In the experiments performed here, we have experimented with a NADE output model for the music data.

6.3.4 Sparser Gradients via Sparse Output Regularization and Rectified Outputs

(Bengio, 2009) hypothesized that one reason for the difficulty in optimizing deep networks is that in ordinary neural networks gradients diffuse through the layers, diffusing credit and blame through many units, maybe making it difficult for hidden units to specialize. When the gradient on hidden units is more sparse, one could imagine that symmetries would be broken more easily and credit or blame assigned less uniformly. This is what was advocated in (Glorot et al., 2011a), exploiting the idea of rectifier non-linearities introduced earlier in (Nair and Hinton, 2010), i.e., the neuron non-linearity is $out = \max(0, in)$ instead of $out = \tanh(in)$ or $out = \text{sigmoid}(in)$. This approach was very successful in recent work on deep learning for object recognition (Krizhevsky et al., 2012), beating by far the state-of-the-art on ImageNet (1000 classes). Here, we apply this deep learning idea to RNNs, using an L1 penalty on outputs of hidden units to promote sparsity of activations. The underlying hypothesis is that if the gradient is concentrated in a few paths (in the unfolded computation graph of the RNN), it will reduce the vanishing gradients effect.

6.3.5 Simplified Nesterov Momentum

Nesterov accelerated gradient (NAG) (Nesterov, 1983) is a first-order optimization method to improve stability and convergence of regular gradient descent. Recently, (Sutskever, 2012) showed that NAG could be computed by the following update rules:

$$v_t = \mu_{t-1} v_{t-1} - \epsilon_{t-1} \nabla f(\theta_{t-1} + \mu_{t-1} v_{t-1}) \tag{6.1}$$

$$\theta_t = \theta_{t-1} + v_t \tag{6.2}$$

where θ_t are the model parameters, v_t the velocity, $\mu_t \in [0, 1]$ the momentum (decay) coefficient and $\epsilon_t > 0$ the learning rate at iteration t, $f(\theta)$ is the objective function and $\nabla f(\theta')$ is a shorthand notation for the gradient $\frac{\partial f(\theta)}{\partial \theta}|_{\theta=\theta'}$. These equations have a form similar to standard momentum updates:

$$v_t = \mu_{t-1} v_{t-1} - \epsilon_{t-1} \nabla f(\theta_{t-1}) \tag{6.3}$$

$$\theta_t = \theta_{t-1} + v_t \tag{6.4}$$

$$= \theta_{t-1} + \mu_{t-1} v_{t-1} - \epsilon_{t-1} \nabla f(\theta_{t-1}) \tag{6.5}$$

and differ only in the evaluation point of the gradient at each iteration. This important difference, thought to counterbalance too high velocities by “peeking ahead” actual objective values in the candidate search direction, results in significantly improved RNN performance on a number of tasks.

In this section, we derive a new formulation of Nesterov momentum differing from (6.3) and (6.5) only in the linear combination coefficients of the velocity and gradient contributions at each iteration, and we offer an alternative interpretation of the method. The key departure from (6.1) and (6.2) resides in committing to the “peeked-ahead” parameters $\Theta_{t-1} \equiv \theta_{t-1} + \mu_{t-1} v_{t-1}$ and backtracking by the same amount before each update. Our new parameters Θ_t updates become:

$$v_t = \mu_{t-1} v_{t-1} - \epsilon_{t-1} \nabla f(\Theta_{t-1}) \tag{6.6}$$

$$\begin{aligned} \Theta_t &= \Theta_{t-1} - \mu_{t-1} v_{t-1} + \mu_t v_t + v_t \\ &= \Theta_{t-1} + \mu_t \mu_{t-1} v_{t-1} - (1 + \mu_t) \epsilon_{t-1} \nabla f(\Theta_{t-1}) \end{aligned} \tag{6.7}$$

Assuming a zero initial velocity $v_1 = 0$ and velocity at convergence of optimization $v_T \simeq 0$, the parameters Θ are a completely equivalent replacement of θ.

Note that equation (6.7) is identical to *regular* momentum (6.5) with different linear combination coefficients. More precisely, for an equivalent velocity update (6.6), the velocity contribution to the new parameters $\mu_t \mu_{t-1} < \mu_t$ is reduced relatively to the gradient contribution $(1 + \mu_t)\epsilon_{t-1} > \epsilon_{t-1}$. This allows storing past velocities for a longer time with a higher μ, while actually using those velocities more conservatively during the updates. We suspect this mechanism is a crucial ingredient for good empirical performance. While the “peeking ahead” point of view suggests that a similar strategy could be adapted for regular gradient descent (misleadingly, because it would amount to a reduced learning rate ϵ_t), our derivation shows why it is important to choose search directions aligned with the current velocity to yield substantial improvement. The general case is also simpler to implement.

6.4 Experiments

In the experimental section we compare vanilla SGD versus SGD plus some of the enhancements discussed above. Specifically we use the letter ‘C‘ to indicate that gradient clipping is used, ‘L‘ for leaky-integration units, ‘R‘ if we use rectifier units with L1 penalty and ‘M‘ for Nesterov momentum.

6.4.1 Music Data

We evaluate our models on the four polyphonic music datasets of varying complexity used in (Boulanger-Lewandowski et al., 2012b): classical piano music (Piano-midi.de), folk tunes with chords instantiated from ABC notation (Nottingham), orchestral music (MuseData) and the four-part chorales by J.S. Bach (JSB chorales). The symbolic sequences contain high-level pitch and timing information in the form of a binary matrix, or *piano-roll*, specifying precisely which notes occur at each time-step. They form interesting benchmarks for RNNs because of their high dimensionality and the complex temporal dependencies involved at different time scales. Each dataset contains at least 7 hours of polyphonic music with an average polyphony (number of simultaneous notes) of 3.9.

Piano-rolls were prepared by aligning each time-step (88 pitch labels that cover the whole range of piano) on an integer fraction of the beat (quarter note) and transposing each sequence in a common tonality (C major/minor) to facilitate learning. Source files and preprocessed piano-rolls split in train, validation and test sets are available on the authors' website[2].

Setup and Results

We select hyperparameters, such as the number of hidden units n_h, regularization coefficients λ_{L1}, the choice of non-linearity function, or the momentum schedule μ_t, learning rate ϵ_t, number of leaky units n_{leaky} or leaky factors α according to

[2] `www-etud.iro.umontreal.ca/~boulanni/icml2012`

log-likelihood on a validation set and we report the final performance on the test set for the best choice in each category. We do so by using random search (Bergstra and Bengio, 2012) on the following intervals:

$$\begin{array}{ll} n_h \in [100, 400] & \epsilon_t \in [10^{-4}, 10^{-1}] \\ \mu_t \in [10^{-3}, 0.95] & \lambda_{L1} \in [10^{-6}, 10^{-3}] \\ n_{leaky} \in \{0\%, 25\%, 50\%\} & \alpha \in [0.02, 2] \end{array}$$

The cutoff threshold for gradient clipping is set based on the average norm of the gradient over one pass on the data, and we used 15 in this case for all music datasets. The data is split into sequences of 100 steps over which we compute the gradient. The hidden state is carried over from one sequence to another if they belong to the same song, otherwise is set to 0.

Table 6.1 presents log-likelihood (LL) and expected frame-level accuracy for various RNNs in the symbolic music prediction task.

Results clearly show that these enhancements allow to improve on regular SGD in almost all cases; they also make SGD competitive with HF for the sigmoid recognition layers RNNs.

6.4.2 Text Data

We use the Penn Treebank Corpus to explore both word and character prediction tasks. The data is split by using sections 0-20 as training data (5017k characters), sections 21-22 as validation (393k characters) and sections 23-24 as test data (442k characters).

For the word level prediction, we fix the dictionary to 10000 words, which we divide into 30 classes according to their frequency in text (each class holding approximately 3.3% of the total number of tokens in the training set). Such a factorization allows for faster implementation, as we are not required to evaluate the whole output layer (10000 units) which is the computational bottleneck, but only the output of the corresponding class (Mikolov et al., 2011).

Setup and Results

In the case of next word prediction, we compute gradients over sequences of 40 steps, where we carry the hidden state from one sequence to another. We use a small grid-search around the parameters used to get state of the art results for this number of classes (Mikolov et al., 2011), i.e., with a network of 200 hidden units yielding a perplexity of 134. We explore learning rate of $0.1, 0.01, 0.001$, rectifier units versus sigmoid units, cutoff threshold for the gradients of 30, 50 or none, and no leaky units versus 50 of the units being sampled from 0.2 and 0.02.

For the character level model we compute gradients over sequences of 150 steps, as we assume that longer dependencies are more crucial in this case. We use 500 hidden units and explore learning rates of 0.5, 0.1 and 0.01.

In table 6.2 we have entropy (bits per character) or perplexity for various RNNs on the word and character prediction tasks. Again, we observe substantial improvements in both training and test perplexity, suggesting that these techniques make optimization easier.

6.5 Conclusions

Through our experiments we provide evidence that part of the issue of training RNN is due to the rough error surface which can not be easily handled by SGD. We follow an incremental set of improvements to SGD, and show that in most cases they improve both the training and test error, and allow this enhanced SGD to compete or even improve on a second-order method which was found to work particularly well for RNNs, i.e., Hessian-Free optimization.

Table 6.1: Log-likelihood and expected accuracy for various RNN models in the symbolic music prediction task. The double line separates sigmoid recognition layers (above) to structured output probability models (below).

Model	Piano-midi.de			Nottingham			MuseData			JSB chorales		
	LL (train)	LL (test)	ACC % (test)	LL (train)	LL (test)	ACC % (test)	LL (train)	LL (test)	ACC % (test)	LL (train)	LL (test)	ACC % (test)
RNN (SGD)	-7.10	-7.86	22.84	-3.49	-3.75	66.90	-6.93	-7.20	27.97	-7.88	-8.65	**29.97**
RNN (SGD+C)	-7.15	**-7.59**	22.98	-3.40	-3.67	67.47	-6.79	-7.04	30.53	-7.81	-8.65	**29.98**
RNN (SGD+CL)	-7.04	**-7.57**	22.97	-3.31	-3.57	67.97	**-6.47**	**-6.99**	**31.53**	-7.78	-8.63	**29.98**
RNN (SGD+CLR)	**-6.40**	-7.80	**24.22**	**-2.99**	-3.55	**70.20**	-6.70	-7.34	29.06	**-7.67**	-9.47	**29.98**
RNN (SGD+CRM)	-6.92	-7.73	23.71	-3.20	**-3.43**	68.47	-7.01	-7.24	29.13	-8.08	-8.81	29.52
RNN (HF)	-7.00	**-7.58**	22.93	-3.47	-3.76	66.71	-6.76	-7.12	29.77	-8.11	**-8.58**	29.41
RNN-RBM	N/A	-7.09	**28.92**	N/A	-2.39	**75.40**	N/A	-6.01	**34.02**	N/A	-6.27	33.12
RNN-NADE (SGD)	-7.23	-7.48	20.69	-2.85	-2.91	64.95	-6.86	-6.74	24.91	-5.46	-5.83	32.11
RNN-NADE (SGD+CR)	-6.70	-7.34	21.22	-2.14	-2.51	69.80	-6.27	-6.37	26.60	-4.44	-5.33	34.52
RNN-NADE (SGD+CRM)	-6.61	-7.34	22.12	-2.11	-2.49	69.54	-5.99	-6.19	29.62	**-4.26**	**-5.19**	**35.08**
RNN-NADE (HF)	**-6.32**	**-7.05**	23.42	**-1.81**	**-2.31**	71.50	**-5.20**	**-5.60**	32.60	-4.91	-5.56	32.50

Table 6.2: Entropy (bits per character) and perplexity for various RNN models on next character and next word prediction task.

Model	Penn Treebank Corpus word level		Penn Treebank Corpus character level	
	perplexity (train)	perplexity (test)	entropy (train)	entropy (test)
RNN (SGD)	112.11	145.16	1.78	1.76
RNN (SGD+C)	78.71	136.63	**1.40**	**1.44**
RNN (SGD+CL)	76.70	129.83	1.56	1.56
RNN (SGD+CLR)	**75.45**	**128.35**	1.45	1.49

7 Prologue to Third Article

7.1 Article Details

High-dimensional Sequence Transduction
Nicolas Boulanger-Lewandowski, Yoshua Bengio and Pascal Vincent
Published in *Proceedings of the 38th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* in 2013.

7.2 Context

The polyphonic transcription method based on the RNN-RBM presented in Chapter 4 combines the acoustic and symbolic models by a product of experts and a greedy chronological search. Existing jointly trained models either operate in the time domain under Markovian assumptions (Cemgil et al., 2006) or neglect intra-frame note correlations and temporal smoothing connections (Böck and Schedl, 2012). A generic sequence transduction framework using RNNs to transform an input sequence into an output sequence was also introduced by Graves (2012), but that approach is designed for a small number of discrete output symbols and is not directly applicable to high-dimensional outputs.

7.3 Contributions

In this chapter, we introduce an input/output extension of the RNN-RBM that can learn the conditional output distribution given the input, whereas the original RNN-RBM only learns the output sequence distribution. We also devise an efficient algorithm to search for the global mode of that distribution. We conduct experiments on five datasets of synthesized sounds and real recordings that show a significant improvement in transcription accuracy over other methods. We demonstrate that our approach can produce musically plausible transcriptions in high levels of stationary and non-stationary noise.

7.4 Recent Developments

The model and inference algorithm introduced in this chapter form the basis of the chord recognition and speech recognition systems developed in Chapters 10 and 12 respectively. In this chapter, we partially control the label bias problem with output noise, L_2 regularization to the input and output weights, and longer time steps; it nevertheless remains an important issue. An alternative generative architecture will be proposed in Chapter 12 to circumvent this problem.

8 High-dimensional sequence transduction

We investigate the problem of transforming an input sequence into a high-dimensional output sequence in order to transcribe polyphonic audio music into symbolic notation. We introduce a probabilistic model based on a recurrent neural network that is able to learn realistic output distributions given the input and we devise an efficient algorithm to search for the global mode of that distribution. The resulting method produces musically plausible transcriptions even under high levels of noise and drastically outperforms previous state-of-the-art approaches on five datasets of synthesized sounds and real recordings, approximately *halving the test error rate*.

8.1 Introduction

Machine learning tasks can often be formulated as the transformation, or *transduction*, of an input sequence into an output sequence: speech recognition, machine translation, chord recognition or automatic music transcription, for example. Recurrent neural networks (RNN) (Rumelhart et al., 1986a) offer an interesting route for sequence transduction (Graves, 2012) because of their ability to represent arbitrary output distributions involving complex temporal dependencies at different time scales.

When the output predictions are high-dimensional vectors, such as tuples of notes in musical scores, it becomes very expensive to enumerate all possible configurations at each time step. One possible approach is to capture high-order interactions between output variables using restricted Boltzmann machines (RBM) (Smolensky, 1986) or a tractable variant called NADE (Larochelle and Murray, 2011), a weight-sharing form of the architecture introduced in (Bengio and Bengio, 2000).

8.1 Introduction

In a recently developed probabilistic model called the RNN-RBM, a series of distribution estimators (one at each time step) are conditioned on the deterministic output of an RNN (Boulanger-Lewandowski et al., 2012b; Sutskever et al., 2008). In this work, we introduce an input/output extension of the RNN-RBM that can learn to map input sequences to output sequences, whereas the original RNN-RBM only learns the output sequence distribution. In contrast to the approach of (Graves, 2012) designed for discrete output symbols, or one-hot vectors, our high-dimensional paradigm requires a more elaborate inference procedure. Other differences include our use of second-order Hessian-free (HF) (Martens and Sutskever, 2011) optimization[1] but not of LSTM cells (Hochreiter and Schmidhuber, 1997) and, for simplicity and performance reasons, our use of a single recurrent network to perform both transcription and temporal smoothing. We also do not need special "null" symbols since the sequences are already aligned in our main task of interest: polyphonic music transcription.

The objective of polyphonic transcription is to obtain the underlying notes of a polyphonic audio signal as a symbolic *piano-roll*, i.e. as a binary matrix specifying precisely which notes occur at each time step. We will show that our transduction algorithm produces more musically plausible transcriptions in both noisy and normal conditions and achieve superior overall accuracy (Bay et al., 2009) compared to existing methods. Our approach is also an improvement over the hybrid method in (Boulanger-Lewandowski et al., 2012b) that combines symbolic and acoustic models by a product of experts and a greedy chronological search, and (Cemgil et al., 2006) that operates in the time domain under Markovian assumptions. Finally, (Böck and Schedl, 2012) employs a bidirectional RNN without temporal smoothing and with independent output note probabilities. Other tasks that can be addressed by our transduction framework include automatic accompaniment, melody harmonization and audio music denoising.

[1] Our code is available online at `http://www-etud.iro.umontreal.ca/~boulanni/icassp2013`.

8.2 Proposed architecture

8.2.1 Restricted Boltzmann machines

An RBM is an energy-based model where the joint probability of a given configuration of the visible vector $v \in \{0,1\}^N$ (output) and the hidden vector h is:

$$P(v,h) = \exp(-b_v^{\mathrm{T}} v - b_h^{\mathrm{T}} h - h^{\mathrm{T}} W v)/Z \tag{8.1}$$

where b_v, b_h and W are the model parameters and Z is the usually intractable partition function. The marginalized probability of v is related to the free-energy $F(v)$ by $P(v) \equiv e^{-F(v)}/Z$:

$$F(v) = -b_v^{\mathrm{T}} v - \sum_i \log(1 + e^{b_h + Wv})_i \tag{8.2}$$

The gradient of the negative log-likelihood of an observed vector v involves two opposing terms, called the positive and negative phase:

$$\frac{\partial(-\log P(v))}{\partial\Theta} = \frac{\partial F(v)}{\partial\Theta} - \frac{\partial(-\log Z)}{\partial\Theta} \tag{8.3}$$

where $\Theta \equiv \{b_v, b_h, W\}$. The second term can be estimated by a single sample v^* obtained from a Gibbs chain starting at v:

$$\frac{\partial(-\log P(v))}{\partial\Theta} \simeq \frac{\partial F(v)}{\partial\Theta} - \frac{\partial F(v^*)}{\partial\Theta}. \tag{8.4}$$

resulting in the well-known contrastive divergence algorithm (Hinton, 2002).

8.2.2 NADE

The neural autoregressive distribution estimator (NADE) (Larochelle and Murray, 2011) is a tractable model inspired by the RBM. NADE is similar to a fully visible sigmoid belief network in that the conditional probability distribution of a visible unit v_j is expressed as a nonlinear function of the vector $v_{<j} \equiv \{v_k, \forall k < j\}$:

$$P(v_j = 1 | v_{<j}) = \sigma(W_{:,j}^{\top} h_j + (b_v)_j) \tag{8.5}$$

$$h_j = \sigma(W_{:,<j} v_{<j} + b_h) \tag{8.6}$$

where $\sigma(x) \equiv (1 + e^{-x})^{-1}$ is the logistic sigmoid function.

In the following discussion, one can substitute RBMs with NADEs by replacing equation (8.4) with the exact gradient of the negative log-likelihood cost $C \equiv -\log P(v)$:

$$\frac{\partial C}{\partial (b_v)_j} = P(v_j = 1 | v_{<j}) - v_j \tag{8.7}$$

$$\frac{\partial C}{\partial b_h} = \sum_{k=1}^{N} \frac{\partial C}{\partial (b_v)_k} W_{:,k} h_k (1 - h_k) \tag{8.8}$$

$$\frac{\partial C}{\partial W_{:,j}} = \frac{\partial C}{\partial (b_v)_j} h_j + v_j \sum_{k=j+1}^{N} \frac{\partial C}{\partial (b_v)_k} W_{:,k} h_k (1 - h_k) \tag{8.9}$$

In addition to the possibility of using HF for training, a tractable distribution estimator is necessary to compare the probabilities of different output sequences during inference.

8.2.3 The input/output RNN-RBM

The I/O RNN-RBM is a sequence of conditional RBMs (one at each time step) whose parameters $b_v^{(t)}, b_h^{(t)}, W^{(t)}$ are time-dependent and depend on the sequence history at time t, denoted $\mathcal{A}^{(t)} \equiv \{x^{(\tau)}, v^{(\tau)} | \tau < t\}$ where $\{x^{(t)}\}, \{v^{(t)}\}$ are respectively the input and output sequences. Its graphical structure is depicted in Figure 8.1. Note that by ignoring the input x, this model would reduce to the RNN-RBM (Boulanger-Lewandowski et al., 2012b). The I/O RNN-RBM is formally defined by its joint probability distribution:

$$P(\{v^{(t)}\}) = \prod_{t=1}^{T} P(v^{(t)} | \mathcal{A}^{(t)}) \tag{8.10}$$

where the right-hand side multiplicand is the marginalized probability of the t^{th} RBM (eq. 8.2) or NADE (eq. 8.5).

Following our previous work, we will consider the case where only the biases are variable:

$$b_h^{(t)} = b_h + W_{\hat{h}h} \hat{h}^{(t-1)} + W_{xh} x^{(t)} \tag{8.11}$$

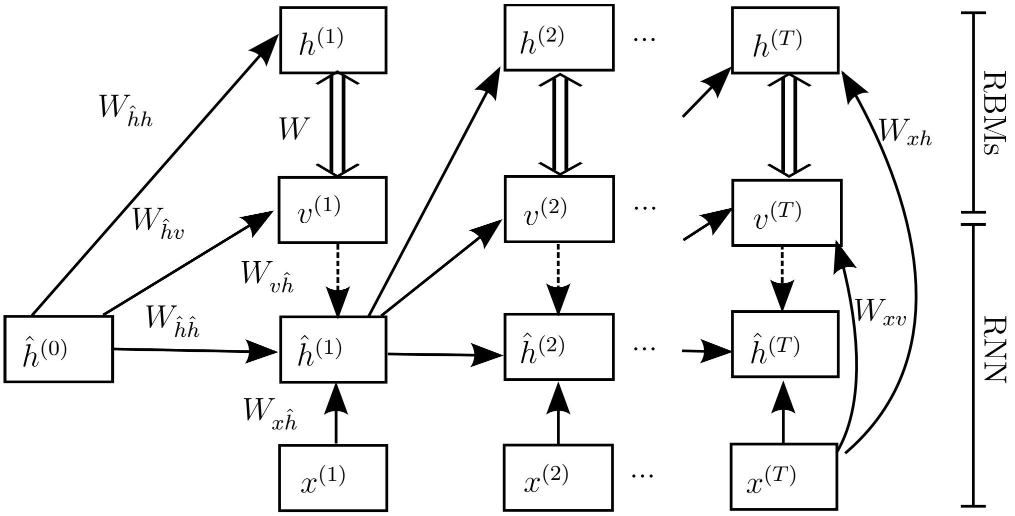


Figure 8.1: Graphical structure of the I/O RNN-RBM. Single arrows represent a deterministic function, double arrows represent the hidden-visible connections of an RBM, dotted arrows represent optional connections for temporal smoothing. The $x \to \{v, h\}$ connections have been omitted for clarity at each time step except the last.

$$b_v^{(t)} = b_v + W_{\hat{h}v}\hat{h}^{(t-1)} + W_{xv}x^{(t)} \tag{8.12}$$

where $\hat{h}^{(t)}$ are the hidden units of a single-layer RNN:

$$\hat{h}^{(t)} = \sigma(W_{v\hat{h}}v^{(t)} + W_{\hat{h}\hat{h}}\hat{h}^{(t-1)} + W_{x\hat{h}}x^{(t)} + b_{\hat{h}}) \tag{8.13}$$

where the indices of weight matrices and bias vectors have obvious meanings. The special case $W_{v\hat{h}} = 0$ gives rise to a transcription network without temporal smoothing. Gradient evaluation is based on the following general scheme:

1. Propagate the current values of the hidden units $\hat{h}^{(t)}$ in the RNN portion of the graph using (8.13),
2. Calculate the RBM or NADE parameters that depend on $\hat{h}^{(t)}, x^{(t)}$ (eq. 8.11-8.12) and obtain the log-likelihood gradient with respect to W, $b_v^{(t)}$ and $b_h^{(t)}$ (eq. 8.4 or eq. 8.7-8.9),
3. Propagate the estimated gradient with respect to $b_v^{(t)}, b_h^{(t)}$ backward through time (BPTT) (Rumelhart et al., 1986a) to obtain the estimated gradient with respect to the RNN parameters.

By setting $W = 0$, the I/O-RNN-RBM reduces to a regular RNN that can be trained with the cross-entropy cost:

$$L(\{v^{(t)}\}) = \frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{N} -v_j^{(t)}\log p_j^{(t)} - (1-v_j^{(t)})\log(1-p_j^{(t)}) \tag{8.14}$$

where $p^{(t)} = \sigma(b_v^{(t)})$ and equations (8.12) and (8.13) hold. We will use this model as one of our baselines for comparison.

A potential difficulty with this training scenario stems from the fact that since v 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 obviously depends on the ambiguousness of the audio and the intrinsic predictability of the output sequences, and can also be controlled by introducing noise to either $x^{(\tau)}$ or $v^{(\tau)}, \tau < t$, or by adding the regularization terms $\alpha(|W_{xv}|^2 + |W_{xh}|^2) + \beta(|W_{\hat{h}v}|^2 + |W_{\hat{h}h}|^2)$ to the objective function. It is trivial to revise the stochastic gradient descent updates to take those penalties into account.

8.3 Inference

A distinctive feature of our architecture are the (optional) connections $v \to \hat{h}$ that implicitly tie $v^{(t)}$ 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 $v^{(t)}$ requires the knowledge of the previous decisions on $v^{(\tau)}$ (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 [2]. We rather favor a global search approach analogous to the Viterbi algorithm for discrete-state HMMs. Since in the general case the partition function of the t^{th} RBM depends on $\mathcal{A}^{(t)}$, comparing sequence likelihoods becomes intractable, hence our use of the tractable NADE.

[2] Note that without temporal smoothing ($W_{v\hat{h}} = 0$), the $v^{(t)}, 1 \le t \le T$ would be conditionally independent given x and the prediction could simply be obtained separately at each time step t.

Algorithm 8.1 High-dimensional beam search

Find the most likely sequence $\{v^{(t)}, 1 \leq t \leq T\}$ under a model m with beam width w and branching factor K.

```
$q \leftarrow$ min-priority queue
$q$.insert$(0, m)$
for $t = 1 \ldots T$ do
    $q' \leftarrow$ min-priority queue of capacity $w$ $^\star$
    while $l, m \leftarrow q$.pop() do
        for $l', v'$ in $m$.find_most_probable$(K)$ do
            $m' \leftarrow m$ with $v^{(t)} := v'$
            $q'$.insert$(l + l', m')$
        end for
    end while
    $q \leftarrow q'$
end for
return $q$.pop()
```

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

Our algorithm is a variant of beam search for high-dimensional sequences, with beam width w and maximal branching factor K (Algorithm 8.1). 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 $v^{(t)}$ is allowed to change among siblings. This structure facilitates identifying the most promising paths by their cumulative log-likelihood. For high-dimensional output however, any non-leaf node has exponentially many children (2^N), which in practice limits the exploration to a fixed number K of siblings. This is necessary because enumerating the configurations at a given time step by decreasing likelihood is intractable (e.g. for RBM or NADE) and we must resort to stochastic search to form a pool of promising children at each node. Stochastic search consists in drawing S samples of $v^{(t)}|A^{(t)}$ and keeping the K unique most probable configurations. This procedure usually converges rapidly with $S \simeq 10K$ samples, especially with strong biases coming from the conditional terms. Note that $w = 1$ or $K = 1$ reduces to a greedy search, and $w = 2^{NT}, K = 2^N$ corresponds to an exhaustive breadth-first search.

When the output units $v_j^{(t)}, 0 \leq j < N$ are conditionally independent given $\mathcal{A}^{(t)}$, such as for a regular RNN (eq. 8.14), it is possible to enumerate configurations by decreasing likelihood using a dynamic programming approach (Algorithm 8.2). This very efficient algorithm in $O(K \log K + N \log N)$ is based on linearly growing priority queues, where K need not be specified in advance. Since inference is usually the bottleneck of the computation, this optimization makes it possible to use much higher beam widths w with unbounded branching factors for RNNs.

Algorithm 8.2 INDEPENDENT OUTPUTS INFERENCE

Enumerate the K most probable configurations of N independent Bernoulli random variables with parameters $0 < p_i < 1$.

```
$v_0 \leftarrow \{i : p_i \geq 1/2\}$
$l_0 \leftarrow \sum_i \log(\max(p_i, 1 - p_i))$
**yield** $l_0$, $v_0$
$L_i \leftarrow |\log \frac{p_i}{1-p_i}|$
sort $L$, store corresponding permutation $R$
$q \leftarrow$ min-priority queue
$q$.insert($L_0$, $\{0\}$)
**while** $l, v \leftarrow q$.pop() **do**
    **yield** $l_0 - l$, $v_0 \triangle R[v]$ $^\star$
    $i \leftarrow \max(v)$
    **if** $i + 1 < N$ **then**
        $q$.insert($l + L_{i+1}$, $v \cup \{i + 1\}$)
        $q$.insert($l + L_{i+1} - L_i$, $v \cup \{i + 1\} \setminus \{i\}$)
    **end if**
**end while**
```

$^\star A \triangle B \equiv (A \cup B) \setminus (A \cap B)$ denotes the symmetric difference of two sets. $R[v]$ indicates the R-permutation of indices in the set v.

A pathological condition that sometimes occurs with beam search over long sequences ($T \gg 200$) 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. Several strategies have been tried with moderate success in those cases, such as committing to the most likely path every M time steps (*periodic restarts* (Richter et al., 2010)), pruning similar paths, or pruning paths with identical τ previous time steps (the *local assumption*), where τ is a maximal time lag that the chosen architecture can reasonably describe (e.g. $\tau \simeq 200$ for RNNs trained with HF). It is also possible to initialize the search with

Dataset	HMM	RNN-RBM	Proposed
Piano-midi.de	59.5%	60.8%	64.1%
Nottingham	71.4%	77.1%	97.4%
MuseData	35.1%	44.7%	66.6%
JSB Chorales	72.0%	80.6%	91.7%

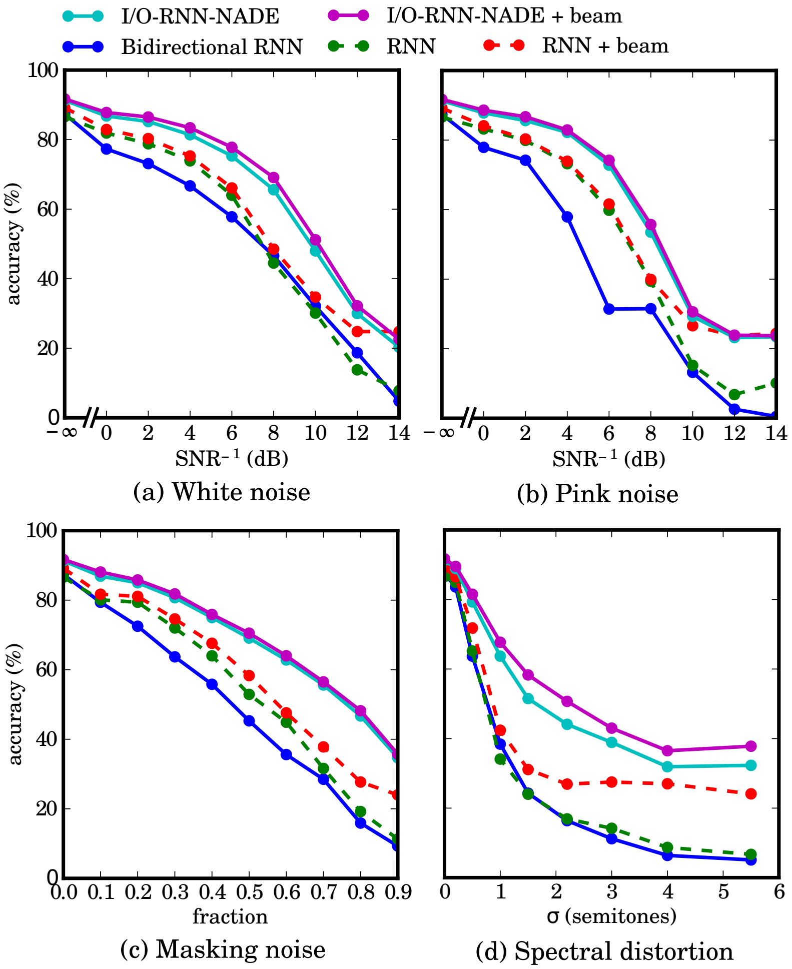
Table 8.1: Frame-level transcription accuracy obtained on four datasets by the Nam et al. algorithm with HMM temporal smoothing (Nam et al., 2011), using the RNN-RBM musical language model (Boulanger-Lewandowski et al., 2012b), or the proposed I/O RNN-NADE model.

Algorithm 8.1 then backtrack at each node iteratively, resulting in an anytime algorithm (Zhou and Hansen, 2005).

8.4 Experiments

In the following experiments, the acoustic input $x^{(t)}$ is constituted of powerful DBN-based learned representations (Nam et al., 2011). The magnitude spectrogram is first computed by the short-term Fourier transform using a 128 ms sliding Blackman window truncated at 6 kHz, normalized and cube root compressed to reduce the dynamic range. We apply PCA whitening to retain 99% of the training data variance, yielding roughly 30–70% dimensionality reduction. A DBN is then constructed by greedy layer-wise stacking of sparse RBMs trained in an unsupervised way to model the previous hidden layer expectation ($v^{l+1} \equiv \mathbb{E}[h^l|v^l]$) (Bengio, 2009). The whole network is finally finetuned with respect to a supervised criterion (e.g. eq. 8.14) and the last layer is then used as our input $x^{(t)}$ for the spectrogram frame at time t.

We evaluate our method on five datasets of varying complexity: Piano-midi.de, Nottingham, MuseData and JSB chorales (see Boulanger-Lewandowski et al., 2012b) which are rendered from piano and orchestral instrument soundfonts, and Poliner and Ellis (2007) that comprises synthesized sounds and real recordings. We use frame-level accuracy (Bay et al., 2009) for model evaluation. Hyperparameters are selected by a random search (Bergstra and Bengio, 2012) on predefined intervals to optimize validation set accuracy; final performance is reported on the test set.



(a) White noise

(b) Pink noise

(c) Masking noise

(d) Spectral distortion

Figure 8.2: Robustness to different types of noise of various RNN-based models on the JSB chorales dataset.

Table 8.1 compares the performance of the I/O RNN-RBM to the HMM baseline (Nam et al., 2011) and the RNN-RBM hybrid approach (Boulanger-Lewandowski et al., 2012b) on four datasets. Contrarily to the product of experts of (Boulanger-Lewandowski et al., 2012b), our model is jointly trained, which eliminates duplicate contributions to the energy function and the related increase in marginals temperature, and provides much better performance on all datasets, approximately halving the error rate in average over these datasets.

We now assess the robustness of our algorithm to different types of noise: white noise, pink noise, masking noise and spectral distortion. In masking noise, parts of the signal of exponentially distributed length ($\mu = 0.4$ s) are randomly destroyed (Vincent et al., 2008); spectral distortion consists in Gaussian pitch shifts of ampli-

SONIC (Marolt, 2004)	39.6%
Note events + HMM (Ryynänen and Klapuri, 2005)	46.6%
Linear SVM (Poliner and Ellis, 2007)	67.7%
DBN + SVM (Nam et al., 2011)	72.5%
BLSTM RNN (Böck and Schedl, 2012)	75.2%
AdaBoost cascade (Boogaart and Lienhart, 2009)	75.2%
I/O-RNN-NADE	79.1%

Table 8.2: Frame-level accuracy of existing transcription methods on the Poliner and Ellis (2007) dataset.

tude σ (Palomäki et al., 2004). The first two types are simplest because a network can recover from them by averaging neighboring spectrogram frames (e.g. Kalman smoothing), whereas the last two time-coherent types require higher-level musical understanding. We compare a bidirectional RNN (Böck and Schedl, 2012) adapted for frame-level transcription, a regular RNN with $v \to \hat{h}$ connections ($w = 2000$) and the I/O RNN-NADE ($w = 50, K = 10$). Figure 8.2 illustrates the importance of temporal smoothing connections and the additional advantage provided by conditional distribution estimators. Beam search is responsible for a 0.5% to 18% increase in accuracy over a greedy search ($w = 1$).

Figure 8.3 shows transcribed piano-rolls for various RNNs on an excerpt of Bach's chorale *Es ist genug* with 6 dB pink noise (Fig. 8.3(a)). We observe that a bidirectional RNN is unable to perform temporal smoothing on its own (Fig. 8.3(b)), and that even a post-processed version (Fig. 8.3(c)) can be improved by our global search algorithm (Fig. 8.3(d)). Our best model offers an even more musically plausible transcription (Fig. 8.3(e)). Finally, we compare the transcription accuracy of common methods on the Poliner & Ellis (Poliner and Ellis, 2007) dataset in Table 8.2, that highlights impressive performance.

8.5 Conclusions

We presented an input/output model for high-dimensional sequence transduction in the context of polyphonic music transcription. Our model can learn basic musical properties such as temporal continuity, harmony and rhythm, and efficiently search for the most musically plausible transcriptions when the audio signal is partially destroyed, distorted or temporarily inaudible. Conditional distribution estimators are important in this context to accurately describe the density of *multiple* potential paths given the weakly discriminative audio. This ability translates well to the transcription of “clean” signals where instruments may still be buried and notes occluded due to interference, ambient noise or imperfect recording techniques. Our algorithm approximately halves the error rate with respect to competing methods on five polyphonic datasets based on frame-level accuracy. Qualitative testing also suggests that a more musically relevant metric would enhance the advantage of our model, since transcription errors often constitute reasonable alternatives.

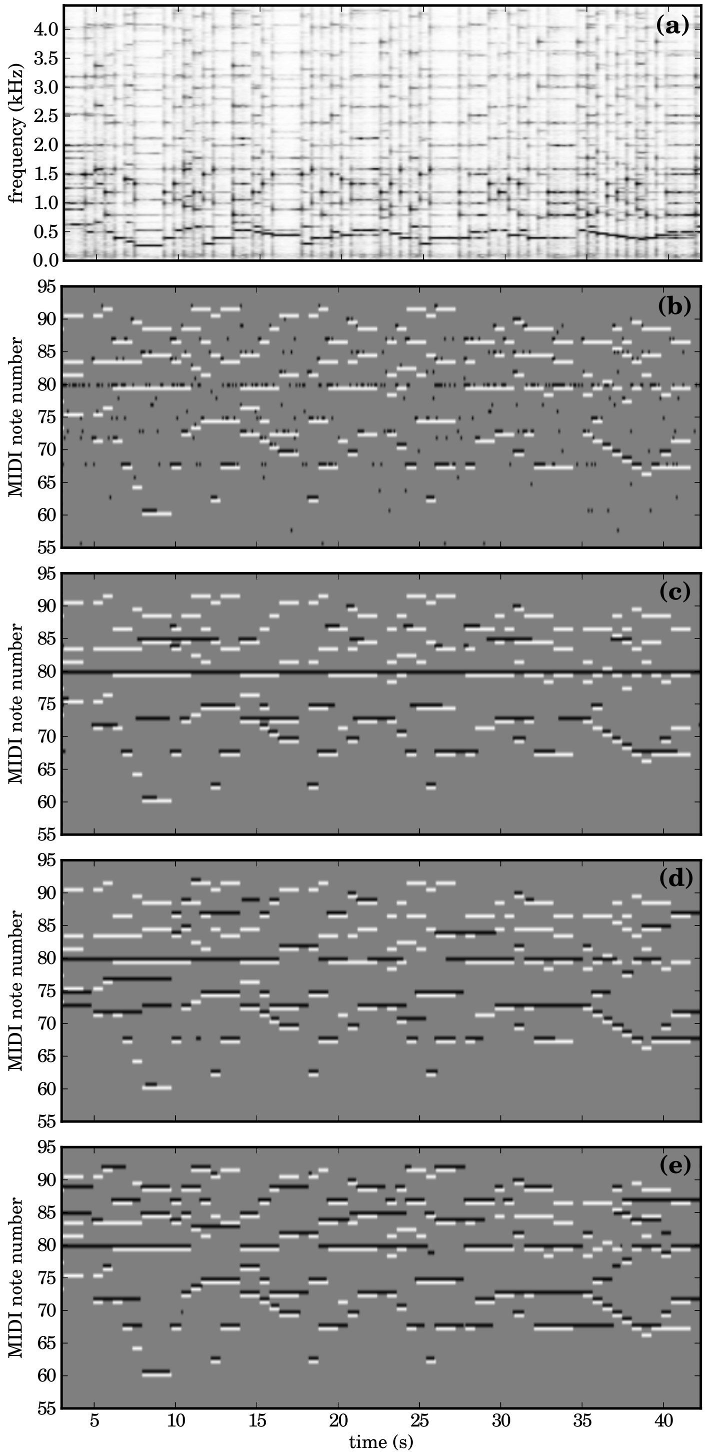


Figure 8.3: Demonstration of temporal smoothing on an excerpt of Bach's chorale *Es ist genug* (BWV 60.5) with 6 dB pink noise. Figure shows (a) the raw magnitude spectrogram, and transcriptions by (b) a bidirectional RNN, (c) a bidirectional RNN with HMM post-processing, (d) an RNN with $v \to \hat{h}$ connections ($w = 75$) and (e) I/O-RNN-NADE ($w = 20$, $K = 10$). Predicted piano-rolls (black) are interleaved with the ground-truth (white) for comparison.

9 Prologue to Fourth Article

9.1 Article Details

Audio Chord Recognition with Recurrent Neural Networks
Nicolas Boulanger-Lewandowski, Yoshua Bengio and Pascal Vincent
Published in *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)* in 2013.

9.2 Context

In this chapter, we apply the transduction framework developed in Chapter 8 to the task of recognizing chords from audio music, an active area of research in music information retrieval (Mauch, 2010; Harte, 2010). Contrarily to polyphonic transcription, the target sequence is over a dictionary of predefined chord labels, which imply a correspondence to the pitch classes present in the music but allows some room for error in the evaluation of the detected fundamental frequencies. Our RNN-based transduction framework is thus well suited for this task.

To compete with the state of the art, we will feed our RNN the most discriminative features possible obtained with deep neural networks. Deep learning has already been applied successfully to music (Hamel and Eck, 2010; Humphrey and Bello, 2012; Nam et al., 2011) and speech (Hinton et al., 2012) audio, and we will employ powerful enhancements with the use of multiscale aggregated features to describe contextual information (Bergstra et al., 2006), as well as a novel way to exploit prior information present in the chord label definitions to encourage the network to learn useful intermediate representations (Gülçehre and Bengio, 2013).

We also address a pathological condition that sometimes occurs with beam search when highly likely quasi-identical candidates surface at the top of the beam and saturate it. Those quasi-identical candidates typically differ only at a few time steps, e.g. due to a chord transition occuring infinitesimally earlier or later than in the next candidate, and tend to duplicate exponentially in the length T' of the sequence history. This causes less immediately promising but fundamentally different paths to be discarded prematurely, as well as requiring very large beams (e.g. $w > 1000$) in order to get optimal accuracy, which slows down the overall method.

9.3 Contributions

Our first contribution is the development of a comprehensive RNN-based system for chord recognition and the realization of experiments that demonstrate a performance competitive with the state of the art on the MIREX dataset. The second contribution is our proposed method to exploit the prior information contained in the active pitch classes present in each chord label by fine-tuning the DBN in two passes: first with respect to the intermediate targets, then with respect to the chord labels. Our third and most significant contribution is the development and validation of the dynamic programming-like beam search pruning technique presented in Section 10.4.3. It allows real-time decoding in live situations while actually *increasing* recognition accuracy, which is a drastic improvement over regular beam search.

9.4 Recent Developments

The dynamic programming inference algorithm introduced in this chapter will be reused and extended in the speech recognition system presented in Chapter 12. The current approach still suffers from the label bias and teacher forcing problems encountered in the preceding article (Chapter 8) and uses similar tricks of output noise and weight regularization to mitigate them. A proper solution to these problems will be investigated in Chapter 12.

10 Audio Chord Recognition with Recurrent Neural Networks

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.

10.1 Introduction

Automatic recognition of chords from audio music is an active area of research in music information retrieval (Mauch, 2010; Harte, 2010). 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 (Graves, 2012; Boulanger-Lewandowski et al., 2013b). 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 (Bengio, 2009) and its successful application to chord recognition (Humphrey and Bello, 2012), music annotation and auto-tagging (Hamel and Eck, 2010), polyphonic music transcription (Nam et al., 2011) and speech recognition (Hinton et al., 2012), 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 (Bergstra et al., 2006; Hamel et al., 2012; Dahl et al., 2012). We also exploit prior information (Gülçehre and Bengio, 2013) in the form of pitch class targets derived from chord labels, known to be a useful intermediate representation for chord recognition (e.g. (Chen et al., 2012)).

Recurrent neural networks (RNN) (Rumelhart et al., 1986a) 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 (Mozer, 1994; Eck and Schmidhuber, 2002; Boulanger-Lewandowski et al., 2012b; Böck and Schedl, 2012; Boulanger-Lewandowski et al., 2013b). While RNN-based musical language models significantly surpass popular alternatives like hidden Markov models (HMM) (Boulanger-Lewandowski et al., 2012b) and offer a principled way to combine the acoustic and language models(Boulanger-Lewandowski et al., 2013b), 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 10.2, we present our feature extraction pipeline based on deep learning. In Sections 10.3 and 10.4 we introduce the recurrent neural network model and the proposed inference procedure. We describe our experiments and evaluate our method in Section 10.5.

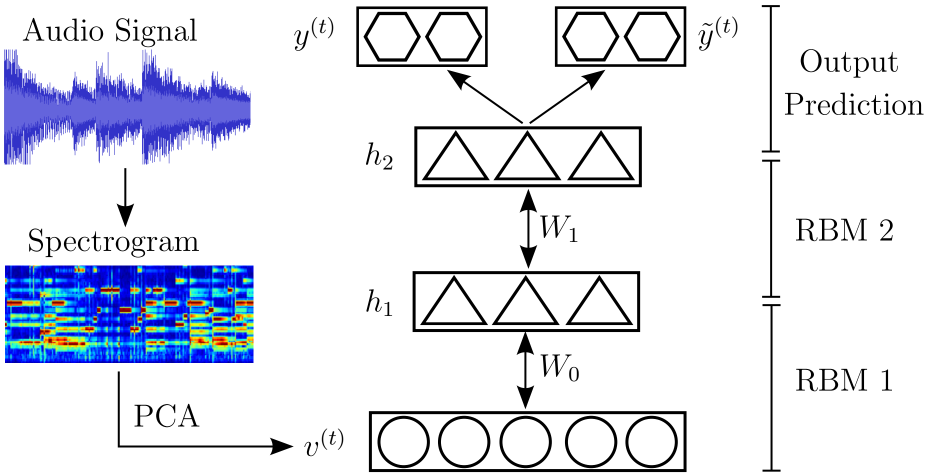


Figure 10.1: Pre-processing pipeline to learn deep audio features with intermediate targets $z^{(t)}, \tilde{z}^{(t)}$. Single arrows represent a deterministic function, double-ended arrows represent the hidden-visible connections of an RBM.

10.2 Learning deep audio features

10.2.1 Overview

The overall feature extraction pipeline is depicted in Figure 10.1. 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 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.

10.2.2 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 (Bengio, 2009).

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) (Smolensky, 1986) to model the joint distribution of the previous layer's units in a deep belief network (DBN) (Hinton et al., 2006) (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:

$$h_{l+1} = \sigma(W_l h_l + b_l) \tag{10.1}$$

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:

$$L(v^{(t)}, z^{(t)}) = -\sum_{j=1}^{N} z_j^{(t)} \log y_j^{(t)} + (1 - z_j^{(t)}) \log(1 - y_j^{(t)}) \tag{10.2}$$

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.

10.2.3 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 (Gülçehre and Bengio, 2013). 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.

10.2.4 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 $\bar{x}, \tilde{x}$ (Bergstra et al., 2006) computed over windows of varying sizes L (Hamel et al., 2012) and offsets τ relative to the current time step t:

$$\bar{x}^{(t)} = \Big\{ \sum_{\Delta t = -\lfloor L/2 \rfloor}^{\lfloor (L-1)/2 \rfloor} x^{(t-\tau+\Delta t)}, \forall (L, \tau) \Big\} \tag{10.3}$$

$$\tilde{x}^{(t)} = \Big\{ \sum_{\Delta t = -\lfloor L/2 \rfloor}^{\lfloor (L-1)/2 \rfloor} (x^{(t-\tau+\Delta t)} - \bar{x}_{L,\tau}^{(t)})^2, \forall (L, \tau) \Big\} \tag{10.4}$$

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.

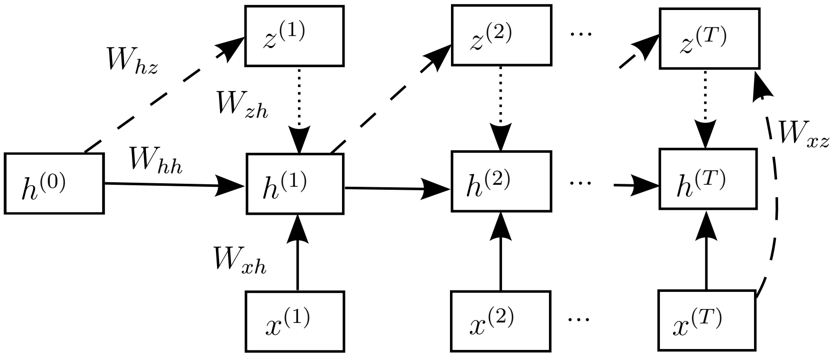


Figure 10.2: Graphical structure of the RNN. Single arrows represent a deterministic function, dotted arrows represent optional connections for temporal smoothing, dashed arrows represent a prediction. The $x \to z$ connections have been omitted for clarity at each time step except the last.

10.3 Recurrent neural networks

10.3.1 Definition

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

$$P(z|x) = \prod_{t=1}^{T} P(z^{(t)}|\mathcal{A}^{(t)}) \tag{10.5}$$

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 (10.9).

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

$$h^{(t)} = \sigma(W_{zh}z^{(t)} + W_{hh}h^{(t-1)} + W_{xh}x^{(t)} + b_h) \tag{10.6}$$

where the indices of weight matrices and bias vectors have obvious meanings. Its graphical structure is illustrated in Figure 10.2.

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

$$y^{(t)} = s(W_{hz}h^{(t-1)} + W_{xz}x^{(t)} + b_z) \tag{10.7}$$

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

$$(s(a))_j \equiv \frac{\exp(a_j)}{\sum_{j'=1}^{N} \exp(a_{j'})}, \tag{10.8}$$

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:

$$P(z^{(t)}|\mathcal{A}^{(t)}) = z^{(t)} \cdot y^{(t)}. \tag{10.9}$$

10.3.2 Training

The RNN model can be trained by maximum likelihood with the following cost (replacing eq. 10.2):

$$L(x, z) = -\sum_{t=1}^{T} \log(z^{(t)} \cdot y^{(t)}) \tag{10.10}$$

where the gradient with respect to the model parameters is obtained by backpropagation through time (BPTT) (Rumelhart et al., 1986a).

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 (Bengio et al., 1994). Several strategies can be used to help reduce these difficulties including gradient clipping, leaky integration, sparsity and Nesterov momentum (Bengio et al., 2013).

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 (Graves, 2012; Boulanger-Lewandowski et al., 2013b), 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 $z \rightarrow h$ 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.

10.4 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.

10.4.1 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:

$$P(x^{(t)}|\{x^{(\tau)}, \tau \neq t\}, z) = P(x^{(t)}|z^{(t)}) \tag{10.11}$$

$$P(z^{(t)}|\{z^{(\tau)}, \tau < t\}) = P(z^{(t)}|z^{(t-1)}) \tag{10.12}$$

where the emission probability can be formulated using Bayes' rule (Hinton et al., 2012):

$$P(x^{(t)}|z^{(t)}) \propto \frac{P(z^{(t)}|x^{(t)})}{P(z^{(t)})} \tag{10.13}$$

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

$$P(z^{(t)}, x^{(t)}|\{z^{(\tau)}, \tau < t\}) \propto \frac{P(z^{(t)}|x^{(t)})}{P(z^{(t)})} P(z^{(t)}|z^{(t-1)}) \tag{10.14}$$

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.

10.4.2 Beam search

An established algorithm for sequence transduction with RNNs is beam search (Algorithm 10.1) (Graves, 2012; Boulanger-Lewandowski et al., 2013b). 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 10.1 Beam search

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

```
$q \leftarrow$ priority queue
$q$.insert$(0, \{\})$
for $t = 1 \dots T$ do
    $q' \leftarrow$ priority queue of capacity $w$ $^\star$
    for $z$ in $C$ do
        for $l, s$ in $q$ do
            $q'$.insert$(l + \log P(z^{(t)} = z|x, s), \{s, z\})$
        end for
    end for
    $q \leftarrow q'$
end for
return $q$.max()
```

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

10.4.3 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 (i.e. assuming that subsequences of the global optimum are always optimal) 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:

$$l_j^{(t)} = l_{k_j^{(t)}}^{(t-1)} + \log P(z^{(t)} = j|x, s_{k_j^{(t)}}^{(t-1)}) \tag{10.15}$$

$$s_j^{(t)} = \{s_{k_j^{(t)}}^{(t-1)}, j\} \tag{10.16}$$

$$\text{with } k_j^{(t)} \equiv \operatorname*{argmax}_{k=1}^{N} \left[l_k^{(t-1)} + \log P(z^{(t)} = j|x, s_k^{(t-1)})\right] \tag{10.17}$$

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 10.2). 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.

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

Algorithm 10.2 Dynamic programming inference

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

```
q ← priority queue
q.insert($0, \{\}$)
for $t = 1 \dots T$ do
    $q'$ ← priority queue of capacity $w$
    for $z$ in $C$ do
        $l, s \leftarrow \text{argmax}_{(l,s)\in q} \left[ l + \log P(z^{(t)} = z | x, s) \right]$
        $q'$.insert($l + \log P(z^{(t)} = z | x, s), \{s, z\}$)
    end for
    $q \leftarrow q'$
end for
return $q$.max()
```

globally optimal solution z^*, but is referred to as DP due to its strong similarity to the Viterbi recurrence relations.

10.5 Experiments

10.5.1 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:

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

where S represents the 12 pitch classes and 'N' is the *no-chord* label (Harte, 2010; Mauch, 2010). 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 (Ni et al., 2012; Mauch and Dixon, 2010).

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

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.

10.5.2 Results

In Table 10.1, 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 (Ni et al., 2012) and DHMM (Chen et al., 2012) state-of-the-art methods are also provided for comparison.

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 (Brown, 1987), the relatively poor performance of the HMM may be due to the context information added to the input $x^{(t)}$ in equations (10.3-10.4). When the input includes information from neighboring frames, the independence property (10.11) breaks down, making it difficult to combine the classifier with the language model in equation (10.14). Intuitively, multiplying the predictions $P(z^{(t)}|x^{(t)})$ and $P(z^{(t)}|z^{(t-1)})$ to estimate

Model	Smoothing	OR	WAOR
DBN-1	None	65.8%	65.2%
	Kernel	75.2%	74.6%
	HMM	74.3%	74.2%
DBN-2	None	68.0%	67.3%
	Kernel	78.1%	77.6%
	HMM	77.3%	77.2%
RNN	DP	**80.6%**	80.4%
HPA (Ni et al., 2012)	HMM	79.4%	78.8%
DHMM (Chen et al., 2012)	HMM	N/A	**84.2%**†

Table 10.1: 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 (Ni et al., 2012) and DHMM (Chen et al., 2012) state-of-the-art methods. †4-fold cross-validation result taken from (Chen et al., 2012).

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 10.2), 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 (Mauch and Dixon, 2010)	80.2%	79.5%
GMM + HMM (Khadkevich and Omologo, 2011)	82.9%	81.6%
HPA (Ni et al., 2012)	83.5%	82.7%
Proposed (DBN-2)	89.5%	89.8%
Proposed (RNN)	**93.5%**	**93.6%**

Table 10.2: 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 10.3 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 10.4.3 holds well in practice.

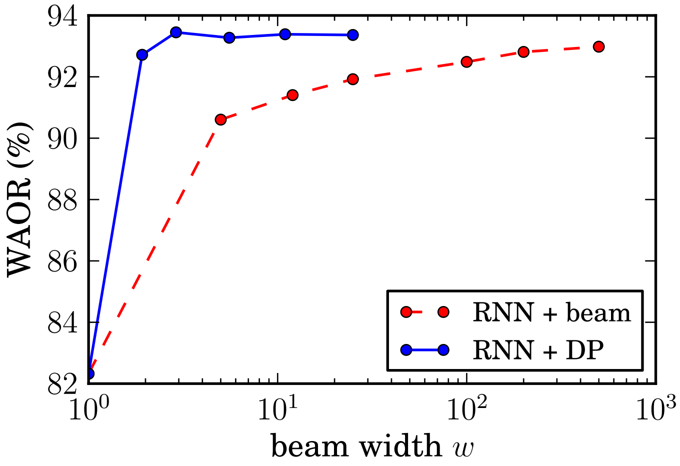


Figure 10.3: WAOR obtained on the MIREX dataset with the beam search and dynamic programming algorithms as a function of the (effective) beam width w.

10.6 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.

11 Prologue to Fifth Article

11.1 Article Details

Phone Sequence Modeling with Recurrent Neural Networks
Nicolas Boulanger-Lewandowski, Jasha Droppo, Mike Seltzer and Dong Yu
Published in *Proceedings of the 39th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* in 2014.

11.2 Context

In this chapter, we investigate phone sequence modeling with RNNs for speech recognition. Existing speech recognition systems commonly comprise three fundamental modules: an acoustic model, a phonetic model and a language model. Traditionally, an HMM is stacked on top of a frame-level classifier (e.g. Dahl et al., 2012), which translates into an implicit N-gram phonetic model. This simple approach leaves much room for improvement, but it is not clear how important a robust phonetic model really is *in complementarity* to powerful acoustic and language models. Efficiency of decoding is also a determining factor for an RNN-based solution.

The label bias problem (McCallum et al., 2000) could be partially controlled with output noise and weight regularization in Chapters 8 and 10 but could never be eliminated completely. Several approaches have been proposed to circumvent this problem, such as conditional random fields (Lafferty et al., 2001) or modeling *unaligned* phonetic sequences with an implicit exponential duration model (Graves, 2012; Graves et al., 2013). We propose a generalization of the HMM that naturally enforces a proper weighting of the acoustic and symbolic predictors and allows the probability flow of a candidate solution to vary according to the acoustic observations (Lafferty et al., 2001), eliminating the label bias problem.

A practical difference with the chord recognition task tackled in Chapter 10 is that time alignments are usually not provided in ground truth phone sequences, i.e. the precise onset and duration of each phone is unknown. This requires the development of a fast optimal alignment procedure to be used during training.

11.3 Contributions

We propose a hybrid architecture to combine an RNN phonetic model with an arbitrary frame-level acoustic classifier such as a DNN in a way that circumvents the label bias problem, and we provide efficient procedures for training via hard expectation-maximization, decoding via pruned beam search and optimal alignment via a two-pass dynamic programming algorithm. The decoding algorithm has the same time complexity (and similar run time) as Viterbi, and leads to improvements of 2–10% in phone accuracy and 3% in word error rate on the TIMIT and Switchboard-mini datasets in complementarity to a DNN acoustic classifier and 3-gram language model. This suggests that phone sequence modeling is an essential component of speech recognition and that RNNs can readily replace HMMs in current state-of-the-art systems, such as those obtained with dropout (Dahl et al., 2013).

Note that the product of model probabilities in equation (12.12) is similar in spirit but technically different than the product of experts in equation (4.23) because in the current chapter the resulting product is not renormalized, which is key to avoid the label bias problem.

11.4 Recent Developments

This paper will be presented at ICASSP in May 2014.

12 Phone sequence modeling with recurrent neural networks

In this paper, we investigate phone sequence modeling with recurrent neural networks in the context of speech recognition. We introduce a hybrid architecture that combines a phonetic model with an arbitrary frame-level acoustic model and we propose efficient algorithms for training, decoding and sequence alignment. We evaluate the advantage of our phonetic model on the TIMIT and Switchboard-mini datasets in complementarity to a powerful context-dependent deep neural network (DNN) acoustic classifier and a higher-level 3-gram language model. Consistent improvements of 2–10% in phone accuracy and 3% in word error rate suggest that our approach can readily replace HMMs in current state-of-the-art systems.

12.1 Introduction

Automatic speech recognition is an active area of research in the signal processing and machine learning communities (Baker et al., 2009). Existing approaches are commonly based on three fundamental modules: (1) an *acoustic* model that focuses on the discriminative aspect of the audio signal, (2) a *phonetic* model that attempts to describe the temporal dependencies associated with the sequence of phone labels, and (3) a *language* model that describes the higher-level dependencies between words and sentences. In this work, we wish to replace the popular hidden Markov model (HMM) approach with a more powerful neural network-based phonetic model.

Recurrent neural networks (RNN) (Rumelhart et al., 1986a) 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 phone labels in a spoken utterance, 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 and language applications (Boulanger-Lewandowski et al., 2012b; Mikolov et al., 2011; Bengio et al., 2013).

While RNN-based language models significantly surpass popular alternatives like HMMs, it is not immediately obvious how to combine the acoustic and phonetic models under a single training objective. The simple approach of multiplying the predictions of both models before renormalizing as in a maximum entropy Markov model (McCallum et al., 2000) often results in the so-called *label bias problem* where the symbolic information overwhelms the acoustic information in low-entropy sequences with frequently reoccuring symbols (Lafferty et al., 2001). Several attempts have been made to reduce those difficulties, such as with conditional random fields (Lafferty et al., 2001), regularization of the symbolic and acoustic sources (Boulanger-Lewandowski et al., 2013b), by increasing the entropy per time step with a lower temporal resolution (Boulanger-Lewandowski et al., 2013a), modeling *unaligned* phonetic sequences with an implicit exponential duration model (Graves, 2012; Graves et al., 2013), or with the popular approach of stacking an HMM on top of a frame-level classifier (e.g. (Dahl et al., 2012)). In this paper, we propose an alternative approach that enforces a proper weighting of the acoustic and symbolic predictors and allows the probability flow of a candidate solution to vary according to the acoustic observations (Lafferty et al., 2001). Our hybrid architecture is a generative model that generalizes the HMM and that can be trained similarly following the expectation-maximization principle while exploiting the predictive power of RNNs in describing complex temporal dependencies. An advantage of our design not present in (Graves et al., 2013) is the possibility of leveraging *arbitrary* frame-level acoustic classifiers such as a DNN trained with dropout or advanced optimization techniques (Dahl et al., 2013). We also propose efficient inference algorithms for decoding and optimal sequence alignment inspired from Viterbi decoding. Finally, we investigate the extent to which phone sequence modeling is relevant in complementarity to powerful context-dependent acoustic classifiers and higher-level language models.

The remainder of the paper is organized as follows. In sections 12.2 and 12.3 we introduce the RNN architecture and our hybrid phone sequence model. In sections 12.4 and 12.5 we detail our decoding and alignment algorithms. Finally, we present our methodology and results in section 12.6.

12.2 Recurrent neural networks

The RNN formally defines the distribution of the output sequence $z \equiv \{z^{(t)} \in C, t \leq T\}$ of length T, where C is the dictionary of possible phone labels ($|C| = N$):

$$P(z) = \prod_{t=1}^{T} P(z^{(t)}|\mathcal{A}^{(t)}) \tag{12.1}$$

where $\mathcal{A}^{(t)} \equiv \{z^{(\tau)}|\tau < t\}$ is the sequence history at time t, and $P(z^{(t)}|\mathcal{A}^{(t)})$ is the conditional probability of observing $z^{(t)}$ according to the model, defined below in equation (12.5).

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

$$h^{(t)} = \sigma(W_{zh}z^{(t-1)} + W_{hh}h^{(t-1)} + b_h) \tag{12.2}$$

where the indices of weight matrices and bias vectors have obvious meanings.

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

$$y^{(t)} = s(W_{hz}h^{(t)} + W_{zz}z^{(t-1)} + b_z) \tag{12.3}$$

where the W_{zz} matrix is useful to explicitly disallow certain state transitions by setting the corresponding entries to very large negative values, and $s(a)$ is the softmax function of an activation vector a:

$$(s(a))_j \equiv \frac{\exp(a_j)}{\sum_{j'=1}^{N} \exp(a_{j'})}, \tag{12.4}$$

and should be as close as possible to the target vector $z^{(t)}$. In the case of multiclass classification problems such as frame-level phone recognition, the target is a one-hot vector and the likelihood of an observation is given by the dot product:

$$P(z^{(t)}|\mathcal{A}^{(t)}) = z^{(t)} \cdot y^{(t)}. \tag{12.5}$$

The RNN model can be trained by maximum likelihood with the cross-entropy cost:

$$L(z) = -\sum_{t=1}^{T} \log(z^{(t)} \cdot y^{(t)}) \tag{12.6}$$

where the gradient with respect to the model parameters is obtained by backpropagation through time (BPTT) (Rumelhart et al., 1986a).

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 (Bengio et al., 1994). Several strategies can be used to help reduce these difficulties including gradient clipping, leaky integration, sparsity and Nesterov momentum (Bengio et al., 2013).

12.3 Phone sequence modeling

In this section, we generalize the popular technique of superposing an HMM to an acoustic model by replacing the HMM with an arbitrary phonetic model. This will allow to exploit the power of RNNs for phone modeling while providing a principled way to combine the two models.

Our hybrid acoustic-phonetic sequence model is a graphical model composed of an underlying phone sequence z:

$$P(z^{(t)}|\{x^{(\tau)}, \tau < t\}, \mathcal{A}^{(t)}) = P(z^{(t)}|\mathcal{A}^{(t)}) \tag{12.7}$$

and an acoustic sequence x *emitted* given the phone sequence:

$$P(x^{(t)}|\{x^{(\tau)}, \tau \neq t\}, z) = P(x^{(t)}|z^{(t)}). \tag{12.8}$$

The emission probability (12.8) can be reformulated using Bayes' rule (Hinton et al., 2012):

$$P(x^{(t)}|z^{(t)}) \propto \frac{P(z^{(t)}|x^{(t)})}{P(z^{(t)})} \tag{12.9}$$

where $P(z^{(t)}|x^{(t)})$ is the output of an acoustic classifier, $P(z^{(t)})$ is the marginal distribution of phones and constant terms given x have been removed. This adjustment is referred to as scaled likelihood estimation in (Dahl et al., 2012).

The next-step phone sequence distribution has a general expression in the right-hand side of equation (12.7) to accomodate different phonetic models. For an HMM, this distribution depends only on $z^{(t-1)}$:

$$P(z^{(t)} = i|\mathcal{A}^{(t)}) = \begin{cases} T_{z^{(t-1)},i} & \text{if } t > 0 \\ \pi_i & \text{if } t = 0 \end{cases} \tag{12.10}$$

where $T_{j,i}$ is the row-normalized transition matrix and π_i the initial occupancy of phone i. In our case, we will replace (12.10) with the distribution of an RNN (eq. 12.5) which depends on the full sequence history $\mathcal{A}^{(t)}$.

By combining equations (12.7)-(12.9), we obtain the conditional distribution over phones z given the input x:

$$P(z|x) \propto P(z, x) = \prod_{t=1}^{T} P(z^{(t)}, x^{(t)}|\mathcal{A}^{(t)}) \tag{12.11}$$

$$P(z^{(t)}, x^{(t)}|\mathcal{A}^{(t)}) \propto \frac{P(z^{(t)}|x^{(t)})}{P(z^{(t)})} P(z^{(t)}|\mathcal{A}^{(t)}) \tag{12.12}$$

which can be interpreted as the output of the hybrid model.

As argued previously (Brown, 1987), a limitation of the hybrid model occurs when the acoustic model has access to contextual information when predicting $z^{(t)}$, either directly in the form of an input window around $x^{(t)}$ or indirectly via the hidden state of an RNN. When the input includes information from neighboring frames, the independence assumption (12.8) breaks down, making it difficult to combine the two models in equation (12.12). Intuitively, multiplying the predictions $P(z^{(t)}|x^{(t)})$ and $P(z^{(t)}|\mathcal{A}^{(t)})$ to estimate the joint distribution will count certain factors twice since both models have been trained separately. Note that the marginals $P(z^{(t)})$ are counted only once with scaled likelihood estimation (eq. 12.9), but it is reasonable to expect that certain temporal dependencies will be captured by both models. In our experiments, we found that this conceptual difficulty surprisingly did not prevent good performance. Furthermore, the alternative approach of multiplying the two predictions and renormalizing at each time step in order to train the system jointly (Graves et al., 2006; Graves, 2012; Graves et al., 2013) suffered heavily from the label bias problem, and we found it crucial not to renormalize the two distributions to achieve good performance. Note that the transducer approach in (Graves et al., 2013) circumvents the label bias problem by modeling *unaligned* phone sequences with an implicit exponential duration model. This has the advantage of significantly increasing the conditional entropy of each time step, but prevents the model from taking phone duration into account.

During training, we wish to maximize the log-likelihood $\log P(x, z)$ of training example pairs x, z.[1] It is easy to see from equations (12.11) and (12.12) that a stochastic gradient ascent update involves terms associated with the phonetic and acoustic models that can be computed separately:

$$\frac{\partial \log P(x,z)}{\partial \Theta_a} = \frac{\partial}{\partial \Theta_a} \sum_{t=1}^{T} \log P(z^{(t)} | x^{(t)}) \tag{12.13}$$

$$\frac{\partial \log P(x,z)}{\partial \Theta_p} = \frac{\partial}{\partial \Theta_p} \sum_{t=1}^{T} \log P(z^{(t)} | \mathcal{A}^{(t)}) \tag{12.14}$$

where Θ_a, Θ_p denote the parameters of the acoustic and phonetic models respectively.

When only unaligned phone sequences $\bar{z} \equiv \{\bar{z}^{(u)}, u \leq U\}$ of length U are available during training, the hard expectation-maximization (EM) approach can be adopted, by regarding the alignments as missing data. After initializing the aligned sequences z from a flat start or another existing method, we alternate updates to the model parameters (M step) and to the estimated alignments given the current parameters (E step) as described in section 12.5. Both of these steps are guaranteed to increase the training objective $\log P(x, z)$ unless a local maximum is already reached.[2]

[1] We refer to the joint distribution $P(x, z)$ and not merely to $P(z|x)$, because in the hybrid model $P(x)$ can be regarded as uniform with no associated parameterization.

[2] More accurately, *hard*-EM guarantees the increase of $\log P(x, \{\bar{z}^{(u_t^*)}\})$ with the notation of section 12.5, i.e. where the optimal alignment is given.

12.4 Decoding

In our architecture, the phonetic model implicitly ties $z^{(t)}$ to its history $\mathcal{A}^{(t)}$ and encourages 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 to infer the sequence $z^* \equiv \{z^{(t)*} | t \leq T\}$ with maximal probability given the input.

For HMM phonetic models, the distribution in equation (12.12) becomes:

$$P(z^{(t)}|x^{(t)}, \mathcal{A}^{(t)}) \propto \frac{P(z^{(t)}|x^{(t)})}{P(z^{(t)})} P(z^{(t)}|z^{(t-1)}). \tag{12.15}$$

Since it 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.

The inference algorithm we propose for RNN phonetic models is based on a dynamic programming-like (DP) pruned beam search introduced in (Boulanger-Lewandowski et al., 2013a). 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 $\mathcal{A}^{(t+1)}$. Note that $w = 1$ reduces to a greedy search, and $w = N^T$ corresponds to an exhaustive breadth-first search.

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. A natural extension to beam search is to make a better use of the available width w via pruning. A particularly efficient pruning strategy is to consider only the most promising path out of all partial paths with identical $z^{(t)}$ when making a decision at time t. This leads to the solution of keeping track of the N most likely paths arriving at each possible label $j \in C$ with the recurrence relations:

$$l_j^{(t)} = l_{k_j^{(t)}}^{(t-1)} + \log P(z^{(t)} = j|x, s_{k_j^{(t)}}^{(t-1)}) \tag{12.16}$$

$$s_j^{(t)} = \{s_{k_j^{(t)}}^{(t-1)}, j\} \tag{12.17}$$

$$\text{with } k_j^{(t)} \equiv \operatorname*{argmax}_{k=1}^{N} \left[l_k^{(t-1)} + \log P(z^{(t)} = j|x, s_k^{(t-1)})\right] \tag{12.18}$$

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 (Boulanger-Lewandowski et al., 2013a). In our case, $P(z^{(t)} = j|x, s_k^{(t-1)})$ is given by equation (12.12): since the acoustic prediction and the marginal distribution do not depend on $\mathcal{A}^{(t)}$, we can compute those contributions in advance.

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.

12.5 Optimal alignment

In this section, we propose an algorithm to search for the aligned phone sequence $z \equiv \{z^{(t)}|t \leq T\}$ with maximal probability $P(z|x)$ according to a trained model (eq. 12.11), that is consistent with a given unaligned phone sequence $\bar{z} \equiv \{\bar{z}^{(u)}|u \leq U\}$ where $U < T$. The sequences z and $\bar{z}$ are said to be consistent if there exists an alignment $a \equiv \{u_t|t \leq T\}$ satisfying $u_1 = 1$, $u_T = U$ and $u_t - u_{t-1} \in \{0, 1\}$ for which $z^{(t)} = \bar{z}^{(u_t)}, \forall t \leq T$. The objective is to find the optimal alignment a^*.

Since an exact solution is intractable in the general case that the predictions fully depend on the sequence history, we hypothesize that it is sufficient to consider only the most promising path out of all partial paths with identical u_t when making a decision at time t.[3] Under this assumption, any subsequence $\{u_t^*|t \leq T'\}$ of the global optimum $\{u_t^*|t \leq T\}$ ending at time $T' < T$ must also be optimal under the constraint $u_{T'} = u_{T'}^*$. This last constraint is necessary to avoid a greedy solution. Setting $T' = T - 1$ leads to the DP-like solution of keeping track of the (at most) U most likely paths arriving at each possible index u, $\max(1, U - T + t) \leq u \leq \min(U, t)$ with the recurrence relations:

$$l_u^{(t)} = l_{k_u^{(t)}}^{(t-1)} + \log P(z^{(t)} = \bar{z}^{(u)}|x, s_{k_u^{(t)}}^{(t-1)}) \tag{12.19}$$

$$s_u^{(t)} = \{s_{k_u^{(t)}}^{(t-1)}, \bar{z}^{(u)}\} \tag{12.20}$$

$$\text{with } k_u^{(t)} \equiv \underset{k \in \{u-1, u\}}{\operatorname{argmax}} \left[l_k^{(t-1)} + \log P(z^{(t)} = \bar{z}^{(u)}|x, s_k^{(t-1)}) \right] \tag{12.21}$$

and initial conditions $l_u^{(0)} = 0, s_u^{(0)} = \{\}$, where the variables $l_j^{(t)}, s_j^{(t)}$ are defined similarly as in equations (12.16)-(12.18). The optimal aligned sequence is then given by $z^* \simeq s_U^{(T)}$. This algorithm has a time complexity $O(TU)$ independent of N.

Since finding an optimal alignment in the inner loop of an EM iteration can be prohibitive, we can further postulate that the optimal alignment a^* is close to an approximate alignment a' that can be computed much more cheaply. Typically, a' would be obtained by an acoustic model whose predictions depend only on x, eliminating the need to maintain the hidden states of multiple RNNs. Assuming that the distance between a^* and a' is δ:

$$|a^*, a'| \equiv \max_{t=1}^{T} |u_t^* - u_t'| = \delta, \tag{12.22}$$

the range of plausible values for u can be significantly reduced in equations (12.19)-(12.21). Values of δ as low as 2–4 were found to work well in practice, producing identical alignments in a majority of cases with less than 10% of the computation.

[3] Replacing the pruning condition on u_t with a condition on $z^{(t)}$ as for decoding is not as effective because $\tilde{u}_t \neq \hat{u}_t, \tilde{z}^{(t)} = \hat{z}^{(t)}$ for two candidates $\tilde{a}, \hat{a}$ indicate a different number of emitted symbols and thus fundamentally different alignments that should not be pruned against each other.

12.6 Experiments

In this section, we evaluate the performance of our RNN phonetic model and hybrid training procedure relatively to a baseline HMM system. We use two datasets to evaluate our method: the TIMIT corpus and the 30 hour "mini-train" subset of the Switchboard corpus. We report phone accuracy on the TIMIT data, which includes expertly-annotated phone sequences. We report phone accuracy and word accuracy on the Switchboard data, where the correct phonetic transcription is approximated by a dictionary-based alignment of the data by our baseline DNN + HMM system.

The TIMIT experiments rely on a 123 dimensional acoustic feature vector, calculated as 40 dimensional mel-frequency log-filterbank features, together with an energy measure and first and second temporal derivatives. The Switchboard experiments use a 52 dimensional acoustic feature vector, consisting of a basic 13-dimensional PLP cepstral vector together with its first, second, and third temporal derivatives.

We consider three acoustic models: a simple logistic regression (LR) classifier, an RNN using x as input (replacing $z^{(t-1)}$ in eq. 12.2) and a DNN with 4×1024 (TIMIT) or 5×2048 (Switchboard) hidden units trained with context-dependent triphones. The DNN features are the activations of the final hidden layer of the fully trained model. For each acoustic model, we compare three phonetic models: an HMM baseline, an RNN trained with fixed baseline alignments, and an RNN trained with our hybrid EM procedure. Early stopping is performed based on the cross-entropy of a held-out development set, which was randomly selected from 5% of the training set for Switchboard. The phone accuracy is determined as:

$$PA = 1 - \frac{\sum_{\bar{z},\bar{z}_0} L(\bar{z},\bar{z}_0)}{\sum_{\bar{z}_0} |\bar{z}_0|} \tag{12.23}$$

where $L(\cdot,\cdot)$ is the Levenshtein distance between two sequences and $\bar{z}, \bar{z}_0$ represent respectively the predicted and ground-truth sequences.

Developement and test phone accuracies are presented for the two datasets in Tables 12.1 and 12.2 for different combinations of acoustic and phonetic models. We observe consistent improvements with the RNN phonetic model, especially when trained using the hybrid procedure, attaining accuracies between 2–10% over the

Acoustic	HMM		RNN		Hybrid	
model	(dev)	(test)	(dev)	(test)	(dev)	(test)
LR	62.6	61.8	63.8	62.8	65.3	63.5
RNN	69.9	68.6	70.6	69.4	74.2	72.2
DNN	79.0	77.1	79.8	77.9	**80.4**	**78.6**

Table 12.1: Development and test phone accuracies (%) obtained on the TIMIT dataset using different combinations of acoustic and phonetic models.

Acoustic model	HMM	RNN	Hybrid
LR	31.8	32.3	34.4
RNN	40.5	43.8	44.7
DNN	70.0	72.7	**73.7**

Table 12.2: Development phone accuracies (%) obtained on the Switchboard dataset using different combinations of acoustic and phonetic models.

baseline. Note that the improvements obtained with CRF full-sequence training are typically more modest in this context (Mohamed et al., 2010), suggesting that Markovian assumptions in linear-chain CRFs are more limiting than the conditional independence assumption violated by our model as discussed in section 12.3.

It could be argued that the improvements brought by our RNN phonetic model capitalize on higher-level dependencies between phones, and that the inclusion of a word language model would nullify those gains. In the next experiments we verify if our method translates in good word recognition performance on the Switchboard dataset. While a 3-gram language model could be directly integrated into a sophisticated context-dependent decoding procedure, we simply provide a performance benchmark by *rescoring* a list of the N best candidates found by a DNN + HMM system ($N = 100$). The word error rates shown in Table 12.3 clearly demonstrate the superiority of an RNN phonetic model when used in complementarity to a language model.

DNN + HMM	33.0
DNN + RNN	32.7
DNN (Hybrid)	**32.0**
Oracle	19.5
Anti-oracle	56.8

Table 12.3: Test word error rates (%) obtained on the Switchboard dataset using different phonetic models.

12.7 Conclusions

In this paper, we presented a principled way to combine an RNN-based phonetic model with an arbitrary frame-level acoustic classifier. The efficiency of the decoding and alignment procedures now allows to use an RNN whenever an HMM was previously used. Interestingly, phone sequence modeling seems to be an important component of accurate speech recognition, even in the case where strong acoustic classifiers and word language models are already available.

13 Prologue to Sixth Article

13.1 Article Details

Exploiting Long-Term Temporal Dependencies in NMF Using Recurrent Neural Networks with Application to Source Separation
Nicolas Boulanger-Lewandowski, Gautham Mysore and Matthew Hoffman
Published in *Proceedings of the 39th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* in 2014.

13.2 Context

Non-negative matrix factorization (NMF) (Lee and Seung, 1999) is a popular method to separate complex audio mixtures into their individual constituent channels, or *sources*. In addition to the reconstruction criterion of baseline NMF, several constraints can be applied to the non-negative matrix factors W and H, such as sparsity (Cont, 2006), harmonicity (Vincent et al., 2010), temporal continuity (e.g. Virtanen, 2007), or adherence to Kalman filters (e.g. Nam et al., 2012) or Markov models (e.g. Mysore et al., 2010). In this chapter, we aim to improve the temporal description in the latter category with an expressive connectionist model that can describe long-term dependencies and high-level structure in the data. Contrary to the purely discriminative approaches developed in earlier chapters, we will integrate our RNN-RBM symbolic model into an NMF generative model that can reconstruct actual audio signals.

13.3 Contributions

The first contribution of this article is a method to incorporate an RNN-based prior on the activity matrix H inside the NMF decomposition, and a gradient-based algorithm to perform inference. A second contribution is the application of that method to audio source separation and the improvement of benchmarks on the MIR-1K dataset.

Note that the generative model proposed in this chapter (eq. 14.18) is equivalent to the hybrid model previously presented in Section 12.3. The correspondence can be seen by replacing the explicit emissions in the NMF model (eq. 14.2) with the Bayes' rule reformulated emissions in the discriminative model (eq. 12.9). As such, the method in this chapter does not suffer from the label bias problem.

13.4 Recent Developments

This paper will be presented at ICASSP in May 2014.

14 Exploiting long-term temporal dependencies in NMF using recurrent neural networks with application to source separation

This paper seeks to exploit high-level temporal information during feature extraction from audio signals via non-negative matrix factorization. Contrary to existing approaches that impose local temporal constraints, we train powerful recurrent neural network models to capture long-term temporal dependencies and event co-occurrence in the data. This gives our method the ability to "fill in the blanks" in a smart way during feature extraction from complex audio mixtures, an ability very useful for a number of audio applications. We apply these ideas to source separation problems.

14.1 Introduction

Non-negative matrix factorization (NMF) is an unsupervised technique to discover parts-based representations underlying non-negative data (Lee and Seung, 1999). When applied to the magnitude spectrogram of an audio signal, NMF can discover a basis of interpretable recurring events and their associated time-varying encodings, or *activities*, that together optimally reconstruct the original spectrogram. In addition to accurate reconstruction, it is often useful to enforce various constraints to influence the decomposition. Those constraints generally act on each time frame independently to encourage sparsity (Cont, 2006), harmonicity of the basis spectra (Vincent et al., 2010) or relevance with respect to a discriminative criterion (Boulanger-Lewandowski et al., 2012a), or include a temporal component such as simple continuity (Virtanen, 2007; Virtanen et al., 2008; Wilson et al., 2008; Fev́otte, 2011), Kalman filtering like techniques (Nam et al., 2012; Fev́otte et al., 2013; Mohammadiha et al., 2013) or Markov chain modeling (Ozerov et al., 2009; Nakano et al., 2010; Mohammadiha and Leijon, 2013; Mysore et al., 2010). In this paper, we aim to improve the temporal description in the latter category with an expressive connectionist model that can describe long-term dependencies and high-level structure in the data.

Recurrent neural networks (RNN) (Rumelhart et al., 1986a) 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 feature vectors in an activity matrix, 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 has made them very successful in music, language and speech applications (Boulanger-Lewandowski et al., 2012b; Mikolov et al., 2011; Graves et al., 2013; Bengio et al., 2013). A recent extension of the RNN, called the RNN-RBM, employs time-dependent restricted Boltzmann machines (RBM) to describe the multimodal conditional densities typically present in audio signals, resulting in significant improvements over N-gram and HMM baselines (Sutskever et al., 2008; Boulanger-Lewandowski et al., 2012b). In this paper, we show how to integrate RNNs into the NMF framework in order to model sound mixtures. We apply our approach to audio source separation problems, but the technique is general and can be used for various audio applications.

The remainder of the paper is organized as follows. In sections 14.2 and 14.3 we introduce the NMF and RNN models. In section 14.4 we incorporate temporal constraints into the feature extraction algorithm. Finally, we present our methodology and results in sections 14.5 and 14.6.

14.2 Non-negative matrix factorization

The NMF method aims to discover an approximate factorization of an input matrix X:

$$\overset{N\times T}{X} \simeq \overset{N\times T}{\Lambda} \equiv \overset{N\times K}{W} \cdot \overset{K\times T}{H} , \tag{14.1}$$

where X is the observed magnitude spectrogram with time and frequency dimensions T and N respectively, Λ is the reconstructed spectrogram, W is a dictionary matrix of K basis spectra and H is the activity matrix. Non-negativity constraints

$W_{nk} \geq 0, H_{kt} \geq 0$ apply on both matrices. NMF seeks to minimize the *reconstruction error*, a distortion measure between the observed spectrogram X and the reconstruction Λ. A popular choice is the generalized Kullback-Leibler divergence:

$$C_{KL} \equiv \sum_{nt} \left(X_{nt} \log \frac{X_{nt}}{\Lambda_{nt}} - X_{nt} + \Lambda_{nt} \right), \tag{14.2}$$

with which we will demonstrate our method. Minimizing C_{KL} can be achieved by alternating multiplicative updates to H and W (Lee and Seung, 2001):

$$H \leftarrow H \circ \frac{W^T(X/\Lambda)}{W^T 1 1^T} \tag{14.3}$$

$$W \leftarrow W \circ \frac{(X/\Lambda)H^T}{1 1^T H^T}, \tag{14.4}$$

where 1 is a vector of ones, the $\circ$ operator denotes element-wise multiplication, and division is also element-wise. These updates are guaranteed to converge to a stationary point of the reconstruction error.

It is often reasonable to assume that active elements H_{kt} should be limited to a small subset of the available basis spectra. To encourage this behavior, a sparsity penalty $C_S \equiv \lambda |H|$ can be added to the total NMF objective (Hoyer, 2002), where $|\cdot|$ denotes the L_1 norm and λ specifies the relative importance of sparsity. In that context, we impose the constraint that the basis spectra have unit norm. Equation (14.3) becomes:

$$H \leftarrow H \circ \frac{W^T(X/\Lambda)}{1 + \lambda}, \tag{14.5}$$

and the multiplicative update to W (eq. 14.4) is replaced by projected gradient descent (Lin, 2007):

$$W \leftarrow W - \mu(1 - X/\Lambda)H^T \tag{14.6}$$

$$W_{nk} \leftarrow \max(W_{nk}, 0), W_{:k} \leftarrow \frac{W_{:k}}{|W_{:k}|}, \tag{14.7}$$

where $W_{:k}$ is the k-th column of W and μ is the learning rate.

14.3 Recurrent neural networks

The RNN formally defines the distribution of the vector sequence $v \equiv \{v^{(t)} \in \mathbb{R}_0^{+K}, 1 \leq t \leq T\}$ of length T:

$$P(v) = \prod_{t=1}^{T} P(v^{(t)}|\mathcal{A}^{(t)}), \tag{14.8}$$

where $\mathcal{A}^{(t)} \equiv \{v^{(\tau)}|\tau < t\}$ is the sequence history at time t, and $P(v^{(t)}|\mathcal{A}^{(t)})$ is the conditional probability of observing $v^{(t)}$ according to the model, defined below.

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

$$\hat{h}^{(t)} = \sigma(W_{v\hat{h}}v^{(t)} + W_{\hat{h}\hat{h}}\hat{h}^{(t-1)} + b_{\hat{h}}), \tag{14.9}$$

where $\sigma(x) \equiv (1 + e^{-x})^{-1}$ is the element-wise logistic sigmoid function, W_{xy} is the weight matrix tying vectors x, y and b_x is the bias vector associated with x.

The model is trained to predict the observation $v^{(t)}$ at time step t given the previous ones $\mathcal{A}^{(t)}$. The prediction $y^{(t)}$ is obtained from the hidden units at the previous time step $\hat{h}^{(t-1)}$:

$$y^{(t)} = o(W_{\hat{h}v}\hat{h}^{(t-1)} + b_v), \tag{14.10}$$

where $o(a)$ is the output non-linearity function of an activation vector a, and should be as close as possible to the target vector $v^{(t)}$. When the target is a non-negative real-valued vector, the likelihood of an observation can be given by:

$$P(v^{(t)}|\mathcal{A}^{(t)}) \propto \frac{v^{(t)} \cdot y^{(t)}}{|v^{(t)}| \cdot |y^{(t)}|} \tag{14.11}$$

$$o(a)_k = \exp(a_k). \tag{14.12}$$

Other forms for P and o are possible; we have found that the cosine distance combined with an exponential non-linearity works well in practice, presumably because predicting the *orientation* of a vector is much easier for an RNN than predicting its magnitude.[1]

[1] The cosine distance cost (eq. 14.11) is not a proper normalizable distribution for real-valued $v^{(t)}$, but it is only used as a prior in cases where the posterior would be normalizable. More rigorously, it should include a multiplicative term in $\exp(-\lambda|v^{(t)}|^2)$ with $\lambda \ll 1$ that would

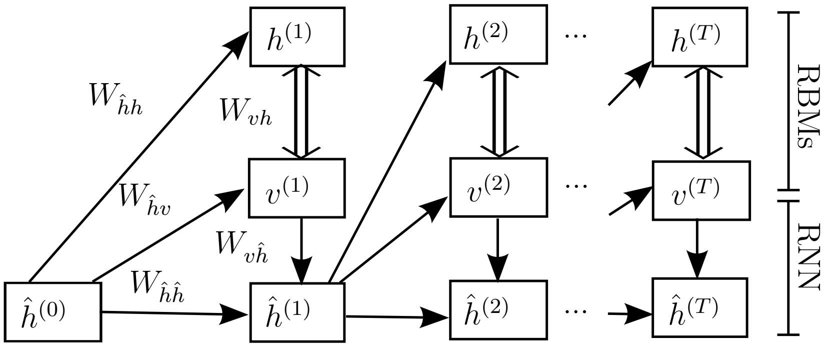


Figure 14.1: Graphical structure of the RNN-RBM. Single arrows represent a deterministic function, double arrows represent the stochastic hidden-visible connections of an RBM. The upper half of the RNN-RBM is the RBM stage while the lower half is a RNN with hidden units $\hat{h}^{(t)}$. The RBM biases $b_h^{(t)}$, $b_v^{(t)}$ are a linear function of $\hat{h}^{(t-1)}$.

When the output observations are multivariate, another approach is to capture the higher-order dependencies between the output variables using a powerful output probability model such as an RBM, resulting in the so-called RNN-RBM (Figure 14.1) (Sutskever et al., 2008; Boulanger-Lewandowski et al., 2012b). The Gaussian RBM variant is typically used to estimate the density of real-valued variables $v^{(t)}$ (Welling et al., 2005). In this case, the RNN's task is to predict the parameters of the conditional distribution, i.e. the RBM biases at time step t:

$$b_v^{(t)} = b_v + W_{\hat{h}v}\hat{h}^{(t-1)} \tag{14.13}$$

$$b_h^{(t)} = b_h + W_{\hat{h}h}\hat{h}^{(t-1)}. \tag{14.14}$$

In an RBM, the likelihood of an observation is related to the free energy $F(v^{(t)})$ by $P(v^{(t)}|\mathcal{A}^{(t)}) \propto e^{-F(v^{(t)})}$:

$$F(v^{(t)}) \equiv \frac{1}{2}||v^{(t)}||^2 - b_v^{(t)} \cdot v^{(t)} - |s(b_h^{(t)} + W_{vh}v^{(t)})|, \tag{14.15}$$

where $s(x) \equiv \log(1 + e^x)$ is the element-wise softplus function and W_{vh} is the weight matrix of the RBM. The log-likelihood gradient with respect to the RBM parameters is generally intractable due to the normalization constant but can be estimated by contrastive divergence (Hinton, 2002; Boulanger-Lewandowski et al., 2012b).

become negligible in the posterior distribution.

The RNN model can be trained by minimizing the negative log-likelihood of the data:

$$C_{RNN}(v) = -\sum_{t=1}^{T} \log P(v^{(t)}|\mathcal{A}^{(t)}), \tag{14.16}$$

whose gradient with respect to the RNN parameters is obtained by backpropagation through time (BPTT) (Rumelhart et al., 1986a). Several strategies can be used to reduce the difficulties associated with gradient-based learning in RNNs including gradient clipping, sparsity and momentum techniques (Bengio et al., 1994, 2013).

14.4 Temporally constrained NMF

In this section, we incorporate RNN regularization into the NMF framework to temporally constrain the activity matrix H during the decomposition. A simple form of regularization that encourages neighboring activity coefficients to be close to each other is temporal smoothing:

$$C_{TS} = \frac{1}{2}\beta \sum_{t=1}^{T-1} ||H_{:t} - H_{:t+1}||^2, \tag{14.17}$$

where the hyperparameter β is a weighting coefficient.

In the proposed model, we add the RNN negative log-likelihood term (eq. 14.16) with $v := \{H_{:t}, 1 \leq t \leq T\}$ to the total NMF cost:

$$C = C_{KL} + C_S + C_{TS} + C_{L2} + \alpha C_{RNN}(H), \tag{14.18}$$

where $C_{L2} \equiv \frac{1}{2}\eta||H||^2$ provides L_2 regularization, and the hyperparameters η, α specify the relative importance of each prior. This framework corresponds to an RNN generative model at temperature α^{-1} describing the evolution of the latent variable $H_{:t}$, the observation $X_{:t}$ at time t being conditioned on $H_{:t}$ via the reconstruction error C_{KL}. The overall graphical model can be seen as a generalization of the non-negative hidden Markov model (N-HMM) (Mysore et al., 2010).

The NMF model is first trained in the usual way by alternating the updates (14.5)–(14.7) and extracting the activity features H; the RNN is then trained to minimize $C_{RNN}(H)$ by stochastic gradient descent. During supervised NMF (Smaragdis et al., 2007), it is necessary to infer the activity matrix H that minimizes the total cost (eq. 14.18) given a pre-trained dictionary W and a test observation X. Our approach is to replace the multiplicative udpate (14.5) with a gradient descent update:

$$H \leftarrow H - \mu\left[W^T(1 - X/\Lambda) + \lambda + \eta H + \tfrac{\partial C_{TS}}{\partial H} + \alpha\tfrac{\partial C_{RNN}}{\partial H}\right] \tag{14.19}$$

where the gradient of C_{TS} is given by:

$$\frac{\partial C_{TS}}{\partial H_{kt}} = \beta\begin{cases} H_{kt} - H_{k(t+1)} & \text{if } t = 1 \\ 2H_{kt} - H_{k(t-1)} - H_{k(t+1)} & \text{if } 1 < t < T \\ H_{kt} - H_{k(t-1)} & \text{if } t = T. \end{cases} \tag{14.20}$$

When deriving $\partial C_{RNN}/\partial H$, it is important to note that $H_{:t}$ affects the cost directly by matching the prediction $y^{(t)}$ in equation (14.11), and also indirectly by influencing the future predictions of the RNN via $\mathcal{A}^{(t+\delta t)}$. By fully backpropagating the gradient through time, we effectively take into account future observations $X_{:(t+\delta t)}$ when updating $H_{:t}$. While other existing approaches require sophisticated inference procedures (Boulanger-Lewandowski et al., 2013b,a), the search for a *globally* optimal H can be facilitated by using gradient descent when the inferred variables are real-valued.

The RNN-RBM requires a different approach due to the intractable partition function of the t^{th} RBM that varies with $\mathcal{A}^{(t)}$. The retained strategy is to consider $\mathcal{A}^{(t)}$ fixed during inference and to approximate the gradient of the cost by:

$$\frac{C_{RNN}}{\partial v^{(t)}} \simeq \frac{\partial F(v^{(t)})}{\partial v^{(t)}} = v^{(t)} - b_v^{(t)} - \sigma(b_h^{(t)} + W_{vh}v^{(t)})W_{vh}^T. \tag{14.21}$$

Since this approach can be unstable, we only update the value of $\mathcal{A}^{(t)}$ every m iterations of gradient descent ($m = 10$) and we use an RNN in conjunction with the RNN-RBM to exploit its tractability and norm independence properties.

14.5 Evaluation

In the next section, we evaluate the performance of our RNN model on a source separation task in comparison with a traditional NMF baseline and NMF with temporal smoothing. Source separation is interesting for our architecture because, contrary to purely discriminative tasks such as multiple pitch estimation or chord estimation where RNNs are known to outperform other models (Boulanger-Lewandowski et al., 2013b,a), source separation requires accurate signal reconstruction.

We consider the supervised and semi-supervised NMF algorithms (Smaragdis et al., 2007) that consist in training submodels on isolated sources before concatenating the pre-trained dictionaries and feeding the relevant activity coefficients into the associated temporal model; final source estimates are obtained by separately reconstructing the part of the observation explained by each submodel. In the semi-supervised setting, an additional dictionary is trained from scratch for each new examined sequence and no temporal model is used for the unsupervised channel. Wiener filtering is used as a final step to ensure that the estimated source spectrograms $X^{(i)}$ add up to the original mixture X:

$$\hat{X}^{(i)} = \frac{X^{(i)}}{\sum_j X^{(j)}} \circ X, \tag{14.22}$$

before transforming each source in the time domain via the inverse short-term Fourier transform (STFT).

Our main experiments are carried out on the MIR-1K dataset[2] featuring 19 singers performing a total of 1,000 Chinese pop karaoke song excerpts, ranging from 4 to 13 seconds and recorded at 16 kHz. For each singer, the available tracks are randomly split into training, validation and test sets in a 8:1:1 ratio. The accompaniment music and singing voice channels are summed directly at their original loudness ($\sim$ 0 dB). The magnitude spectrogram X is computed by the STFT using a 64 ms sliding Blackman window with hop size 30 ms and zero-padded to produce a feature vector of length 900 at each time step. The source separation quality is evaluated with the BSS Eval toolbox[3] using the standard

[2] https://sites.google.com/site/unvoicedsoundseparation/mir-1k

[3] http://bass-db.gforge.inria.fr/bss_eval/

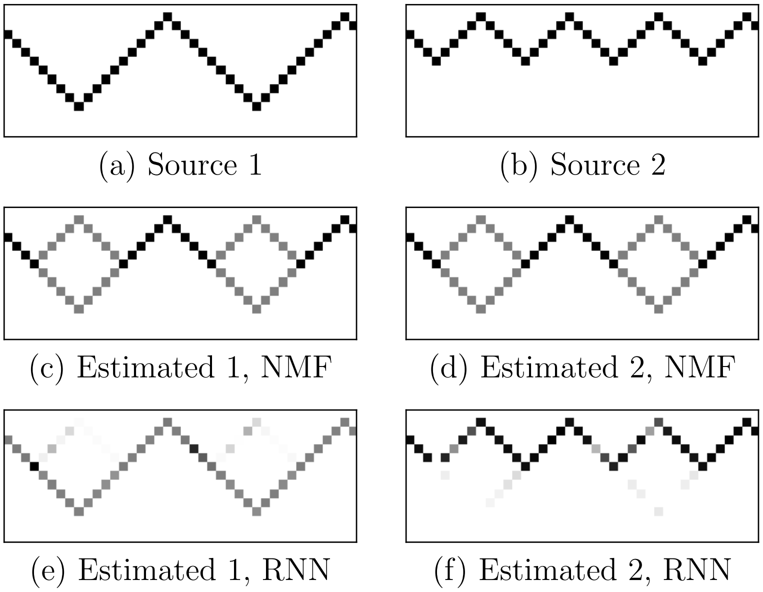
metrics SDR, SIR and SAR that measure for each channel the ratios of source to distortion, interference and artifacts respectively (Vincent et al., 2006). For each model and singer combination, we use a random search on predefined intervals to select the hyperparameters that maximize the mean SDR on the validation set; final performance is reported on the test set.

14.6 Results

To illustrate the effectiveness of our temporally constrained model, we first perform source separation experiments on a synthetic dataset of two sawtooth wave sources of different amplitudes and randomly shifted along both dimensions. Figure 14.2 shows an example of such sources (Fig. 14.2(a–b)), along with the sources estimated by supervised NMF with either no temporal constraint (Fig. 14.2(c–d)) or with an RNN with the cosine distance cost (Fig. 14.2(e–f)). While this problem is provably unsolvable for NMF alone or with simple temporal smoothing (eq. 14.17), the RNN-constrained model successfully separates the two mixed sources. This extreme example demonstrates that temporal constraints become crucial when the *content* of each time frame is not sufficient to distinguish each source.

Source separation results on the MIR-1K dataset are presented in Table 14.1 for supervised (top) and semi-supervised[4] (bottom) NMF ($K = 15$). The RNN-based models clearly outperform the baselines in SDR and SIR for both sources with a moderate degradation in SAR. To illustrate the trade-off between the suppression of the unwanted source and the reduction of artifacts, we plot in Figure 14.3 the performance metrics as a function of the weight α/α_0 of the RNN-RBM model, where $\alpha_0 \in [10, 20]$ is the hyperparameter value selected on the validation set. This inherent trade-off was also observed elsewhere (Mysore et al., 2010). Overall, the observed improvement in SDR is indicative of a better separation quality.

[4] Only the singing voice channel is supervised in this case.


(a) Source 1

(b) Source 2

(c) Estimated 1, NMF

(d) Estimated 2, NMF

(e) Estimated 1, RNN

(f) Estimated 2, RNN

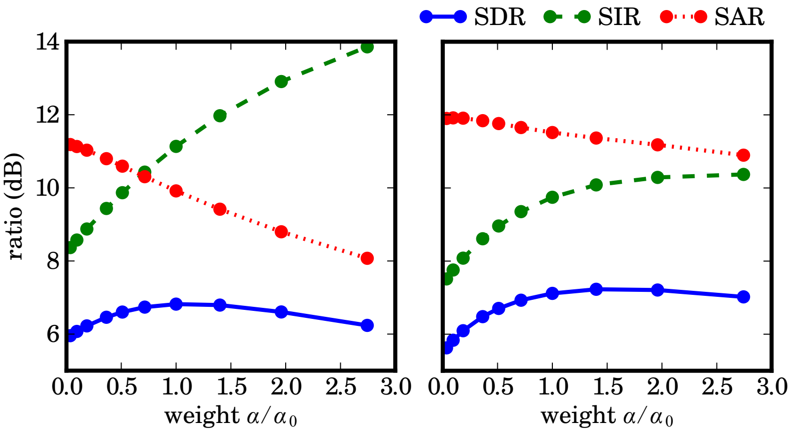
Figure 14.2: Toy example: separation of sawtooth wave sources of different amplitudes (a–b) using supervised NMF with either no prior (c–d) or an RNN with the cosine distance cost (e–f).

14.7 Conclusion

We have presented a framework to leverage high-level information during feature extraction by incorporating an RNN-based prior inside the NMF decomposition. While the combined approach surpasses the baselines in realistic audio source separation settings, it could be further improved by employing a deep bidirectional RNN with multiplicative gates (Graves et al., 2013), replacing the Gaussian RBMs with the recently developed tractable distribution estimator for real-valued vectors RNADE (Uría et al., 2013; Boulanger-Lewandowski et al., 2012b), implementing an EM-like algorithm to jointly train the NMF and RNN models, and transitioning to a universal speech model for singer-independent source separation (Sun and Mysore, 2013).

Model	SDR		SIR		SAR	
	acc.	sing.	acc.	sing.	acc.	sing.
NMF	5.04	5.05	7.75	7.59	10.00	10.25
NMF-sm	6.08	5.59	8.77	7.42	**10.96**	**11.93**
RNN	6.13	5.80	9.46	7.79	10.34	11.52
RNN-RBM	**6.83**	**7.12**	**11.25**	**9.75**	9.86	11.52
NMF	5.20	3.58	9.54	4.95	8.80	11.43
NMF-sm	5.57	3.71	9.48	4.94	**9.57**	11.84
RNN	5.94	3.70	10.49	4.86	9.36	**12.07**
RNN-RBM	**6.16**	**5.05**	**11.81**	**7.12**	9.04	10.59

Table 14.1: Audio source separation performance on the MIR-1K test set obtained via singer-dependent supervised (top) and semi-supervised (bottom) NMF with either no prior, simple temporal smoothing, an RNN (eq. 14.11) or the RNN-RBM.



(a) Accompaniment (b) Singing voice

Figure 14.3: Source separation performance trade-off on the MIR-1K test set by supervised NMF with an RNN-RBM model weighted by α, where α_0 maximizes the validation SDR.

15 Conclusions

15.1 Summary of contributions

In this thesis, we proposed, analyzed and applied several RNN-based models of high-dimensional sequences especially well suited for polyphonic music and speech. Our contributions can be summarized as follows:

1. We introduced the RNN-RBM (NADE) (Section 4.4) to model high-dimensional sequences, derived an efficient training procedure based on contrastive divergence, and demonstrated the use of pre-training techniques and of Hessian-free optimization;
2. We reviewed the issues giving rise to the optimization difficulties in RNNs and several techniques that have been proposed to reduce them (Chapter 6); we proposed a simplified formulation of Nesterov momentum and offered an alternative interpretation of the method (Section 6.3.5);
3. We proposed different ways to combine a generalized language model with an acoustic classifier in:
   (a) A product of experts (Section 4.7), applicable to any acoustic model that can estimate $P(z^{(t)}|x^{(t)})$,
   (b) An input/output architecture (Section 8.2.3) that expresses $P(z|x)$ under a single training objective,
   (c) A generative hybrid model (Section 12.3) that replaces the HMM with a more realistic symbolic model trainable via EM,
   (d) A generative NMF-based model (Section 14.4) that allows to reconstruct the observations x;
4. We derived inference algorithms for RNNs that present speed and accuracy improvements over greedy chronological search and beam search:

   (a) High-dimensional beam search, optimized for either multimodal (Algorithm 8.1) or factored (Algorithm 8.2) output distributions,
   (b) Pruned beam search (Algorithm 10.2), a dynamic programming-like pruning technique for beam search inspired by Viterbi decoding,
   (c) Fast alignment (Section 12.5), an efficient two-pass algorithm to search for the optimal alignment of a given sequence,
   (d) Gradient descent inference (eq. 14.19; 14.21), a technique applicable for real-valued vectors $z^{(t)}$;

5. We demonstrated applications of our approach and (in some cases) improved the state of the art in the following areas:
   (a) Polyphonic music generation (Section 4.6), with substantial improvement over the RTRBM and other popular models,
   (b) Polyphonic music transcription (Section 8.4), with improvement of the state of the art (Table 8.2) and of robustness to noise (Figure 8.2),
   (c) Audio chord recognition (Section 10.5), with accuracies competitive with the state of the art on the MIREX dataset,
   (d) Phone sequence modeling (Section 12.6), shown to be an important component of speech recognition even in the presence of powerful acoustic and natural language models,
   (e) Audio source separation (Section 14.6), for which we have shown clear improvements in the supervised and semi-supervised NMF baselines.

15.2 Future directions

An interesting avenue to improve the dynamic programming-like inference algorithm would be to consider various *hash functions* that implicitly determine if different sequences are sufficiently similar to be subject to pruning, and that generalize the extreme cases of considering only the last emitted symbol or the full sequence as presented here. Possible hash functions include the n previously emitted symbols, the n previously emitted unaligned symbols, and a vector-quantized version of the RNN state. A further generalization would be to maintain at most a fixed number of candidates per unique hash value, that could be greater than 1.

Given the success of gradient descent inference to quickly find local maxima of $P(z|x)$ for real-valued outputs, it may prove fruitful to adapt this method to binary vectors, e.g. by approaching z^* via gradient descent on an interpolation of the density in $\mathbb{R}^{NT}$. Such an approach would also enable us to explicitly optimize our deterministic test-time predictions z^* with respect to a relevant cost function during training by backpropagating through the inference procedure.

Training RNNs, especially to discover long-term complex dependencies, obviously remains an important area of research. We believe that stochastic methods (e.g. Bayer et al., 2014) and active learning will become prevalent in the future. Given the difficulty to learn a sense of meter and rhythm by our symbolic model, it may also be instructive to provide "metronome" intermediate targets and self-pacing mechanisms to gain insight into the learning of periodical patterns.

In speech recognition, the rudimentary approach of rescoring the top predictions of an N-gram word language model as in Section 12.6 could be replaced by an end-to-end system that directly performs speech to text. Another avenue would be to learn a distributed representation of context-dependent phones with the RNN-RBM by using tuples of phonemes as a target $z^{(t)} \leftarrow (\dots, z^{(t-1)}, z^{(t)}, z^{(t+1)}, \dots)$ at each time step and concatenated softmax RBMs as conditional distribution estimators.

Another potential area of investigation is to substitute RNNs with deep LSTM networks (Graves et al., 2013) and conditional Gaussian RBMs with RNADEs (Uría et al., 2013) in the models presented in this thesis.

References

Abdallah, S. and M. Plumbley (2006). Unsupervised analysis of polyphonic music by sparse coding. *IEEE Trans. on Neural Networks 17*(1), 179–196. (Cited on pages 6, 26 and 28.)

Abe, M. and J. O. Smith (2005). AM/FM rate estimation for time-varying sinusoidal modeling. In *ICASSP*, Volume 3, pp. 201–204. (Cited on page 6.)

Aljanaki, A. (2011). Automatic musical key detection. Master's thesis, University of Tartu. (Cited on page 4.)

Allan, M. and C. Williams (2005). Harmonising chorales by probabilistic inference. In *NIPS 17*, pp. 25–32. (Cited on pages 47, 48 and 49.)

Baker, J., L. Deng, J. Glass, S. Khudanpur, C. Lee, N. Morgan, and D. O'Shaughnessy (2009). Developments and directions in speech recognition and understanding. *IEEE Signal Processing Magazine 26*(3), 75–80. (Cited on pages 7 and 102.)

Bastien, F., P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley, and Y. Bengio (2012). Theano: new features and speed improvements. In *Deep Learning and Unsupervised Feature Learning NIPS Workshop*. (Cited on page 30.)

Bay, M., A. Ehmann, and J. Downie (2009). Evaluation of multiple-F0 estimation and tracking systems. In *ISMIR*. (Cited on pages 5, 38, 48, 72 and 79.)

Bayer, J., C. Osendorfer, N. Chen, S. Urban, and P. van der Smagt (2014). On fast dropout and its applicability to recurrent networks. In *ICLR*. (Cited on pages 35 and 129.)

Benaroya, L., F. Bimbot, and R. Gribonval (2006, January). Audio source separation with a single sensor. *IEEE Transactions on Audio, Speech, and Language Processing 14*(1). (Cited on page 8.)

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning 2*(1), 1–127. (Cited on pages 29, 44, 54, 58, 62, 79, 86 and 89.)

Bengio, Y., F. Bastien, A. Bergeron, N. Boulanger-Lewandowski, T. Breuel, Y. Chherawala, M. Cisse, M. Côté, D. Erhan, J. Eustache, et al. (2011). Deep learners benefit more from out-of-distribution examples. In *AISTATS*. (Cited on page 30.)

Bengio, Y. and S. Bengio (2000). Modeling high-dimensional discrete data with multi-layer neural networks. In *NIPS 12*, pp. 400–406. (Cited on pages 13, 40 and 71.)

Bengio, Y., N. Boulanger-Lewandowski, and R. Pascanu (2013). Advances in optimizing recurrent networks. In *ICASSP 38*. (Cited on pages 92, 103, 105, 117 and 121.)

Bengio, Y., A. Courville, and P. Vincent (2012). Representation learning: A review and new perspectives. Technical report, arXiv:1206.5538. (Cited on page 58.)

Bengio, Y. and P. Frasconi (1996). Input-output HMMs for sequence processing. *IEEE Transactions on Neural Networks 7*(5), 1231–1249. (Cited on page 17.)

Bengio, Y., P. Lamblin, D. Popovici, and H. Larochelle (2006). Greedy layer-wise training of deep networks. In *NIPS*. (Cited on page 58.)

Bengio, Y., P. Simard, and P. Frasconi (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. on Neural Networks 5*(2), 157–166. (Cited on pages 1, 21, 33, 36, 54, 56, 57, 59, 92, 105 and 121.)

Bergstra, J. and Y. Bengio (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research 13*, 281–305. (Cited on pages 65 and 79.)

Bergstra, J., O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio (2010). Theano: a CPU and GPU math expression compiler. In *SciPy*. (Cited on pages 30 and 32.)

Bergstra, J., N. Casagrande, D. Erhan, D. Eck, and B. Kégl (2006). Aggregate features and adaboost for music classification. *Machine Learning 65*(2-3), 473–484. (Cited on pages 31, 84, 87 and 90.)

Böck, S. and M. Schedl (2012). Polyphonic piano note transcription with recurrent neural networks. In *ICASSP*, pp. 121–124. (Cited on pages 69, 72, 81 and 87.)

Boogaart, C. and R. Lienhart (2009). Note onset detection for the transcription of polyphonic piano music. In *ICME*, pp. 446–449. (Cited on page 81.)

Boulanger-Lewandowski, N., Y. Bengio, and P. Vincent (2012a). Discriminative non-negative matrix factorization for multiple pitch estimation. In *ISMIR*, pp. 205–210. (Cited on pages 8, 26, 29 and 116.)

Boulanger-Lewandowski, N., Y. Bengio, and P. Vincent (2012b). Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *ICML 29*. (Cited on pages 61, 64, 72, 74, 79, 80, 87, 103, 117, 120 and 125.)

Boulanger-Lewandowski, N., Y. Bengio, and P. Vincent (2013a). Audio chord recognition with recurrent neural networks. In *ISMIR*. (Cited on pages 103, 108, 109, 122 and 123.)

Boulanger-Lewandowski, N., Y. Bengio, and P. Vincent (2013b). High-dimensional sequence transduction. In *ICASSP*. (Cited on pages 86, 87, 92, 94, 103, 122 and 123.)

Boulanger-Lewandowski, N., J. Droppo, M. Seltzer, and D. Yu (2014). Phone sequence modeling with recurrent neural networks. In *ICASSP 39*. (Not cited.)

Boulanger-Lewandowski, N., G. Mysore, and M. Hoffman (2014). Exploiting long-term temporal dependencies in NMF using recurrent neural networks with application to source separation. In *ICASSP 39*. (Not cited.)

Brakel, P., D. Stroobandt, and B. Schrauwen (2013). Training energy-based models for time-series imputation. *Journal of Machine Learning Research 14*(1), 2771–2797. (Cited on page 34.)

Brown, P. (1987). *The acoustic-modeling problem in automatic speech recognition.* Ph. D. thesis, Carnegie-Mellon University. (Cited on pages 18, 97 and 106.)

Cardoso, J. (1998). Blind signal separation: Statistical principles. *Proceedings of the IEEE 9*(10), 2009–2025. (Cited on page 9.)

Cemgil, A. (2004). *Bayesian music transcription.* Ph. D. thesis, Radboud University Nijmegen. (Cited on pages 3, 4, 34 and 38.)

Cemgil, A., H. Kappen, and D. Barber (2006). A generative model for music transcription. *IEEE Transactions on Audio, Speech, and Language Processing 14*(2), 679–694. (Cited on pages 3, 69 and 72.)

Chen, R., W. Shen, A. Srinivasamurthy, and P. Chordia (2012). Chord recognition using duration-explicit hidden Markov models. In *ISMIR*. (Cited on pages 87, 97 and 98.)

Conklin, D. (2003). Music generation from statistical models. In *AISB Symposium on Artificial Intelligence and Creativity in the Arts and Sciences*, pp. 30–35. (Cited on page 16.)

Cont, A. (2006). Realtime multiple pitch observation using sparse non-negative constraints. In *ISMIR*. (Cited on pages 6, 8, 26, 29, 114 and 116.)

Dahl, G., T. Sainath, and G. Hinton (2013). Improving deep neural networks for LVCSR using rectified linear units and dropout. In *ICASSP*. (Cited on pages 3, 8, 101 and 103.)

Dahl, G., D. Yu, L. Deng, and A. Acero (2012). Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing 20*(1), 30–42. (Cited on pages 8, 31, 87, 100, 103 and 106.)

Davies, M. (2007). *Towards automatic rhythmic accompaniment.* Ph. D. thesis, University of London. (Cited on page 4.)

de Haas, B., J. Magalhães, and F. Wiering (2012). Improving audio chord transcription by exploiting harmonic and metric knowledge. In *ISMIR*, pp. 295–300. (Cited on page 3.)

Dessein, A., A. Cont, and G. Lemaitre (2010). Real-time polyphonic music transcription with non-negative matrix factorization and beta-divergence. In *ISMIR*. (Cited on pages 6, 26 and 29.)

Eck, D. and J. Lapalme (2008). Learning musical structure directly from sequences of music. Technical report, Université de Montréal. (Cited on page 23.)

Eck, D. and J. Schmidhuber (2002). Finding temporal structure in music: Blues improvisation with LSTM recurrent networks. In *NNSP*, pp. 747–756. (Cited on pages 4, 23, 34, 38 and 87.)

El Hihi, S. and Y. Bengio (1996). Hierarchical recurrent neural networks for long-term dependencies. In *NIPS 8*, pp. 493–499. (Cited on pages 22, 23, 54 and 60.)

Erhan, D., Y. Bengio, A. Courville, P. Manzagol, P. Vincent, and S. Bengio (2010). Why does unsupervised pre-training help deep learning? *JMLR 11*, 625–660. (Cited on page 58.)

Erhan, D., P. Manzagol, Y. Bengio, S. Bengio, and P. Vincent (2009). The difficulty of training deep architectures and the effect of unsupervised pre-training. In *AISTATS*, pp. 153–160. (Cited on page 30.)

Fev́otte, C. (2011). Majorization-minimization algorithm for smooth Itakura-Saito nonnegative matrix factorization. In *ICASSP*. (Cited on page 116.)

Fev́otte, C., J. L. Roux, and J. R. Hershey (2013). Non-negative dynamical system with application to speech and audio. In *ICASSP*. (Cited on page 116.)

Fitzgerald, D., M. Cranitch, and E. Coyle (2005). Generalised prior subspace analysis for polyphonic pitch transcription. In *DAFX 8*. (Cited on page 26.)

Franklin, J. (2006). Recurrent neural networks for music computation. *INFORMS Journal on Computing 18*(3), 321–338. (Cited on page 23.)

Fujishima, T. (1999). Realtime chord recognition of musical sound: A system using Common Lisp Music. In *ICMC*, pp. 464–467. (Cited on page 7.)

Glorot, X., A. Bordes, and Y. Bengio (2011a). Deep sparse rectifier neural networks. In *AISTATS*. (Cited on pages 20, 30 and 62.)

Glorot, X., A. Bordes, and Y. Bengio (2011b). Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*. (Cited on page 30.)

Goodfellow, I., M. Mirza, X. Da, A. Courville, and Y. Bengio (2014). An empirical investigation of catastrophic forgetting in gradient-based neural networks. In *ICLR*. (Cited on page 6.)

Graves, A. (2012). Sequence transduction with recurrent neural networks. In *ICML 29*. (Cited on pages 69, 71, 72, 86, 92, 94, 100, 103 and 106.)

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*. (Cited on page 2.)

Graves, A., S. Fernández, F. Gomez, and J. Schmidhuber (2006). Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *ICML 23*, pp. 369–376. (Cited on page 106.)

Graves, A., A. Mohamed, and G. Hinton (2013). Speech recognition with deep recurrent neural networks. In *ICASSP*. (Cited on pages 8, 100, 103, 106, 107, 117, 125 and 129.)

Gülçehre, C. and Y. Bengio (2013). Knowledge matters: Importance of prior information for optimization. In *ICLR*. (Cited on pages 84, 87 and 89.)

Hamel, P., Y. Bengio, and D. Eck (2012). Building musically-relevant audio features through multiple timescale representations. In *ISMIR*. (Cited on pages 31, 87 and 90.)

Hamel, P., M. Davies, K. Yoshii, and M. Goto (2013). Transfer learning in MIR: Sharing learned latent representations for music audio classification and similarity. In *ISMIR*. (Cited on page 30.)

Hamel, P. and D. Eck (2010). Learning features from music audio with deep belief networks. In *ISMIR*, pp. 339–344. (Cited on pages 30, 84 and 87.)

Harte, C. (2010). *Towards automatic extraction of harmony information from music signals.* Ph. D. thesis, University of London. (Cited on pages 6, 7, 84, 86 and 96.)

Hermans, M. and B. Schrauwen (2013). Training and analysing deep recurrent neural networks. In *NIPS*, pp. 190–198. (Cited on page 23.)

Hinton, G. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation 14*(8), 1771–1800. (Cited on pages 12, 39, 73 and 120.)

Hinton, G., L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury (2012). Deep neural networks for acoustic modeling in speech recognition. *Signal Processing Magazine 29*(6), 82–97. (Cited on pages 30, 84, 87, 93 and 105.)

Hinton, G., S. Osindero, and Y. Teh (2006). A fast learning algorithm for deep belief nets. *Neural computation 18*(7), 1527–1554. (Cited on pages 30, 58 and 89.)

Hinton, G., N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*. (Cited on page 30.)

Hochreiter, S. (1991). Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, T.U. Münich. (Cited on pages 54, 56 and 57.)

Hochreiter, S. and J. Schmidhuber (1997). Long short-term memory. *Neural Computation 9*(8), 1735–1780. (Cited on pages 21, 32, 54, 60 and 72.)

Hoyer, P. (2002). Non-negative sparse coding. In *Neural Networks for Signal Processing*, pp. 557–565. IEEE. (Cited on pages 27 and 118.)

Hsu, E., K. Pulli, and J. Popović (2005). Style translation for human motion. In *SIGGRAPH*, pp. 1082–1089. (Cited on page 47.)

Humphrey, E. and J. Bello (2012). Rethinking automatic chord recognition with convolutional neural networks. In *ICMLA 11*, Volume 2, pp. 357–362. (Cited on pages 30, 84 and 86.)

Jaeger, H., M. Lukosevicius, D. Popovici, and U. Siewert (2007). Optimization and applications of echo state networks with leaky- integrator neurons. *Neural Networks 20*(3), 335–352. (Cited on pages 23, 54 and 60.)

Khadkevich, M. and M. Omologo (2011). Time-frequency reassigned features for automatic chord recognition. In *ICASSP*, pp. 181–184. IEEE. (Cited on page 98.)

Kompass, R. (2007). A generalized divergence measure for nonnegative matrix factorization. *Neural computation 19*(3), 780–791. (Cited on page 26.)

Krizhevsky, A., I. Sutskever, and G. Hinton (2012). ImageNet classification with deep convolutional neural networks. In *NIPS*. (Cited on page 62.)

Krumhansl, C. and E. Kessler (1982). Tracing the dynamic changes in perceived tonal organization in a spatial representation of musical keys. *Psychological Review 89*(4), 334. (Cited on page 4.)

Lafferty, J., A. McCallum, and F. Pereira (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML 18*, pp. 282–289. (Cited on pages 19, 100, 101 and 103.)

Larochelle, H. and I. Murray (2011). The neural autoregressive distribution estimator. *JMLR: W&CP 15*, 29–37. (Cited on pages 13, 39, 40, 61, 71 and 73.)

Lavrenko, V. and J. Pickens (2003). Polyphonic music modeling with random fields. In *ACM MM*, pp. 120–129. (Cited on pages 19, 48 and 49.)

Le, Q., M. Ranzato, R. Monga, M. Devin, G. Corrado, K. Chen, J. Dean, and A. Ng (2012). Building high-level features using large scale unsupervised learning. In *ICML*. (Cited on page 58.)

LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*(11), 2278–2324. (Cited on page 20.)

Lee, C., Y. Yang, K. Lin, and H. Chen (2010). Multiple fundamental frequency estimation of piano signals via sparse representation of Fourier coefficients. In *ISMIR*. (Cited on page 6.)

Lee, D. and H. Seung (1999). Learning the parts of objects by non-negative matrix factorization. *Nature 401*(6755), 788–791. (Cited on pages 8, 25, 114 and 116.)

Lee, D. D. and H. S. Seung (2001). Algorithms for non-negative matrix factorization. In *NIPS 13*. (Cited on pages 27 and 118.)

Li, Y. and D. Wang (2007). Pitch detection in polyphonic music using instrument tone models. *ICASSP 2*, 481. (Cited on pages 5 and 38.)

Lin, C. J. (2007). Projected gradient methods for nonnegative matrix factorization. *Neural computation 19*(10), 2756–2779. (Cited on pages 28 and 118.)

Lin, T., B. Horne, P. Tino, and C. Giles (1995). Learning long-term dependencies is not as difficult with NARX recurrent neural networks. Technical Report UMICAS-TR-95-78, U. Mariland. (Cited on page 60.)

Mallat, S. and Z. Zhang (1993). Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing 41*(12), 3397–3415. (Cited on page 26.)

Marolt, M. (2004). A connectionist approach to automatic transcription of polyphonic piano music. *IEEE Transactions on Multimedia 6*(3), 439–449. (Cited on pages 6 and 81.)

Martens, J. (2010). Deep learning via Hessian-free optimization. In *ICML*, pp. 735–742. (Cited on pages 32 and 58.)

Martens, J. and I. Sutskever (2011). Learning recurrent neural networks with Hessian-free optimization. In *ICML 28*. (Cited on pages 23, 32, 33, 36, 50, 54, 57, 58, 59, 60, 61 and 72.)

Mauch, M. (2010). *Automatic chord transcription from audio using computational models of musical context.* Ph. D. thesis, University of London. (Cited on pages 6, 7, 84, 86 and 96.)

Mauch, M. and S. Dixon (2010). Approximate note transcription for the improved identification of difficult chords. In *ISMIR*, pp. 135–140. (Cited on pages 7, 96 and 98.)

McCallum, A., D. Freitag, and F. Pereira (2000). Maximum entropy Markov models for information extraction and segmentation. In *ICML 17*, pp. 591–598. (Cited on pages 18, 100 and 103.)

Mikolov, T. (2012). *Statistical Language Models based on Neural Networks*. Ph. D. thesis, Brno University of Technology. (Cited on pages 54 and 60.)

Mikolov, T., A. Deoras, S. Kombrink, L. Burget, and J. Cernocký (2011). Empirical evaluation and combination of advanced language modeling techniques. In *INTERSPEECH*, pp. 605–608. (Cited on pages 103 and 117.)

Mikolov, T., S. Kombrink, L. Burget, J. Cernocky, and S. Khudanpur (2011). Extensions of recurrent neural network language model. In *ICASSP 36*. (Cited on pages 54, 57, 65 and 66.)

Mikolov, T. and G. Zweig (2012). Context dependent recurrent neural network language model. In *Workshop on Spoken Language Technology*. (Cited on page 61.)

Mnih, V., H. Larochelle, and G. Hinton (2011). Conditional restricted Boltzmann machines for structured output prediction. In *UAI 27*. (Cited on page 50.)

Mohamed, A., G. Dahl, and G. Hinton (2009). Deep belief networks for phone recognition. In *NIPS Workshop*. (Cited on page 30.)

Mohamed, A., D. Yu, and L. Deng (2010). Investigation of full-sequence training of deep belief networks for speech recognition. In *INTERSPEECH*, pp. 2846–2849. (Cited on pages 20 and 112.)

Mohammadiha, N. and A. Leijon (2013). Nonnegative HMM for babble noise derived from speech HMM: Application to speech enhancement. *IEEE Trans. on Acoustics, Speech, and Lang. Proc. 21*(5), 998–1011. (Cited on pages 9 and 116.)

Mohammadiha, N., P. Smaragdis, and A. Leijon (2013). Prediction based filtering and smoothing to exploit temporal dependencies in NMF. In *ICASSP*. (Cited on page 116.)

Mozer, M. C. (1994). Neural network music composition by prediction. *Connection Science 6*(2), 247–280. (Cited on pages 4, 23, 33, 38 and 87.)

Murphy, K. (2002). *Dynamic Bayesian Networks: Representation, Inference and Learning.* Ph. D. thesis, UC Berkeley, Computer Science Division. (Cited on page 17.)

Mysore, G., P. Smaragdis, and B. Raj (2010). Non-negative hidden Markov modeling of audio with application to source separation. In *LVA/ICA*, pp. 140–148. (Cited on pages 9, 114, 116, 121 and 124.)

Nair, V. and G. Hinton (2010). Rectified linear units improve restricted Boltzmann machines. In *ICML.* (Cited on pages 20, 30 and 62.)

Nakano, M., J. Le Roux, H. Kameoka, Y. Kitano, N. Ono, and S. Sagayama (2010). Nonnegative matrix factorization with Markov-chained bases for modeling time-varying patterns in music spectrograms. In *LVA/ICA*. (Cited on pages 9 and 116.)

Nam, J., G. J. Mysore, and P. Smaragdis (2012). Sound recognition in mixtures. In *LVA/ICA*. (Cited on pages 9, 114 and 116.)

Nam, J., J. Ngiam, H. Lee, and M. Slaney (2011). A classification-based polyphonic piano transcription approach using learned feature representations. In *ISMIR*. (Cited on pages 3, 30, 34, 38, 51, 52, 53, 79, 80, 81, 84 and 87.)

Nesterov, Y. (1983). A method for unconstrained convex minimization problem with the rate of convergence $o(1/k^2)$. *Doklady AN SSSR (translated as Soviet. Math. Docl.) 269*, 543–547. (Cited on pages 54 and 62.)

Ni, Y., M. McVicar, R. Santos-Rodríguez, and T. De Bie (2012). An end-to-end machine learning system for harmonic analysis of music. *Audio, Speech, and Language Processing 20*(6), 1771–1783. (Cited on pages 7, 96, 97 and 98.)

Ozerov, A., C. Févotte, and M. Charbit (2009). Factorial scaled hidden markov model for polyphonic audio representation and source separation. In *WASPAA*. (Cited on pages 9 and 116.)

Paiement, J., S. Bengio, and D. Eck (2009). Probabilistic models for melodic prediction. *Artificial Intelligence 173*(14), 1266–1274. (Cited on pages 4, 17, 34 and 38.)

Paiement, J., Y. Grandvalet, S. Bengio, and D. Eck (2007). A generative model for rhythms. In *NIPS*. (Cited on page 16.)

Paiement, J.-F., D. Eck, S. Bengio, and D. Barber (2005). A graphical model for chord progressions embedded in a psychoacoustic space. In *ICML 22*, pp. 641–648. (Cited on page 23.)

Palomäki, K., G. Brown, and J. Barker (2004). Techniques for handling convolutional distortion with 'missing data' automatic speech recognition. *Speech Communication 43*(1), 123–142. (Cited on page 81.)

Pascanu, R., C. Gulcehre, K. Cho, and Y. Bengio (2014). How to construct deep recurrent neural networks. In *ICLR*. (Cited on page 35.)

Pascanu, R., T. Mikolov, and Y. Bengio (2012). Understanding the exploding gradient problem. *arXiv preprint arXiv:1211.5063*. (Cited on pages 35, 54, 59 and 60.)

Pascanu, R., T. Mikolov, and Y. Bengio (2013). On the difficulty of training recurrent neural networks. *JMLR W&CP 28*(3), 1310–1318. (Cited on pages 35 and 54.)

Pearlmutter, B. (1994). Fast exact multiplication by the Hessian. *Neural Computation 6*(1), 147–160. (Cited on page 32.)

Pickens, J. (2000). A comparison of language modeling and probabilistic text information retrieval approaches to monophonic music retrieval. In *ISMIR*, pp. 36. (Cited on page 16.)

Pickens, J., J. Bello, G. Monti, M. Sandler, T. Crawford, M. Dovey, and D. Byrd (2002). Polyphonic score retrieval using polyphonic audio queries: A harmonic modeling approach. In *ISMIR*, pp. 140–149. (Cited on page 16.)

Plumbley, M., S. Abdallah, T. Blumensath, and M. Davies (2006). Sparse representations of polyphonic music. *Signal Processing 86*(3), 417–431. (Cited on page 6.)

Poliner, G. and D. Ellis (2005). A classification approach to melody transcription. In *ISMIR*, pp. 161–166. (Cited on page 6.)

Poliner, G. and D. Ellis (2007). A discriminative model for polyphonic piano transcription. *JASP 2007*(1), 154–164. (Cited on pages xi, 6, 29, 47, 79 and 81.)

Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE 77*(2), 257–286. (Cited on page 2.)

Raczynski, S., E. Vincent, and S. Sagayama (2013). Dynamic Bayesian networks for symbolic polyphonic pitch modeling. *IEEE Transactions on Audio, Speech and Language Processing 21*(9), 1830–1840. (Cited on page 17.)

Raina, R., A. Battle, H. Lee, B. Packer, and A. Ng (2007). Self-taught learning: Transfer learning from unlabeled data. In *ICML*, pp. 759–766. (Cited on page 6.)

Ranzato, M., C. Poultney, S. Chopra, and Y. LeCun (2007). Efficient learning of sparse representations with an energy-based model. In *NIPS*. (Cited on page 58.)

Raphael, C. (2002). Automatic transcription of piano music. In *ISMIR*, Volume 2, pp. 13–17. (Cited on page 3.)

Richter, S., J. Thayer, and W. Ruml (2010). The joy of forgetting: Faster anytime search via restarting. In *ICAPS*, pp. 137–144. (Cited on page 78.)

Rumelhart, D., G. Hinton, and R. Williams (1986a). Learning internal representations by error propagation. In *Parallel Dist. Proc.*, pp. 318–362. MIT Press. (Cited on pages ii, iv, 1, 20, 21, 36, 43, 71, 75, 87, 92, 102, 105, 117 and 121.)

Rumelhart, D., G. Hinton, and R. Williams (1986b). Learning representations by back-propagating errors. *Nature 323*, 533–536. (Cited on page 57.)

Ryynänen, M. and A. Klapuri (2005). Polyphonic music transcription using note event modeling. In *ASPAA*, pp. 319–322. (Cited on page 81.)

Salakhutdinov, R. and I. Murray (2008). On the quantitative analysis of deep belief networks. In *ICML 25*. (Cited on page 48.)

Schmidhuber, J. (1992). Learning complex, extended sequences using the principle of history compression. *Neural Computation 4*(2), 234–242. (Cited on page 23.)

Schraudolph, N. (2002). Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation 14*(7), 1723–1738. (Cited on page 32.)

Schrauwen, B. and L. Buesing (2009). A hierarchy of recurrent networks for speech recognition. In *NIPS 21*. (Cited on pages 25 and 37.)

Schuster, M. (1999a). Better generative models for sequential data problems: Bidirectional recurrent mixture density networks. In *NIPS*, pp. 589–595. (Cited on page 33.)

Schuster, M. (1999b). *On supervised learning from sequential data with applications for speech recognition*. Ph. D. thesis, Nara Institute of Science and Technology. (Cited on page 3.)

Schuster, M. and K. Paliwal (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing 45*(11), 2673–2681. (Cited on page 22.)

Shewchuk, J. (1994). An introduction to the conjugate gradient method without the agonizing pain. Technical report, School of Computer Science, Carnegie Mellon University. (Cited on page 32.)

Siewert, U. and W. Wustlich (2007). Echo-state networks with band-pass neurons: Towards generic time-scale-independent reservoir structures. (Cited on pages 23, 54 and 61.)

Smaragdis, P. and J. Brown (2003). Non-negative matrix factorization for polyphonic music transcription. In *WASPAA*, pp. 177–180. (Cited on pages 6, 26 and 28.)

Smaragdis, P., B. Raj, and M. Shashanka (2007). Supervised and semi-supervised separation of sounds from single-channel mixtures. In *ICA*. (Cited on pages 121 and 123.)

Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. In *Parallel Dist. Proc.*, pp. 194–281. MIT Press. (Cited on pages 2, 30, 37, 71 and 89.)

Sun, D. L. and G. J. Mysore (2013). Universal speech models for speaker independent single channel source separation. In *ICASSP*. (Cited on page 125.)

Sutskever, I. (2012). *Training Recurrent Neural Networks*. Ph. D. thesis, CS Dept., U. Toronto. (Cited on pages 54, 57 and 62.)

Sutskever, I. and G. Hinton (2007). Learning multilevel distributed representations for high-dimensional sequences. In *AISTATS*, pp. 544–551. (Cited on pages 24, 33, 37 and 46.)

Sutskever, I. and G. Hinton (2010). Temporal kernel recurrent neural networks. *Neural Networks 23*(2). (Cited on page 60.)

Sutskever, I., G. Hinton, and G. Taylor (2008). The recurrent temporal restricted Boltzmann machine. In *NIPS 20*, pp. 1601–1608. (Cited on pages 25, 33, 37, 41, 42, 46, 61, 72, 117 and 120.)

Sutskever, I., J. Martens, and G. Hinton (2011). Generating text with recurrent neural networks. In *ICML*. (Cited on pages 2 and 32.)

Taylor, G., G. Hinton, and S. Roweis (2007). Modeling human motion using binary latent variables. In *NIPS 19*, pp. 1345. (Cited on pages 24, 33, 37 and 47.)

Uría, B., I. Murray, and H. Larochelle (2013). RNADE: The real-valued neural autoregressive density-estimator. In *NIPS 26*. (Cited on pages 14, 15, 125 and 129.)

Vincent, E., N. Bertin, and R. Badeau (2010). Adaptive harmonic spectral decomposition for multiple pitch estimation. *IEEE Trans. on Audio, Speech, and Lang. Proc. 18*(3), 528–537. (Cited on pages 8, 26, 29, 114 and 116.)

Vincent, E., R. Gribonval, and C. Févotte (2006). Performance measurement in blind audio source separation. *IEEE Trans. on Audio, Speech, and Lang. Proc. 14*(4), 1462–1469. (Cited on page 124.)

Vincent, P., H. Larochelle, Y. Bengio, and P. Manzagol (2008). Extracting and composing robust features with denoising autoencoders. In *ICML 25*, pp. 1096–1103. (Cited on pages 30 and 80.)

Virtanen, T. (2007). Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria. *IEEE Trans. on Acoustics, Speech, and Lang. Proc. 15*(3), 1066–1074. (Cited on pages 3, 9, 26, 114 and 116.)

Virtanen, T., A. Cemgil, and S. Godsill (2008). Bayesian extensions to non-negative matrix factorisation. In *ICASSP*. (Cited on page 116.)

Welling, M., M. Rosen-Zvi, and G. Hinton (2005). Exponential family harmoniums with an application to information retrieval. In *NIPS 17*, pp. 1481–1488. (Cited on pages 11, 13, 47 and 120.)

Wilson, K., B. Raj, and P. Smaragdis (2008). Regularized non-negative matrix factorization with temporal dependencies for speech denoising. In *INTERSPEECH*. (Cited on page 116.)

Wooller, R. and A. Brown (2005). Investigating morphing algorithms for generative music. In *Third International Conference on Generative Systems in the Electronic Arts*. (Cited on page 4.)

Yao, K., B. Peng, G. Zweig, D. Yu, X. Li, and F. Gao (2013). Recurrent conditional random fields. In *NIPS Deep Learning Workshop*. (Cited on page 23.)

Yeh, C. (2008). *Multiple Fundamental Frequency Estimation of Polyphonic Recordings*. Ph. D. thesis, Université Paris VI. (Cited on pages 6 and 26.)

Yeh, C., N. Bogaards, and A. Röbel (2007). Synthesized polyphonic music database with verifiable ground truth for multiple f0 estimation. In *ISMIR*, pp. 393–398. (Cited on page 6.)

Yeh, C. and A. Röbel (2009). The expected amplitude of overlapping partials of harmonic sounds. In *ICASSP*. (Cited on page 8.)

Zhou, R. and E. Hansen (2005). Beam-stack search: Integrating backtracking with beam search. In *ICAPS*, pp. 90–98. (Cited on page 79.)